НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
завідувач кафедри
_____ Оксана ТИМОЩУК
«__»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 "Системний аналіз"
на тему: «Аналітика трафіку з сорсу Facebook з урахуванням воронки продажів (аукціони інтернет реклами)»

Виконала:
студентка IV курсу, групи КА-94
Рабошук Ангеліна Олександрівна _____

Керівник:
Доцент, к.ф.-м.н. Барановська Леся Валеріївна _____

Консультант з норм о контролю:
Доцент, к.ф.-м.н. Статкевич Віталій Михайлович _____

Консультант з економічного розділу:
Доцент, к.е.н. Рощина Надія Василіївна _____

Рецензент:
Доцент, к.т.н., Харченко Костянтин Васильович _____

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

Київ–2023 року

Національний технічній університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

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

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«____» травня 2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Рабошук Ангеліні Олександрівні

1. Тема роботи «Аналітика трафіку з сорсу Facebook з урахуванням воронки продажів(аукціони інтернет реклами)», керівник роботи Барановська Леся Валеріївна, доцент, к.ф.-м.н., затверджені наказом по університету від «30» травня 2023 р. № 2065-с

2. Термін подання студентом роботи: 12.06.2023.

3. Вихідні дані роботи: CSW файл з даними юзерів та вірогіднстю повторного депозиту.

4. Зміст роботи: Дослідження предметної області, Аналіз факторів впливу на конверсію з встановлення додатоку в продаж, Аналіз трафіку сорсу ФБ з урахуванням воронки продажів, Економічна частина, Функціонально-вартісний аналіз

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) : презентація
6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Н.В., доцент, к.е.н.		

7. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Дослідження предметної області	01.12.2022 – 12.12.2022	Виконано
2	Виділення ключових цілей	15.12.2022 – 20.12.2022	Виконано
3	Розробка програмного забезпечення для графічного аналізу	20.01.2023 – 18.02.2023	Виконано
4	Аналітика даних	28.02.2023 – 28.03.2023	Виконано
5	Розробка програмного забезпечення	01.04.2023 – 01.06.2023	Виконано
6	Інтерація в продакшн	03.06.2023 – 08.06.2023	Виконано
7	Висновки	10.06.2023 – 11.06.2023	Виконано

Студентка Ангеліна РАБОШУК

Керівник Леся БАРАНОВСЬКА

РЕФЕРАТ

Дипломна робота: 168 с., 5 табл., 45 рис., 2 додатка, 17 джерел.

ПРОГНОЗУВАННЯ ЙМОВІРНОСТІ ДЕПОЗИТ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ГРАФІЧНИЙ АНАЛІЗ, АВТОМАТИЗАЦІЯ, FACEBOOK.

Об’єкт дослідження – застосування та автоматизації нейронних моделей для відслідковування якості трафіку і зниження витрат.

Предмет дослідження – технічна воронка продажу з використанням штучного інтелекту. Побудова маршруту трафіку.

Мета роботи – побудова і автоматизація технічної воронки для реалізації іздешевлення трафіку для просування онлайн казино.

Методи дослідження – методи штучного інтелекту: алгоритми машинного навчання, а саме, Random Forest Regresor, графічний аналіз.

Актуальність – задача оптимізації діючої системи та розробка мікро-сервісів для покращення та оптимізації роботи технічної складової воронки продажів. Зменшення ціни на отримання користувача. Збільшення якості приведеної аудиторії. Оптимізація внутрішніх виробничих процесів.

Результати роботи – створено програмний продукт, який забезпечує покращення якості та конверсійності трафіку. Додано графічно-аналітичний інструмент та інструмент для відслідковування деяких показників воронки у реальному часі.

Шляхи подальшого розвитку предмету дослідження – використання та порівняння великих об’ємів даних аудиторії з даними рекламодавців. Використання сементації аудиторії для переливу у сміжні сфери, та поглиблених сегментацій у системі пуш-сповіщень. Побудова системи порядкування цільових подій та всіх чинників, які на неї впливають, подальший розгорнутий автоматизований аналіз у реальному часі для впливу на ключові показники.

ABSTRACT

Diploma thesis: 168 p., 5 tabl., 45 fig., 2 appendices, 17 sources.

FORECASTING DEPOSIT PROBABILITY, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, GRAPHICAL ANALYSIS, AUTOMATION, FACEBOOK.

Research Object: application and Automation of Neural Models for Traffic Quality Tracking and Cost Reduction.

Research Subject: technical sales Funnel Utilizing Artificial Intelligence. Traffic Routing Construction.

Research Objective: construction and automation of a technical funnel to achieve traffic cost reduction for online Casino promotion.

Research Methods: Artificial Intelligence methods including Machine Learning algorithms, specifically Random Forest Regressor, and graphical analysis.

Relevance: optimization of the existing system and development of microservices to enhance and optimize the technical sales funnel. Reduction in user acquisition costs. Increase in the quality of the acquired audience. Optimization of internal production processes.

Research Results: a software product has been created that improves the quality and conversion rate of traffic. A graphical-analytical tool and real-time tracking tool for certain funnel indicators have been added.

Future Directions of Research: utilization and comparison of large audience data with advertiser data. Audience segmentation for expansion into related industries and advanced segmentation in the push notification system. Construction of an event sequencing system and analysis of all factors influencing it, further deployment of real-time automated analysis to impact key metrics.

ЗМІСТ

ВСТУП

Маркетинг є невід'ємною складовою будь-якого сучасного продукту. Так само не виключенням є і сфера affiliate-маркетингу. Ця досить нова та ефективна система маркетингу дозволяє досягти успіху в просуванні продукту.

Affiliate-маркетинг - це форма співпраці між компанією-постачальником продукту і партнером-маркетологом, який рекламує продукт на своєму ресурсі або каналі комунікації. Партнер отримує комісійну винагороду за кожну успішну продажу, яка залежить від умов домовленості.

Переваги affiliate-маркетингу очевидні. Постачальник продукту отримує доступ до нових аудиторій та збільшує свою обізнаність на ринку. Партнер-маркетолог, у свою чергу, має можливість заробляти гроші на рекламі продукту, не ризикуючи власним капіталом стосовно виробництва самого продукту.

Однією з найпопулярніших сфер використання affiliate-маркетингу сьогодні стали такі ніші як: гемблінг та беттінг.

Гемблінг, або азартні ігри, включає в себе різноманітні види азартних розваг, таких як казино, покер, слоти, рулетка тощо. Це сфера, в якій багато людей проявляють інтерес та бажання випробувати свою удачу. Гемблінг є великим ринком, компанії, що пропонують азартні ігри, активно використовують affiliate-маркетинг для залучення нових гравців.

Беттінг, або ставки на спорт, є ще однією популярною галуззю, де використовується affiliate-маркетинг. Компанії, які пропонують букмекерські послуги, спрямовують зусилля на привертання нових клієнтів, які бажають зробити ставки на спортивні події. Партнери-маркетологи можуть рекламувати ці букмекерські компанії на своїх веб-сайтах, блогах або соціальних мережах та отримувати комісійну винагороду за кожного нового зареєстрованого користувача або за внесені ними ставки.

У світі affiliate-маркетингу існують різні моделі роботи, включаючи Cost Per Action (CPA) та Revenue Share.

Модель CPA передбачає виплату комісії партнеру-маркетологу лише в разі певної дії, наприклад, реєстрації нового користувача або здійснення покупки. За допомогою цієї моделі компанії можуть просувати свої продукти та послуги, не ризикуючи фінансово, оскільки вони платять лише за конкретну результативну дію.

Модель Revenue Share, або відсоткова частка виручки, означає, що партнер-маркетолог отримує винагороду від певного отриманого фінансового прибутку самим продуктом, казино чи букмекерською компанією.

Для просування будь-якого з вищенаведених продуктів індустрії існує безліч маркетингових та технічних воронок, кожна з яких має свій вплив на вартість та якість залученного клієнту, які ми розглянемо пізніше.

Також варто зазначити, що на ці ж показники суттєвий вплив мають й джерела рекламного трафіку, такі як: Facebook, Google PPC, UAC, TikTok, In App, SEO, ASO та інші…

Протягом роботи дуже важливим стане вибрати правильні маркетингові та технологічні підходи для оптимізації виробництва та ціни клієнта.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження джерел трафіку та їх особливостей

Кожне джерело трафіку має свої особливості, тому для досягнення маркетингових цілей нам важливо вибрати відповідну платформу для реклами. Це дозволить нам привернути увагу правильної цільової аудиторії та збільшити кількість активних користувачів. Ретельний аналіз особливостей різних джерел трафіку допоможе гемблінг-операторам визначити найоптимальнішу стратегію рекламних кампаній та зосередитися на найефективніших каналах привертання нових клієнтів.

Далі розглянемо та оберемо одне з джерел трафіку для подальшого дослідження.

1. Facebook: Враховуючи широку аудиторію та різноманітні рекламні інструменти, Facebook стає одним з найпопулярніших каналів привертання трафіку для гемблінг-операторів. Аналіз рекламних можливостей та цільової аудиторії Facebook дозволить оцінити ефективність цього джерела трафіку в контексті гемблінг-індустрії.
2. Google PPC: Платна контекстна реклама у пошуковій системі Google дозволяє гемблінг-операторам з'явитися перед потенційними клієнтами, коли ті активно шукають відповідні гемблінг-сервіси. Вивчення особливостей Google PPC та ключових слів в контексті гемблінгу допоможе визначити найбільш ефективні стратегії рекламних кампаній.
3. UAC (Universal App Campaigns): Якщо гемблінг-оператори мають мобільні додатки, UAC від Google дозволяє автоматично просувати ці додатки в різних

рекламних каналах. Дослідження можливостей UAC дозволить визначити переваги та недоліки цього джерела трафіку в контексті гемблінг-індустрії.

4. TikTok: Як одна з найшвидше зростаючих платформ соціальних медіа, TikTok набуває популярності серед молодої аудиторії. Вивчення можливостей реклами на TikTok та її впливу на гемблінг-сферу допоможе з'ясувати, наскільки ефективною є ця платформа для привертання нових користувачів та підвищення уваги до гемблінг-брендів.

5. In App: Враховуючи зростання популярності мобільних додатків, в тому числі і гемблінг-додатків, реклама всередині додатків стає привабливим джерелом трафіку. Аналіз можливостей рекламного просування всередині додатків та його впливу на гемблінг-індустрію дозволить оцінити ефективність цього каналу трафіку та його потенціал для залучення нових користувачів.

6. SEO (Search Engine Optimization): Оптимізація пошукових систем є важливим аспектом просування гемблінг-сайтів у пошукових системах, таких як Google. Дослідження SEO-стратегій у сфері гемблінгу допоможе виявити найефективніші підходи до ранжування та підвищення видимості гемблінг-сайтів у пошукових системах.

7. ASO (App Store Optimization): Для гемблінг-операторів, які мають мобільні додатки, оптимізація в магазинах додатків, таких як App Store або Google Play, має велике значення. Вивчення ASO-стратегій та їх впливу на позиціонування гемблінг-додатків в магазинах додатків допоможе визначити найбільш ефективні підходи для залучення нових користувачів.

На мою думку, серед всіх джерел трафіку найцікавішою платформою є Facebook. Використовуючи його потужні можливості, ми можемо отримати не лише докладну аналітику щодо рекламних компаній, але й встановити різні рекламні та маркетингові цілі. Крім того, ми можемо задати різноманітні цілі для оптимізації саме на платформі Facebook.

1.2 Дослідження загальних різновидів цілей кінцевої оптимізації маркетингових воронок

Існують декілька загальноприйнятих стратегій просування продукту, кожний з яких має як свої плюси так і мінуси.

1. Реклама через мобільні Android-додатки.

Ці мобільні застосунки знаходяться безпосередньо в Google Play Market та мають відкритий доступ для будь-якого користувача.
У даному випадку використовуються мобільні додатки типу web-view.

WebView - це компонент розробки програмного забезпечення, який дозволяє вбудовувати веб-сторінки в мобільні додатки. WebView дозволяє відображати вміст веб-сторінок, такий як HTML, CSS та JavaScript, безпосередньо в мобільному додатку, що дозволяє комбінувати веб-інтерфейс зі звичайним інтерфейсом мобільних додатків. А на додаток, використовуючи спеціальну систему для аналітики та керування трафіком, кожному користувачу додатку за замовчуванням можна відкрити будь-яку сторінку будь-якого казино чи беттінг платформи.

Далі систему керування трафіком будемо називати TDS.

2. Реклама через мобільні IOS-додатки.

Створення застосунків за основною методологією дуже схоже на Android-додатки, проте має свої відмінності:

1.1 Розміщення додатку відбувається в App Store.

1.2 Трафік майже не атрибуцюється, адже IOS має досить суворі правила до обробки персональних даних.

2 Реклама через PWA.

PWA - це технологія розробки веб-додатків, яка дозволяє створювати веб-сторінки, що мають функціональність та вигляд традиційних мобільних додатків. PWA поєднує в собі переваги веб-технологій і мобільних додатків, що дозволяє користувачам отримувати багатофункціональні додатки, які можна використовувати безпосередньо через веб-браузер.

З Трафік на посилання. Досить тривіальний метод оптимізації реклами, де реклама спрямовується безпосередньо на сайт самого рекламодавця.

1.3 Збір статистичних даних при рекламі для ефективного використання AppsFlyer та Keitaro

З метою досягнення успішного впровадження рекламних кампаній і максимізації їх результативності, збір статистичних даних є невід'ємною складовою процесу реклами. В даному розділі розглядається важливість збору статистичних даних, а також розглядаються два основні інструменти для цього - AppsFlyer та Keitaro.

Важливість збору статистичних даних при рекламі

Збір статистичних даних є ключовим етапом у плануванні, впровадженні та оптимізації рекламних кампаній. Отримані дані дозволяють здійснювати обґрунтовані рішення на основі об'єктивних фактів, а не лише інтуїції. Нижче наведено деякі ключові причини, чому збір статистичних даних є необхідним для ефективної реклами.

1. Оцінка ефективності: Статистичні дані дозволяють оцінити ефективність рекламної кампанії, визначити, які канали та рекламні платформи працюють найкраще для досягнення поставлених цілей. Збір даних дозволяє виявляти успішні та неуспішні елементи кампаній і здійснювати відповідні коригуючі дії.
2. Оптимізація витрат: Детальний аналіз статистичних даних дозволяє виявити елементи рекламних кампаній, які не приносять достатньої вартості і потребують оптимізації. На основі цих даних можна здійснити перерозподіл рекламного бюджету та зосередитися на ефективніших каналах та стратегіях реклами. Збір статистичних даних допомагає визначити ROI (Return on Investment - показник ефективності витрат) кожної рекламної кампанії та забезпечити оптимальне використання ресурсів.
3. Розуміння цільової аудиторії: Збір статистичних даних дозволяє отримати більш глибоке розуміння цільової аудиторії. Дані про демографічні характеристики, поведінку користувачів, їхні вподобання та інтереси допомагають налаштувати рекламні кампанії з точністю під потреби цільової аудиторії.
4. Аналіз конкурентів: Збір статистичних даних дозволяє провести аналіз діяльності конкурентів. Це дає можливість виявити їхні сильні та слабкі сторони, успішні стратегії та помилки, що вони роблять. Аналіз конкурентів допомагає удосконалити власну рекламну стратегію та знайти ніші для подальшого розвитку.

AppsFlyer

AppsFlyer є провідною платформою аналітики та атрибуції мобільних додатків. Цей інструмент надає розширені функціональні можливості для вимірювання та аналізу рекламних кампаній на мобільних пристроях, розподілення користувачів додатків не тільки за джерелом трафіку а й за вебмайстром. Веб Майстер - це людина, яка безпосередньо запускає трафік.

Keitaro

Система керування трафіком, яка дозволяє показувати користувачу той чи інший веб сайт у додатку в залежності від: країни, мови, операційної системи, самого додатку, важелів тощо.
Також Keitaro використовується для поглибленої аналітики трафіку за понад 20 показниками, що дозволяє знайти найоптимальніший підхід для заливу трафіку.

OneSignal

OneSignal - це платформа повідомлень в реальному часі (Real-Time Messaging Platform), яка надає можливість розсилати повідомлення на різні пристрої, такі як веб-сайти, мобільні додатки та інші канали зв'язку.

OneSignal дозволяє розробникам включити функціональність сповіщень у свої додатки або веб-сайти, щоб взаємодіяти з користувачами через повідомлення. Це можуть бути сповіщення про нові повідомлення, оновлення, акції, спеціальні пропозиції та інше.

Завдяки OneSignal розробники можуть створювати різні види повідомлень, включаючи сповіщення з картинками, звуком, додатковою інформацією та іншими елементами, які допомагають залучити увагу користувачів. Платформа підтримує сегментування аудиторії, що дозволяє надсилати повідомлення лише певним групам користувачів в залежності від їхніх інтересів або дій.

OneSignal також надає аналітичні звіти, які дозволяють розробникам оцінювати ефективність їхніх повідомлень та взаємодії з користувачами. Це дає можливість вдосконалювати стратегію комунікації та забезпечувати кращий досвід для користувачів.

У загальному розумінні, OneSignal є потужним інструментом для розробників, який допомагає встановлювати зв'язок зі своєю аудиторією через повідомлення в реальному часі на різних платформах та пристроях.

Висновки до розділу 1

У рамках дослідження було проведено аналіз різних методологій реклами з точки зору контенту та технічної складової. Виявлено, що існує значна кількість таких методологій, що надають можливість ефективно комунікувати з аудиторією.

З моєї точки зору, в процесі дослідження вдалося встановити, що FaceBook є найбільш ефективним та цікавим інструментом реклами. Його гнучкість, потужність та наявність відкритого API дозволяють рекламодавцям реалізувати свої рекламні кампанії на високому рівні.

FaceBook надає широкі можливості в аспекті взаємодії зі споживачами та налаштування цільової аудиторії. Його аналітичні засоби та зручний інтерфейс допомагають рекламодавцям зрозуміти ефективність своїх кампаній та вносити необхідні корективи.

Враховуючи всі вищезазначені фактори, можна зробити висновок, що використання FaceBook як інструменту реклами на даний момент є дуже перспективним. Його можливості в поєднанні з правильною стратегією рекламної компанії можуть допомогти досягти бажаних результатів та ефективно взаємодіяти з цільовою аудиторією.

2 АНАЛІЗ ФАКТОРІВ ВПЛИВУ НА КОНВЕРСІЮ З ВСТАНОВЛЕННЯ ДОДАТКУ В ПРОДАЖ

Наведемо перелік факторів впливу на конверт з встановлення додатку в продаж.

1. Вага мобільного додатку.
2. Дизайн мобільного додатку.
3. Рейтинг та кл. негативних відгуків в сторі.
4. Перелив аудиторії.
5. Цільова країна.
6. Різкі стрибки об’єму трафіку.
7. Воронки пуш-сповіщень.
8. Вік.
9. День тижня / місяця.
10. Умови зарахування проданого користувача.

Кожен елемент переліку розкриємо у відповідному підрозділі.

2.1. Вага мобільного додатку

Вага мобільного додатку є суттєвим фактором, який впливає на конверсію з кліку по рекламному оголошенню в фактичне встановлення мобільного додатку. Якщо додаток важить забагато мегабайт або вимагає швидкого та стабільного інтернет-з'єднання, то користувачі з обмеженими можливостями інтернету або на мобільних пристроях з недостатньою пам'яттю можуть

зіткнутись з проблемами при завантаженні додатку. Це може призвести до відмови від встановлення додатку, що знижує конверсію.

У даному розділі проводиться аналіз впливу ваги мобільного додатку на конверсію. Досліджується, які конкретно параметри ваги (розмір файлу, обсяг пам'яті, кількість ресурсів, необхідних для роботи додатку) впливають на конверсію та яким чином це відображається на результативності рекламної кампанії. Для цього проводяться аналітичні дослідження та експерименти, збираються дані про вагу додатків, розподіл конверсій за категоріями ваги, а також виконується порівняльний аналіз результатів.

Також вивчаються рекомендації та методи оптимізації ваги мобільних додатків, які дозволяють знизити розмір додатку, оптимізувати використання ресурсів та забезпечити швидке завантаження. Це може включати використання компресії файлів, оптимізацію графіки та звукових ефектів, видалення непотрібних компонентів та оптимізацію коду.

Аналіз ваги мобільного додатку допомагає рекламодавцям та розробникам зрозуміти, як впливає цей фактор на конверсію та ефективність рекламних кампаній через сорс Фейсбук. Висновки з цього аналізу можуть використовуватись для удосконалення мобільних додатків, зменшення їх ваги і покращення користувацького досвіду.

Окрім того, рекомендації та методи оптимізації ваги мобільних додатків можуть бути використані для вдосконалення рекламних кампаній на платформі Фейсбук. Розуміння того, як розмір та вага додатку впливають на конверсію, дозволяє рекламодавцям приймати обґрунтовані рішення стосовно оптимізації рекламної стратегії та забезпечення кращих результатів.

2.2 Дизайн мобільного додатку

Розглядаємо вплив дизайну мобільного додатку на конверсію з кліку по рекламному оголошенню до фактичного встановлення та подальших продажів.

На початку підрозділу проводиться огляд літератури та наявних досліджень, що висвітлюють важливість дизайну мобільних додатків для користувацького досвіду та досягнення мети рекламної кампанії. Зазначаються такі аспекти, як візуальний дизайн, інтерфейс, навігація, взаємодія з користувачем та інші фактори, що впливають на сприйняття та залучення користувачів.

Далі розглядається результати проведеного дослідження впливу дизайну мобільного додатку на конверсію. Описується методика збору даних, включаючи аналіз рекламних кампаній через Фейсбук, встановлення додатку та моніторинг продажів. Застосовуються кілька метрик, таких як відсоток конверсії з кліку до встановлення, час, проведений в додатку, та кількість здійснених покупок.

Проводиться аналіз отриманих даних та виявляється зв'язок між дизайном мобільного додатку та конверсією. Аналізуємо різні елементи дизайну, такі як колірна палітра, розташування елементів, типографіка, використання візуальних ефектів та інші аспекти, які можуть впливати на сприйняття та залучення користувачів.

На основі отриманих результатів аналізу дизайну мобільного додатку, було зроблено наступні висновки.

1. Візуальний дизайн: використання привабливих кольорових схем та естетично збалансованих елементів дизайну позитивно впливає на сприйняття та залучення користувачів. Мобільний додаток з привабливим візуальним

оформленням має більше шансів залучити увагу потенційних клієнтів і збільшити конверсію.

2. Інтерфейс та навігація: зручний та інтуїтивно зрозумілий інтерфейс додатку зробить його використання зручним для користувачів. Ефективна навігація та простота використання допоможуть зберегти увагу користувачів та підвищити шанси на встановлення та подальші продажі.

3. Взаємодія з користувачем: важливо враховувати зручність взаємодії користувача з додатком. Інтуїтивність та логічність дій, наявність кнопок та елементів, що викликають бажану реакцію від користувача, сприяють поліпшенню користувацького досвіду та збільшують ймовірність продажів.

4. Відповідність меті додатку: дизайн мобільного додатку повинен відповідати його меті та сприяти досягненню конкретних цілей рекламної кампанії. Наприклад, якщо головною метою додатку є продаж певного товару, то дизайн повинен зосередитись на показі користувачеві цього товару та зручному замовленні.

Загальним висновком є те, що дизайн мобільного додатку має значний вплив на конверсію з кліку по рекламному оголошенню до фактичного встановлення додатку та подальших продажів. Оптимізація дизайну може сприяти поліпшенню користувацького досвіду та збільшенню ефективності рекламних кампаній.

Врахування візуального оформлення, інтерфейсу та навігації, зручності взаємодії з користувачем та відповідності меті додатку становить ключові фактори успіху. Важливо проводити тестування та аналіз результатів для виявлення найбільш ефективних елементів дизайну та вдосконалення додатку.

2.3 Рейтинг та клієнтських негативних відгуків в сторі

У пункті 2.3 дослідження, було проаналізовано вплив рейтингу та клієнтських негативних відгуків в магазинах додатків на конверсію з кліку по рекламному оголошенню до встановлення додатку та подальших продажів. Я звернула увагу на наступні аспекти:

Рейтинг: рейтинг додатку є одним із ключових показників його якості та надійності для потенційних користувачів. Високий рейтинг може підвищити довіру до додатку та спонукати користувачів до його встановлення. Був врахований рейтинг, оцінки та коментарі користувачів, які допомагають створити уявлення про якість та задоволення, отримане від використання додатку.

Негативні відгуки: наявність негативних відгуків у магазинах додатків може відлякувати потенційних користувачів. Люди часто звертаються до відгуків, щоб оцінити досвід інших користувачів та визначити, наскільки добре додаток задовольняє їхні потреби. Я звернула увагу на негативні відгуки, що стосуються швидкості, стабільності, функціональності або обслуговування, оскільки вони можуть вплинути на конверсію та створити негативне враження про додаток.

З моїх досліджень випливає, що рейтинг та клієнтські відгуки є важливими факторами впливу на конверсію та сприяють будуванню довіри до додатку. Я рекомендую активно взаємодіяти з користувачами, постійно вдосконалювати функціональність додатку та швидко вирішувати їхні проблеми. Позитивна комунікація з користувачами, відповіді на їх запити та зворотний зв'язок допоможуть зберегти їхню довіру та зацікавленість у додатку. Постійне вдосконалення функціональності, враховуючи відгуки користувачів та їхні потреби, допоможе покращити якість взаємодії та забезпечити позитивний

досвід використання. Реагування на виявлені недоліки, усунення помилок та швидке вирішення проблем, згаданих у негативних відгуках, допоможуть зменшити їх вплив на конверсію. За допомогою постійного вдосконалення та активної взаємодії з користувачами, ми можемо побудувати позитивну репутацію додатку та сприяти його успіху.

2.4 Перелив аудиторії

Перелив аудиторії - це процес перенесення внутрішніх ідентифікаторів мобільних пристроїв у поєднанні із зазначенням ключових подій оптимізації для подальшого мапінгу у межах рекламної мережі Facebook.

Тобто, з додатка “Додаток 1” збираємо інформацію, що юзери з ідентифікаторами: 111, 222, 333, 444 зробили реєстрації чи депозити, після цього серед них обираємо потрібні, механіку розглянемо далі. Наступним кроком, за допомогою postback, передаємо ці ідентифікатори у Facebook, зазначаючи ідентифікацію “Додатку 2”, користувача і потрібні події. Саме цей процес називається “Перелив аудиторії”.

Ціль, яка переслідується - це надати рекламній мережі Facebook стартові дані для оптимізації у поєднанні: мобільний додаток - країна - група ідентифікаторів користувачів, які виконали цільову подію. Саме таке поєднання вказує нейромережі Facebook, яка надалі буде підбирати аудиторію якій показувати рекламу при використанні запуску на “Еквазішн”, які саме користувачі потрібні.

Постбек (англ. postback) - це технологічний механізм, що дозволяє отримати повідомлення або повідомлення про подію або дію, яка відбулась на сторонньому ресурсі (у цьому випадку - Facebook), і передати цю інформацію до сервера або системи, що здійснює оптимізацію рекламних кампаній.

У контексті реклами мобільних додатків на Facebook, постбеки дозволяють відстежувати конкретні дії або події користувачів, які взаємодіють з рекламою додатку. Наприклад, це може бути реєстрація - lead, чи депозит (покупка в мобільному додатку) - sale. За допомогою постбеків, рекламні системи можуть отримати інформацію про ці події і використовувати її для оптимізації рекламних кампаній.

2.5 Цільова країна

Вибір цільової країни має значний вплив на успіх компанії, оскільки різні країни мають різний менталітет, фінансову забезпеченість, звички користування мобільними додатками та інші фактори, які впливають на конверсію та поведінку користувачів.

У цьому розділі зконцентруємо увагу на наступних аспектах.

1. Дослідження ринку: проведено аналіз ринку країн, в яких ми плануємо запускати рекламну кампанію. Для цього ми вивчили соціально-економічні показники, такі як ВВП на душу населення, рівень споживчої активності, популярність мобільних додатків та інші статистичні дані.
2. Менталітет та культурні особливості: вивчено менталітет та культурні особливості цільової країни. Це допомагає нам розуміти, які цінності, інтереси та потреби мають користувачі цієї країни. Ми адаптуємо нашу рекламну стратегію та контент з урахуванням цих особливостей.
3. Мова та локалізація: встановлено, чи необхідна локалізація нашого додатку та рекламного контенту для кожної цільової країни. Це включає переклад і адаптацію текстів, зображень, інтерфейсу додатку та інших

елементів, що забезпечують більшу зрозумілість та привабливість для місцевих користувачів.

4. Географічні налаштування: оцінено географічний охоплення рекламної кампанії в різних країнах та регіонах. Ми враховуємо географічні особливості, такі як розміщення цільової аудиторії, конкуренцію на ринку та цінову політику для визначення найбільш ефективних географічних зон для розміщення реклами.

5. Фінансова забезпеченість: аналізується фінансова забезпеченість цільової аудиторії в різних країнах. Враховуються середні доходи населення, рівень безробіття та витрати на мобільні додатки. Це допомагає встановити оптимальні цінові стратегії та пропозиції для різних ринків.

6. Юридичні та регуляторні вимоги: вивчено юридичні та регуляторні вимоги для реклами та мобільних додатків в кожній цільовій країні. Дотримання цих вимог є важливим аспектом успішної рекламної кампанії та збереження репутації бренду.

На основі отриманих результатів аналізу цільових країн, ми розробляємо стратегію таргетингу та оптимізації рекламної кампанії для кожної окремої країни. Ми налаштовуємо наші рекламні канали, вибираємо відповідну мову, контент та пропозиції, щоб максимально залучити цільову аудиторію та забезпечити успішність рекламної кампанії у кожній країні.

2.6 Різкі стрибки об’єму трафіку

У даному розділі проведено аналіз та дослідження впливу різкого зростання обсягу трафіку на ефективність рекламної кампанії та конверсію встановлення додатку в продаж.

1. Збільшення обсягу трафіку: вивчено різкі стрибки в обсязі трафіку, які відбуваються під час рекламних кампаній. Аналізується причина цих стрибків, такі як великі розміри аудиторії, популярність оголошень або спеціальні пропозиції.
2. Вплив на конверсію: досліджено вплив різкого збільшення трафіку на конверсію з кліку на рекламне оголошення до встановлення додатку. Встановлено, що при різкому зростанні в обсязі трафіку, показник конверсії може зменшитись.
3. Реакція системи: вивчено реакцію системи на різке збільшення трафіку та його вплив на швидкість завантаження сторінок, функціонування додатку та користувацький досвід. Аналізуються можливі проблеми, які можуть виникнути внаслідок перевантаження системи та шляхи їх вирішення.
4. Стратегії оптимізації: розроблено стратегії оптимізації рекламної кампанії для ефективного управління різкими стрибками трафіку. Включаються такі заходи, як розподіл рекламного бюджету, управління розкладом показу оголошень, оптимізація рекламних матеріалів та використання автоматизованих систем управління трафіком.
5. Моніторинг та аналітика: встановлено систему моніторингу та аналітики для постійного відстеження різких стрибків обсягу трафіку. Застосовуються інструменти аналізу даних, які дозволяють виявляти тенденції, виявляти причини стрибків та оцінювати їх вплив на конверсію та ефективність кампанії.
6. Вдосконалення системи: на основі отриманих результатів проведеного дослідження різких стрибків обсягу трафіку розроблено пропозиції щодо вдосконалення системи управління трафіком та рекламними кампаніями. Включаються такі аспекти, як оптимізація інфраструктури, розширення ресурсів, вдосконалення алгоритмів таргетингу та біддінгу.

7. Рекомендації щодо стратегій: на основі проведеного аналізу різких стрибків обсягу трафіку розроблено рекомендації щодо використання стратегій управління трафіком. Зокрема, розглядаються можливі підходи до розподілу бюджету, контролю за часом показу оголошень, використання автоматизованих систем оптимізації та розробка гнучких стратегій реагування на різкі зміни обсягу трафіку.
8. Переваги та виклики: обговорено переваги та виклики, пов'язані з різкими стрибками обсягу трафіку. Зокрема, розглядаються можливості прискореного збільшення показників, розширення аудиторії та залучення нових користувачів, а також виклики, пов'язані зі збільшеним навантаженням на систему, можливістю погіршення користувацького досвіду та ефективності кампанії.

2.7 Воронки пуш-сповіщень

Проведено детальний аналіз використання пуш-сповіщень та їх вплив на ефективність взаємодії з користувачами та досягнення поставлених цілей. Розглянуто основні аспекти цього пункту.

1. Визначення цілей: у цьому підрозділі були визначені основні цілі використання пуш-сповіщень, такі як збільшення залучення користувачів, підвищення взаємодії з додатком, підвищення ретенції та конверсії.
2. Сегментація аудиторії: проведений аналіз сегментації аудиторії для ефективного використання пуш-сповіщень. Визначено ключові характеристики та поведінкові ознаки користувачів, що дозволяють налаштовувати сповіщення для різних груп користувачів.

3. Розробка воронок: розроблено воронки пуш-сповіщень, що представляють послідовність повідомлень, спрямованих на досягнення конкретних цілей. Визначено етапи воронки та зміст сповіщень на кожному етапі, що допомагає ефективно керувати комунікацією з користувачами.
4. Тестування та оптимізація: проведено тестування різних варіантів пуш-сповіщень для визначення найефективніших комбінацій. Аналізуються показники відкриття, конверсії та взаємодії з пуш-сповіщеннями для виявлення можливостей оптимізації та покращення результатів.
5. Аналіз результатів: здійснено аналіз результатів використання пуш-сповіщень, включаючи відкриття, конверсії та взаємодію з пуш-сповіщеннями. Визначено ключові метрики успішності, такі як частота відкриття сповіщень, частота конверсії, відсоток відписок та активність користувачів після отримання пуш-сповіщень.
6. Персоналізація та релевантність: удосконалено підхід до персоналізації та надання релевантного контенту в пуш-сповіщеннях. Досліджено вплив персоналізованих повідомлень на ефективність та задоволеність користувачів. Розроблено стратегії, які дозволяють враховувати індивідуальні потреби та інтереси користувачів при формуванні пуш-сповіщень.
7. Автоматизація та розклад сповіщень: розроблено систему автоматизованого управління пуш-сповіщеннями, яка дозволяє планувати та налаштовувати розклад сповіщень для оптимального впливу на користувачів. Використовуються інструменти автоматичного відправлення, аналізу та оптимізації пуш-сповіщень.
8. Рекомендації та підсумки: на основі проведеного дослідження розроблено рекомендації щодо використання пуш-сповіщень для досягнення поставлених цілей. Висвітлено переваги та потенційні виклики цього інструменту комунікації з користувачами. Зроблено підсумок результатів використання

пуш-сповіщень та наведено пропозиції щодо подальшого вдосконалення та оптимізації стратегій використання пуш-сповіщень.

9. Заключні висновки: проведено дослідження та аналіз використання пуш-сповіщень з метою підвищення ефективності взаємодії з користувачами та досягнення поставлених цілей. Виявлено, що правильно налаштовані воронки пуш-сповіщень можуть значно покращити результативність рекламної кампанії та сприяти залученню нових користувачів, збільшенню конверсії та поліпшенню ретенції.

Застосування воронок пуш-сповіщень дозволяє більш ефективно спрямовувати комунікацію з користувачами на різних етапах їх взаємодії з додатком чи платформою. Це включає в себе надсилання вітальних повідомлень для нових користувачів, сповіщень з персоналізованими пропозиціями, нагадувань про неактивність або пропозицій відновлення використання додатку для підтримання наявних користувачів, а також спеціальних пропозицій або акцій для стимулювання конверсії.

Процес воронки пуш-сповіщень передбачає не тільки налаштування та відправку повідомлень, але й систематичний аналіз результатів. Проведення А/Б-тестування різних варіантів повідомлень, вимірювання показників успішності та взаємодії з користувачами, аналіз отриманих даних та вдосконалення стратегій допомагають досягти кращих результатів.

Крім того, важливим аспектом воронок пуш-сповіщень є їх релевантність та відповідність інтересам та потребам користувачів. Шляхом збору та аналізу даних про поведінку користувачів, їхніх уподобань та демографічних характеристик можна персоналізувати пуш-сповіщення для кожного користувача. Це означає, що повідомлення можуть містити індивідуальні рекомендації, спеціальні пропозиції або знижки, які відповідають їхнім інтересам та попередній взаємодії з додатком. Персоналізовані пуш-

сповіщення демонструють вищу ефективність і здатні залучати більше уваги користувачів.

З метою автоматизації та покращення ефективності використання пуш-сповіщень розроблена система управління, яка дозволяє планувати, налаштовувати та автоматично відправляти повідомлення відповідно до встановленого розкладу або певних подій. Це допомагає забезпечити своєчасну доставку сповіщень та зменшити необхідність ручної роботи.

В процесі роботи з воронками пуш-сповіщень слід також звернути увагу на збільшення активності користувачів. На підставі аналізу результатів можна виявити можливості для покращення взаємодії, включаючи збільшення частоти відкриття сповіщень, залучення до додатку, збільшення часу використання та конверсії.

Загалом, воронки пуш-сповіщень є потужним інструментом для підвищення ефективності комунікації з користувачами та досягнення поставлених цілей. Їх правильне використання, поєднане з аналізом результатів та вдосконаленням стратегій, дозволяє покращити взаємодію з користувачами та досягти успіху в рекламних кампаніях.

2.8 Вік

У даному дослідженні був вивчений вплив віку користувачів на конверсію та продажі у мобільному додатку. Вік є важливим фактором, оскільки різні вікові групи мають відмінності у поведінці, потребах та звичках.

Починаючи зібраними даними, було проведено аналіз та порівняння конверсії та продажів серед різних вікових груп. В результаті були виявлені наступні спостереження:

1. Молодша вікова група (18-24 роки): ця група користувачів часто виявляє високий рівень зацікавленості у нових технологіях та мобільних додатках. Вони активні та відкриті до експериментування з новими продуктами. Однак, конверсія з кліку на встановлення додатку до фактичного продажу може бути меншою через відсутність фінансової стійкості або відсутність власних засобів.
2. Середня вікова група (25-34 роки): ця група представляє собою значну частину активних користувачів мобільних додатків. Вони мають стабільний фінансовий стан і готовність інвестувати у якісні продукти та послуги. Конверсія у цій віковій групі може бути високою, особливо якщо пропозиція додатку відповідає їхнім потребам та інтересам.
3. Старша вікова група (35+ років): ця група користувачів може бути менш активною у встановленні та використанні мобільних додатків порівняно з молодшими групами. Однак, вони можуть бути більш стійкими та лояльними користувачами, які здійснюють покупки з регулярністю. Важливо забезпечити простоту використання додатку та зрозумілість його функціоналу для цієї вікової групи. Також слід враховувати, що розміщення реклами та комунікація з користувачами можуть вимагати іншого підходу, щоб привернути їх увагу та збільшити конверсію.

Для покращення результатів у вікових сегментах застосовано наступні стратегії.

1. Сегментація рекламних кампаній: рекламні кампанії повинні бути націлені на конкретні вікові групи, враховуючи їхні інтереси, потреби та покупний потенціал. Це дозволить оптимізувати бюджет та забезпечити більшу конверсію.
2. Персоналізація досвіду користувача: враховуючи вікові особливості, можна надати користувачам індивідуальний підхід. Це може включати

персоналізовані рекомендації, спеціальні пропозиції та знижки, які відповідають їхнім потребам та вподобанням.

3. Тестування та оптимізація: проведення A/B тестування дозволяє визначити найефективніші стратегії та канали комунікації для кожної вікової групи. Це допомагає зрозуміти, які зміни та покращення можуть позитивно вплинути на конверсію та продажі.

4. Зворотний зв'язок та взаємодія: важливо створити механізми для збору зворотного зв'язку від користувачів різних вікових груп. Це може бути через опитування, огляди, коментарі тощо. Зворотний зв'язок допомагає виявити потреби, проблеми та очікування користувачів різних вікових груп. На основі цієї інформації можна розробити вдосконалення та нові функції, що задовольняють їхні потреби та покращують їхній досвід використання додатку.

5. Навчання та підтримка: враховуючи вікові особливості, слід надати зрозумілі та детальні інструкції щодо використання додатку. Також можна запровадити підтримку через онлайн-чат, електронну пошту або телефон, де спеціалісти можуть допомогти користувачам у вирішенні їхніх проблем та відповісти на питання.

6. Вдосконалення інтерфейсу та взаємодії: для різних вікових груп варто враховувати особливості їхньої зорової сприйнятливості та зручності використання. Можна впровадити великі, зрозумілі та прості елементи управління, зручні розташування кнопок та інтуїтивно зрозумілі іконки.

7. Аналіз даних та вдосконалення: слід постійно аналізувати дані щодо використання додатку користувачами різних вікових груп. Це дозволить виявити тренди, проблемні місця та можливості для покращення. На основі цього аналізу можна вносити зміни, вдосконалювати функціонал та пристосовувати додаток до потреб користувачів різного віку.

Застосування цих стратегій та підходів дозволить більш ефективно привернути та утримати користувачів різних вікових груп, забезпечити зростання конверсії та покращення продажів у віковому сегменті. Вік є лише одним з факторів, що впливають на поведінку користувачів. Враховуємо й інші аспекти, такі як географічне положення, інтереси, статус.

Наводимо наступні рекомендації з метою оптимізації досвіду користувача в різних вікових групах.

1. Аналізувати поведінку користувачів за віковими сегментами: збирати дані про використання додатку від користувачів різних вікових груп та проводити ретельний аналіз поведінки. Слід звернути увагу на популярність функцій, тривалість сесій, частоту використання та інші показники.
2. Персоналізувати контент та пропозиції: використовувати дані про вікові групи для налаштування персоналізованого контенту та пропозицій. Зробити акцент на тих функціях, які є особливо привабливими для певної вікової групи та відповідають їхнім потребам та інтересам.
3. Полегшити навігацію та використання: приділити особливу увагу зрозумілості та простоті використання додатку. Впевнитися, що інтерфейс є інтуїтивно зрозумілим, кнопки та функції розташовані зручно для користувачів різного віку.
4. Забезпечити підтримку та навчання: забезпечити користувачів підтримкою та навчанням, якщо вони мають питання або проблеми з використанням додатку. Забезпечити доступ до контактної інформації, довідкової документації, онлайн-порад та інструкцій.
5. Пропонувати інтерактивні елементи та гейміфікацію: для привертання уваги користувачів різних вікових груп можна використовувати елементи гейміфікації, такі як досягнення, лідерборди, бейджі, виклики тощо. Це дозволить зробити використання додатку більш захоплюючим та залучити користувачів до активної взаємодії з ним.

6. Враховувати фізіологічні особливості: різні вікові групи можуть мати різні фізіологічні особливості, такі як зорова чутливість, моторика рук, аудіальна сприйнятливість тощо. Потрібно враховувати ці особливості при розробці інтерфейсу та взаємодії з додатком.
7. Застосовувати тестування та зворотний зв'язок: проводити тестування залучених користувачів різних вікових груп для оцінки їхнього досвіду використання та збору зворотного зв'язку. Це допоможе виявити можливі проблеми та здобути ідеї для подальшого покращення.

Враховуючи вікові особливості користувачів, розробка та впровадження відповідних стратегій та підходів допоможе забезпечити кращу конверсію, задоволеність користувачів та збільшення продажів у різних вікових сегментах. Важливо продовжувати вивчати та аналізувати дані для постійного вдосконалення додатку та забезпечення високої якості користувальницького досвіду.

2.9 День тижня / місяця

Вплив дня тижня та місяця на ефективність рекламного трафіку та конверсію встановлення додатку є важливим аспектом аналізу. Врахування цих факторів дозволяє оптимізувати рекламні кампанії та маркетингові зусилля, забезпечуючи більшу успішність просування додатку на ринку.

День тижня: дослідження показують, що різні дні тижня можуть мати вплив на активність користувачів і результати рекламних кампаній. Наприклад, в деяких випадках вихідні дні можуть мати вищу активність та більшу ймовірність конверсії, оскільки багато людей вільні в цей час та мають більше часу на дослідження та завантаження додатків. З іншого боку, будні дні

можуть бути ефективними для специфічних аудиторій, наприклад, для професіоналів, які використовують додатки на робочих місцях. Важливо аналізувати дані та визначити оптимальні дні тижня для запуску рекламних кампаній та спрямування бюджету на найбільш результативні періоди.

Місяць: сезонність та коливання попиту також можуть впливати на ефективність рекламного трафіку. Деякі додатки можуть бути популярнішими в певні місяці або відображати пікову активність під час особливих подій і свят. Наприклад, додатки, пов'язані з подорожами або туризмом, можуть мати більший попит влітку або під час великих свят. Дослідження місячної активності та зворотній зв'язок від користувачів допомагають ідентифікувати оптимальні періоди для рекламних кампаній. Наприклад, під час знижок або розпродажів попит на додатки може зростати, що робить цей період привабливим для активності у рекламі. Також, спеціальні події або сезонність можуть впливати на відгуки та рейтинги додатків, що в свою чергу впливає на сприйняття користувачами та їх готовність встановити додаток.

Аналіз місячної динаміки дозволяє виявити залежності між рекламними зусиллями, активністю користувачів та конверсією. Наприклад, може бути помітно, що під час певних місяців активність користувачів зростає, або навпаки, спостерігається зниження активності. Це дає змогу налаштувати рекламні кампанії, зміщуючи бюджети та інтенсивність просування у більш перспективні періоди, коли спостерігається вища конверсія.

Врахування дня тижня та місяця у плануванні та оптимізації рекламних кампаній допомагає досягти кращих результатів, ефективно використовувати рекламний бюджет та налагоджувати зв'язок зі специфічними аудиторіями. Аналіз даних, моніторинг та аналітика дозволяють виявити закономірності та оптимізувати стратегію рекламних кампаній з урахуванням дня тижня та місяця, що сприяє збільшенню конверсії та покращенню результатів.

День тижня: проведений аналіз показав, що різні дні тижня мають варіації в активності користувачів та ефективності рекламних кампаній. Наприклад, у будні дні, коли багато людей працюють або вчаться, може спостерігатись знижена активність та менша ймовірність конверсії. З іншого боку, вихідні дні, такі як субота та неділя, можуть мати вищу активність користувачів, оскільки люди мають більше вільного часу та можуть бути більш зацікавлені в пошуку нових додатків.

Місяць: аналіз впливу місяця на рекламний трафік показав, що певні місяці можуть мати вищу активність та більшу конверсію встановлення додатку. Наприклад, під час святкових періодів, таких як Різдво або Новий рік, може спостерігатись збільшений попит на додатки, пов'язані з подарунками, розвагами та плануванням. Також, вплив сезонності, наприклад, літній сезон або шкільні канікули, може мати значний вплив на активність користувачів та конверсію.

Врахування дня тижня та місяця при плануванні та оптимізації рекламних кампаній є важливим кроком для досягнення кращих результатів. На основі проведеного аналізу, можна визначити оптимальні дні та місяці для розміщення реклами, налаштувати бюджетні виділення та рекламні зусилля залежно від активності користувачів та їх конверсії. Наприклад, якщо в будні дні спостерігається менша активність та конверсія, можна зменшити бюджет або перерозподілити його на більш результативні вихідні дні. Також, можна спрямовувати рекламні кампанії на місяці з вищим потенціалом конверсії, наприклад, під час святкових періодів або сезону пік.

Важливо продовжувати моніторити та аналізувати дані про активність користувачів у залежності від дня тижня та місяця. Це дозволить виявити можливі зміни в поведінці та інтересах аудиторії та вчасно реагувати на них. Розробка гнучкої стратегії рекламних кампаній, що враховує ці фактори,

сприятиме підвищенню ефективності просування додатку та залученню більшої кількості цільової аудиторії.

Таким чином, аналіз дня тижня та місяця дозволяє зрозуміти, коли саме користувачі більш схильні взаємодіяти з рекламними пропозиціями та встановлювати додаток. Це дає змогу планувати та розміщувати рекламу в оптимальні моменти, забезпечуючи максимальну конверсію та досягаючи мети просування додатку.

2.10 Умови зарахування проданого користувача

Умови зарахування визначають спосіб визнання та відліку успішно здійснених продажів та впливають на показники конверсії та прибутковості кампаній.

Для визначення умов зарахування проданого користувача, необхідно враховувати кілька аспектів. Перш за все, важливо встановити чіткі критерії того, що вважається продажем. Це може бути оплата за товар або послугу, підписка на платний контент, реєстрація на платформі, тощо. Необхідно врахувати особливості вашого додатку та бізнес-моделі для визначення цих критеріїв.

Далі, потрібно визначити, які джерела трафіку Фейсбук будуть призначені за зарахуванням продажу. Це можуть бути прямі кліки з рекламних оголошень на Фейсбук, або можливо, ви враховуєте інші джерела трафіку, такі як органічний пошук або реферальний трафік. Належить чітко визначити, які джерела трафіку заслуговують на зарахування продажу та як вони будуть враховуватись в аналізі.

Крім того, умови зарахування можуть включати інші фактори, такі як часові обмеження, правила повернення товару або відмови від послуги, врахування повторних продажів та інші параметри, які відображають реальну прибутковість продажів. Наприклад, якщо у вашому бізнесі часто відбуваються повторні покупки та продажі після першої угоди, важливо враховувати ці фактори при визначенні умов зарахування.

Окрім самого факту продажу, умови зарахування можуть також включати врахування додаткових параметрів, які відображають якість та цінність продажу. Наприклад, ви можете враховувати середній чек покупки, частоту покупок, участь у промоційних акціях або програмах лояльності, щоб з'ясувати, які продажі є найбільш вигідними та сприяють довгостроковому успіху бізнесу.

У дослідженні та аналізі умов зарахування проданого користувача важливо зосередитися на зборі та аналізі достовірних даних. Використовуючи відповідні інструменти аналітики та системи відстеження, необхідно зібрати інформацію про продажі та інші ключові метри, які допоможуть визначити ефективність рекламної кампанії та оцінити прибутковість.

Для аналізу умов зарахування можна використовувати різні методи, включаючи статистичний аналіз, порівняння показників у різних груп користувачів, A/B тестування та інші аналітичні підходи. Такий підхід дозволяє отримати глибоке розуміння процесу продажів та впливу різних факторів на конверсію та прибутковість.

В заключенні пункту 2.10, необхідно виділити рекомендації щодо оптимізації умов зарахування проданого користувача. Це можуть бути рекомендації щодо зміни критеріїв продажу, врахування додаткових параметрів і відмінностей в розрахунку конверсії залежно від джерела трафіку. Наприклад, якщо було помічено, користувачі, які приходять з певного джерела

трафіку, мають вищу конверсію та більший середній чек, можливо, варто збільшити вагу цього джерела при зарахуванні продажів.

Крім того, важливо постійно моніторити та оновлювати умови зарахування відповідно до змін в бізнес-середовищі та стратегії компанії. Проводити регулярні аналізи, перевіряти ефективність встановлених умов та робити відповідні корективи для досягнення максимальної точності та об'єктивності вимірювання продажів.

Результатом детального дослідження та аналізу умов зарахування проданого користувача будуть рекомендації щодо оптимізації рекламних кампаній та стратегії залучення нових користувачів. Це дозволить підвищити ефективність маркетингових зусиль, збільшити конверсію та прибутковість бізнесу.

Отже, "Умови зарахування проданого користувача" є важливою складовою аналітики трафіку сорсу Фейсбук. Визначення чітких критеріїв продажу, врахування джерел трафіку, встановлення додаткових параметрів та постійний моніторинг допоможуть точно виміряти ефективність рекламної кампанії та визначити найбільш прибуткові канали залучення користувачів. Рекомендації щодо оптимізації умов зарахування сприятимуть покращенню результатів маркетингових зусиль і досягненню більшої конкурентоспроможності та досягненню поставлених цілей.

Окрім умов зарахування проданого користувача, також варто враховувати можливість встановлення і прогнозування цілей продажів для майбутніх періодів. Аналізуючи дані про попередні продажі та прогнозуючи тенденції ринку, можна визначити амбіційні, але досяжні цілі для підвищення обсягу продажів. Це може включати встановлення цілей на місячну, квартальну або річну основу, залежно від специфіки бізнесу та його циклів.

Для ефективного контролю та досягнення цілей продажів, необхідно розробити систему моніторингу та звітності. Ця система має включати

метрики та ключові показники продажів, які будуть відстежуватись регулярно і порівнювати з поставленими цілями. Також важливо аналізувати причини відхилень від цілей та приймати відповідні корективи для покращення результатів.

З метою покращення ефективності продажів, можна розглянути впровадження стратегій стимулювання продажів, таких як промоційні акції, знижки, програми лояльності тощо. Враховуючи відповідні метрики та аналізуючи результати, можна визначити ефективність таких стратегій та внести необхідні зміни для їх оптимізації.

Необхідно також встановити відповідальність та комунікаційні ланцюжки між відділами, відповідальними за продажі та аналітику. Це допоможе забезпечити правильний обмін інформацією, вчасний аналіз та прийняття рішень для досягнення максимальних результатів продажів. Комунікація між відділами може відбуватися через регулярні збори, звітність та обговорення прогресу. Такий підхід допоможе забезпечити взаєморозуміння, обмін кращими практиками та швидке реагування на зміни у ринкових умовах.

Визначення чітких критеріїв продажу, встановлення цілей, розробка системи моніторингу та звітності, використання стратегій стимулювання продажів та забезпечення ефективної комунікації між відділами є ключовими компонентами для досягнення успіху у продажах. Постійний аналіз та внесення відповідних коректив допоможуть підвищити ефективність рекламних кампаній, залучення користувачів та досягнення прибутковості бізнесу.

Висновки до розділу 2

З дослідження випливає, що елементи технічного та демографічного характеру мають таку саму важливість, як і маркетингові аспекти. Кожен з цих елементів впливає на конверсію від установки додатку до здійснення продажу. Тому, кожен параметр має свою значущість і не може бути знехтуваним. Навіть невеликі зміни в кожному з цих параметрів можуть мати суттєвий вплив на загальний рівень конверсії, оскільки навіть піввідсотка може перетворитися на 5-10%, що є критичним для бізнесу.

Загалом, аналіз трафіку сорсу Фейсбук з урахуванням воронки продажів підтверджує, що успішність конверсії залежить від різних факторів, включаючи технічні, маркетингові, демографічні та контекстуальні аспекти. Для досягнення оптимальних результатів необхідно уважно аналізувати ці параметри, вдосконалювати стратегії та використовувати нові технології.

У наступному розділі роботи ми зосередимося на переносі аудиторії та використанні нейронних мереж та пуш-повідомлень для цього процесу.

3 АНАЛІЗ ТРАФІКУ СОРСУ ФБ З УРАХУВАННЯМ ВОРОНКИ ПРОДАЖІВ

Далі виділимо ключові зони ефективності впливу на ціну трафіку, побудуємо деякі інструменти аналітики а також покращення його якості.

Перед розгляданням механіки переносу аудиторії, розглянемо ключову ціль цієї дії.

3.1. Секція мапінг аудиторії для мобільного додатку у Facebook

Процесс мапінгу аудиторії базується на зборі та аналізі даних про користувачів, які взаємодіють з додатком. Для реалізації даної функціональності використовується SDK Facebook, що є інструментом розробки для мобільних платформ, зокрема Android і iOS.

У наступному описано етапи роботи з мапінгом аудиторії у Facebook для мобільних додатків:

1. Встановлення SDK Facebook: у рамках розробки мобільного додатку необхідно включити SDK Facebook до його коду. Цей набір інструментів дозволяє додатку взаємодіяти з платформою Facebook та передавати необхідні дані про користувачів.
2. Ідентифікація користувача: під час першого запуску додатку SDK Facebook надає унікальний ідентифікатор користувача (Facebook User ID) для поточного пристрою. Цей ідентифікатор пов'язується з профілем користувача на платформі Facebook.

3. Збір подій: розробники мають змогу визначати конкретні події, які відбуваються в додатку, та передавати їх до Facebook. Наприклад, це можуть бути події, пов'язані з реєстрацією користувача, виконанням певних дій, здійсненням покупок тощо. Кожна подія має свій унікальний ідентифікатор, який дозволяє розробникам категоризувати дії користувачів.

4. Побудова аудиторії: після отримання даних від мобільного додатку, Facebook проводить обробку та аналіз, що дозволяє створювати аудиторію залежно від різних факторів. Основні з них включають демографічні характеристики, такі як вік, стать та місцезнаходження користувачів. Крім того, враховуються інтереси та поведінка користувачів, які дозволяють точно налаштовувати спрямовану рекламу.

5. Розміщення реклами: за допомогою зібраної аудиторії розробники мобільного додатку можуть створювати таргетовані рекламні кампанії. Facebook надає можливість точно вибирати аудиторію для показу рекламних оголошень залежно від різних критеріїв, що включають демографічні дані, інтереси та поведінку користувачів. Це дозволяє максимізувати ефективність рекламної кампанії та забезпечувати зворотний зв'язок з цільовою аудиторією.

Загально кажучи, мапінг аудиторії для мобільного додатку у Facebook передбачає використання SDK Facebook, щоб збирати дані про користувачів та їх дії у додатку, а потім на основі цих даних створювати та налаштовувати аудиторію для ефективної рекламної стратегії. Цей процес дозволяє розробникам забезпечити більш персоналізоване та цілеспрямоване взаємодію з користувачами додатку на платформі Facebook, а у випадку розробки великої кількості мобільних додатків під одну й ту саму аудиторію мапінг можливо викликати штучно, передаючи аудиторію зібрану в одному мобільному додатку в інший.

Саме ця технічна можливість Facebook дозволяє від самого початку показувати рекламу максимально цільовій аудиторії.

3.2. Дві механіки роботи з рекламою додатків

Дві механіки роботи з рекламою додатків у Facebook: Lookalike Audiences та App Event Optimization.

У платформі Facebook існує кілька механік, що дозволяють ефективно просувати мобільні додатки. Дві з них - Lookalike Audiences (Подібні аудиторії) та App Event Optimization (Оптимізація подій додатків) - заслуговують на особливу увагу.

Lookalike Audiences - це механізм, що дозволяє створювати нові аудиторії, які подібні до вже існуючої аудиторії. Але в нашому випадку, воно не є приоритетним для нас, так як в даному випадку виключається використання аудиторії перенесеної на мобільний додаток.

App Event Optimization - ця механіка базується на оптимізації рекламних кампаній відповідно до дій, які користувачі виконують у додатку. За допомогою цього механізму ви можете налаштовувати рекламну стратегію таким чином, щоб ваша реклама максимально спрямовувалась на користувачів, які мають найбільшу ймовірність виконати певні дії у вашому додатку, наприклад, встановлення, покупки або реєстрацію.

Використовуючи ці механіки, ви можете досягти великого успіху у просуванні вашого мобільного додатку на платформі Facebook. Здійснюючи рекламу перед подібною аудиторією та оптимізуючи показ реклами з урахуванням конкретних подій додатку, ви збільшуєте шанси на залучення нових користувачів та підвищуєте ефективність вашої рекламної кампанії. Це особливо важливо для мобільних додатків, оскільки дозволяє спрямовувати рекламу на тих користувачів, які мають великий інтерес та ймовірність здійснити бажані дії, такі як встановлення додатку, здійснення покупок чи реєстрація.

Крім того, використання подібних аудиторій дозволяє ефективно розширювати ваше користувацьке бази, залучаючи нових користувачів, які мають великий потенціал для використання вашого додатку. Це може позитивно позначитися на конверсії та прибутковості вашого додатку.

Також, оптимізація подій додатку дозволяє вам більш точно налаштовувати ваші рекламні кампанії, враховуючи специфіку вашого додатку та потреби вашої цільової аудиторії. Це дозволяє досягти кращих результатів у просуванні вашого додатку та покращити його взаємодію з користувачами.

Враховуючи ці механіки та стратегії реклами додатків на Facebook, ви можете створити успішну рекламну кампанію, яка спрямована на залучення нових користувачів та підвищення популярності вашого додатку на ринку.

Зважаючи на вище наведену інформацію доходимо висновку, що для максимальної якості та дешевизни трафіку - необхідно на кожен новий мобільний додаток переносити аудиторію. І саме звідси випливає питання, яку аудиторію варто використовувати, якщо у межах інструментів збору інформації дані досить сильно подроблені, надані скоупом з багатьох джерел трафіку, а ми говоримо саме про Facebook.

Але перед тим як розглядати механіки переносу і шукати найліпшу - розробимо інструмент, який допоможе відслідковувати результативність наших дій;

Вище ми вже трохи розглядали аналітичний інструмент Keitaro, його й візьмемо за основу. Цю TDS будемо використовувати для збору великого обсягу даних, а у нашому випадку нас перш за все буде цікавити кількість унікальних кліків (унікальних встановлень мобільного додатку), кількість реєстрацій та кількість депозитів, безперечно з урахуванням мобільного додатку та країни надання рекламних послуг.

Структура самого інтерфейсу для користувачів самого Keitaro дуже нагадує бази даних, що в свою чергу є досить не зручним для аналітики, тож було прийнято рішення розробити програмний web-інтерфейс для побудови графіків та інших прикладних програмних задач. А так як продукт буде використовуватися в подальшому на комерційній основі, то одразу передбачимо систему авторизації та доступів до програмних ресурсів.

3.3. Використання Django для розробки

При розробці web-інтерфейсу для зручної візуальної аналітики даних та виконання інших прикладних завдань, використання Django є доцільним і має ряд переконливих аргументів. Ось кілька з них:

1. Ефективна розробка: Django є потужним інструментом для швидкої і ефективної розробки веб-додатків. Він надає готові компоненти та структуру проекту, що дозволяє значно зменшити час розробки. Django також пропонує багато вбудованих функціональностей, таких як аутентифікація, маршрутизація URL, керування формами та базами даних, що спрощують процес розробки.
2. Масштабованість: Django забезпечує потужні інструменти для масштабування веб-додатків. Його модульна архітектура дозволяє розбити додаток на компоненти і легко розширювати його функціональність. Django також підтримує розподілену роботу та горизонтальне масштабування, що дозволяє збільшувати обсяги даних та навантаження.
3. Безпека: Django має вбудовану систему безпеки, яка допомагає запобігти багатьом загрозам, таким як хакерські атаки, внедрення зловмисного коду та злам системи. Він має вбудовану систему аутентифікації та авторизації, захист

від перехоплення сесій, захист від міжсайтового скриптінгу та інші важливі механізми безпеки.

4. Багатогранність: Django підтримує різноманітні бази даних, такі як PostgreSQL, MySQL, SQLite та інші, що дозволяє вибрати найбільш підходящу базу даних для конкретного проекту.Крім того, Django надає розширені можливості роботи з базами даних, такі як ORM (об'єктно-реляційне відображення), яке спрощує взаємодію з базою даних через об'єктно-орієнтований підхід. Це дозволяє зручно працювати з даними та виконувати складні запити без прямого написання SQL-коду.

5. Гнучкість та розширюваність: Django забезпечує можливості гнучкої налаштування та розширення функціональності. За допомогою модульної архітектури, ви можете додавати власні компоненти, розширювати функціонал Django та налаштовувати його під свої потреби. Це дозволяє створювати унікальні та спеціалізовані веб-інтерфейси для аналітики даних та виконання прикладних завдань.

6. Велика спільнота розробників: Django має активну та велику спільноту розробників, яка забезпечує підтримку, розвиток і надання корисних документаційних матеріалів. Це означає, що ви можете знайти відповіді на свої запитання, розробляти на основі найкращих практик та скористатися різноманітними розширеннями та плагінами, розробленими спільнотою.

3.4. Процесс розробки

Розробка проекту почалася з безпосереднього налаштування віртуального середовища проекту, конфігурації внутрішніх налаштувань Django та проектування структури додатків у середині проекту, адже розробка з даним

framework має саме таку структуру: 1 проект - всередині велика кількість мікро-сервісів, які й називаються додатками.

3.4.1. Проектування розробки

Після створення проекту, організації віртуального оточення і встановлення Django, для початку розробки створимо сам Django проект (рис. 3.1) за допомогою команди: “django-admin startproject wwl-dashboard”.

А також, створено перший додаток всередині, за допомогою команди: “python manage.py startapp dasboard”. Адже для подальшої роботи дуже важливо створити базові шаблони та View, які будуть доповнюватися далі.

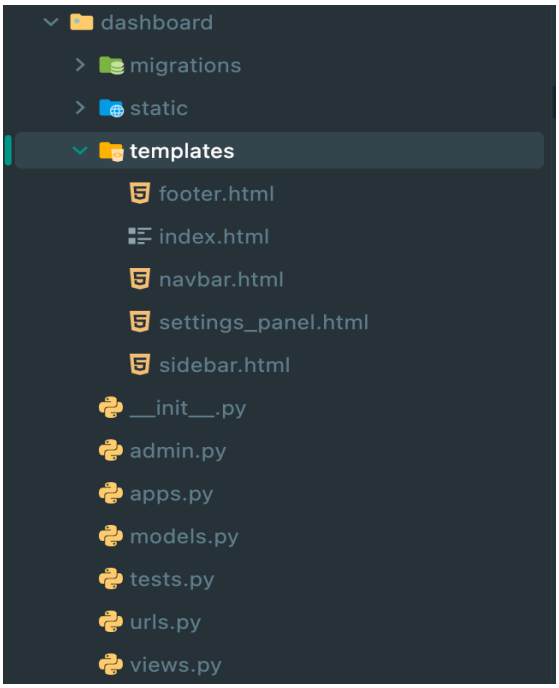


Рис. 3.1.

Також варто зазначити, що всі стилі будуть зберігатися в розділах static всередині кожного додатку, тож зазначимо це у файлі конфігурацій. (рис. 3.2)

```
STATIC_URL = '/static/'
MEDIA_ROOT = BASE_DIR / 'static/img'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

Рис. 3.2.

3.4.2. Створення каркасу сторінок (base template) та дочірні

Далі організуємо таку структуру проекту (рис. 3.3), яка в середині папки templates буде обов'язково мати файл index.html.

```
<!DOCTYPE html>
{% load static %}
<html lang="en">

<head...>

<body>
  <div class="container-scroller">
    {% include 'navbar.html' %}

    <div class="container-fluid page-body-wrapper">
      {% include 'settings_panel.html' %}
      {% include 'sidebar.html' %}
      <div class="main-panel">

        {% if messages %}
        <ul class="messages">
          {% for message in messages %}
          {% if message.tags %}
          <li class="{{ message.tags }}">{{message}}</li>
          {% endif %}
          {% endfor %}
        </ul>
        {% endif %}

        {% block content %}

        {% endblock %}
      </div>
```

Рис. 3.3.

Файл index.html створено безпосередньо для подальшої вбудови в нього внутрішніх сторінок, тобто він є каркасом самого дашборду. Додатково, окремо винесемо допоміжні html-файли.

3.4.3. Підключення статики / стилів / bootstrap:

Так як для подальшого спрощення оформлення UI був обраний bootstrap, то підключаємо його також у налаштуваннях. (рис. 3.4)

```
CRISPY_ALLOWED_TEMPLATE_PACKS = "bootstrap5"
CRISPY_TEMPLATE_PACK = "bootstrap5"
CSRF_COOKIE_SECURE = True
```

Рис. 3.4.

В середині самого ж базового шаблону index.html підключаємо вже і bootstrap і стилі. (рис. 3.5)

```
</title>
<link rel="stylesheet" href="{% static 'css/feather.css' %}">
<link rel="stylesheet" href="{% static 'css/materialdesignicons.min.css' %}">
<link rel="stylesheet" href="{% static 'css/themify-icons.css' %}">
<link rel="stylesheet" href="{% static 'css/typicons.css' %}">
<link rel="stylesheet" href="{% static 'css/simple-line-icons.css' %}">
<link rel="stylesheet" href="{% static 'css/vendor.bundle.base.css' %}">
<link rel="stylesheet" href="https://cdn.datatables.net/1.13.4/css/dataTables.bootstrap4.min.css">
<!-- нет шаблона -->
<link rel="stylesheet" href="{% static 'css/select.dataTables.min.css' %}">
<link rel="stylesheet" href="{% static 'css/style.css' %}">
<link rel="stylesheet" href="{% static 'img/favicon.png' %}">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@mdi/font@7.2.96/css/materialdesignicons.min.css">
<script src="{% static 'js/vendor.bundle.base.js' %}"></script>
```

Рис. 3.5.

Завершивши базові налаштування, створивши гіпер-розмітку для базової сторінки та стилів для неї - перейдемо безпосередньо до розробки першого підпроекту.

3.4.4. CR-Reviewer.

Для візуалізації даних я обрала framework plotly, адже він дає можливість побудувати повноцінні графіки із даними, використовуючи лише python, без підключення javascript/JQuery та іншого.

Перейдемо безпосередньо до створення інструменту візуальної аналітики, який буде передбачати можливість графічного відображення 4-х показників.

1. Кількість унікальних встановлень мобільного додатку
2. Показник відношення кількості реєстрацій до кількості унікальних встановлень мобільного додатку - CR install / lead
3. Показник відношення кількості депозитів до кількості реєстрацій - CR lead / sale
4. Показник відношення кількості депозитів до кількості унікальних встановлень мобільного додатку - CR install / lead

Кожну криву можна буде вмикати та вимикати, збільшувати або зменшувати масштаб, переміщуватись графіком, робити скріншоти потрібної частини графіку для подальшого порівняння аналізу.

На ці показники будить впливати далі наведені передбачені фільтри.

1. Можливість фільтрації трафіку з потрібної рекламної мережі
2. Вибір офферу - клієнту оренди мобільних додатків
3. Цільова країна для аналітики
4. bundle name - унікальний ідентифікатор додатка в Google Play Market та Keitaro
5. Date Range - інтервал дат, період за який хочемо побудувати графіки для подальшого аналізу.

За прок графіку було обрано 24 години, адже саме за цей час проходить повний цикл життя людини, включаючи час відпочинку, у який користувач грає у додатку. (рис 3.6)

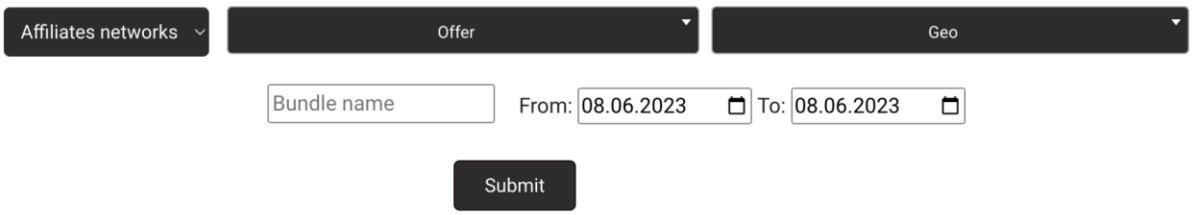


Рис. 3.6.

3.4.5. Розробка:

Першим кроком для візуалізації даних буде безпосередньо їх отримання, так як вся статистика зберігається на вищезазначеному сервісі Keitaro а також створення нового додатку в середині проекту. Додаток називається “crreview”.

Всередині додатку створили аналогічну структуру до базового додатку, єдине що інтерфейс будемо вбудовувати вже у базовий template index.html, тож у поточному тімплейті вкажемо наступне розбражене на рис 3.7.

```
{% extends 'index.html' %}
{% load static %}

{% block title %} Crreview {% endblock%}

{% block import %}
<link rel="stylesheet" type="text/css" href="{% static 'styles/crreview.css' %}" xmlns="http://www.w3.org/1999/html">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet"/>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
{% endblock %}

{%block content%}
```

Рис. 3.7.

Підключивши тімплейт до базового одразу створила форму, яка передбачає всі вищезазначені фільтри, а також виставляє стартові значення полів.

На рис 3.8. зображена частина форми, щоб показати саме принцип створення форми.

```
<form method="post" action="">
    {% csrf_token %}
    <div id="first-options">

        <div class="dropdown-crreview">
            <select class="dropbtn" name="social" required>
                <option disabled value="" selected hidden>Affiliates networks</option>
                <option value="FB">FB</option>
                <option value="UAC">UAC</option>
                <option value="FB+UAC">FB+UAC</option>
            </select>
        </div>


        <div class="offer-search">
            <select id="offer" name="offer" required>
                <option disabled value="" selected hidden>Offer</option>
                <option value="All offers">All offers</option>
                {% for offer in all_offers %}
                <option class="offer-option" value="{{ offer }}">{{ offer }}</option>
```

Рис. 3.8.

Після створення графічного відображення форми, знову ж таки пропрацювавши стилі, перейдемо до GET та POST запитів, а також отримання та виведення самих графіків.

Для виклику самої форми у файлі views.py створемо View-шку, яка буде відповідати за роботу безпосередньо з цією формою та графіком. Так як я обрала роботу саме через TemplateView, то оголосивши клас “CrReview”, одразу в середині оголосила 2 функції, які будуть відповідати за Get та Post запити.

Створимо 3 окремих сервісних файли.

1. kt_statistics.py - вміщує в собі досить великий клас, який використовується для побудови запитів до keitaro у всіх різновидах і комбінаціях фільтрів, а також для отримання переліку офферів при GET запиті для відображення форми. Також підчас розробки були передбачені умови не вибору деяких чи всіх фільтрів, ділення на 0 та всі можливі ситуації, які призвели би до сбоя роботи продукту.
2. kt_geo_data.py - вміщує в собі клас, основний метод якого повертає список із словарів, які вказують на співвідношення таймзони, коду геолокації та назви країни. Ці дані використовуються як при виведенні форми при GET запиті, так і для коректної побудови графіку.
3. interactive_graph.py - вміщує в собі досить простий клас, який безпосередньо описує відображення / виведення графіку, осей, даних, викликається тільки після коректного опрацювання POST запиту і отримання даних від Keitaro, якраз за допомогою сервісного файлу kt_statistics.py.

GET запит до цієї TemplateView:

При виконанні запиту, при першому завантаженні сторінки, підвантажується index.html, який в свою чергу доповнюється шаблоном із поточного додатку. Проте перед повним завантаженням сторінки, на бек-енді відбувається наступне (рис 3.9):

1. Відбувається запит до Keitaro, який викликається за допомогою методу get_all_offers, класу KTStatistic, response записується у змінну all_offers
2. all_geo_data - отримує весь список із країнами на тайм-зонами
3. Дані, отримані вище потрапляють у контекст, який безпосередньо передається на рендер на фронт-енд, завдяки чому форма стає заповненою.

Рис. 3.9.

Робота даного додатку побудована на сесіях для збереження статусу і даних поточного користувача, також це дає можливість використовувати після POST запиту замість рендеру - редірект.

Далі розглянемо POST запит до цієї TemplateView. (рис. 3.10.)

При POST запиті, як вже вказувала раніше, забираємо динамічно підвантаженні дані, які отримали на фронт при GET запиті з Keitaro. Фіксуючи обрані параметри, формуємо запит до Keitaro для отримання даних із кроком в 1 добу, як зазначалося раніше. Відповідно для цього, отримавши date range, розбиваємо його по днях, та із кроком в 24 години, отримуємо дані за кожну окрему дату. Дану механіку реалізовано у файлі kt_statistics.py, як зазначалося вище, в ньому передбачені всі можливі випадки і помилки як зі сторони

користувача так і з сторони машинного інтерфейсу Keitaro, даних з нулями тощо.

```
def post(self, request):
    kt_statistic = KTStatistic(
        social=request.POST.get('social'),
        offer=request.POST.get('offer'),
        bundle_name=request.POST.get('bundle'),
        geo=request.POST.get('geo'),
        date_from=request.POST.get('date_from'),
        date_to=request.POST.get('date_to')
    )
    unique_clicks_data, inst_lead_data, inst_sale_data, lead_sale = kt_statistic.get_statistic_by_metrix()

    request.session['date_from'] = request.POST.get('date_from')
    request.session['date_to'] = request.POST.get('date_to')
    request.session['unique_clicks_data'] = unique_clicks_data
    request.session['inst_lead_data'] = inst_lead_data
    request.session['inst_sale_data'] = inst_sale_data
    request.session['lead_sale'] = lead_sale

    return redirect('crreview')
```

Рис. 3.10.

Після виконання самого запиту і отримання даних, отриману інформацію зберігаємо в межах сесії, після чого перезавантажуємо сторінку користувача, і оскільки це вже буде запит, під час якого у сесії буде зберігатися інформація про date range - звертаємося вже у GET запиті до методів класу PlotlyInteractiveGraph, завдяки чому при фінальному завантаженні сторінки отримуємо вже не тільки форму а й графік із статистикою.

Під час отримання статистичних даних, на стороні Keitaro було знайдено декілька endpoint-ів для отримання статистичної інформації.

1. API - Програмний інтерфейс відпрацьовував некоректно, адже при отриманні response, кожного разу його зміст суттєво відрізнявся, хоча й при формуванні request щоразу використовувалися одні й ті ж статичні дані.
2. Запит сконвертований із C-URL на завантаження статистики за допомогою завантаження файлу із статистичними даними. (рис. 3.11.)

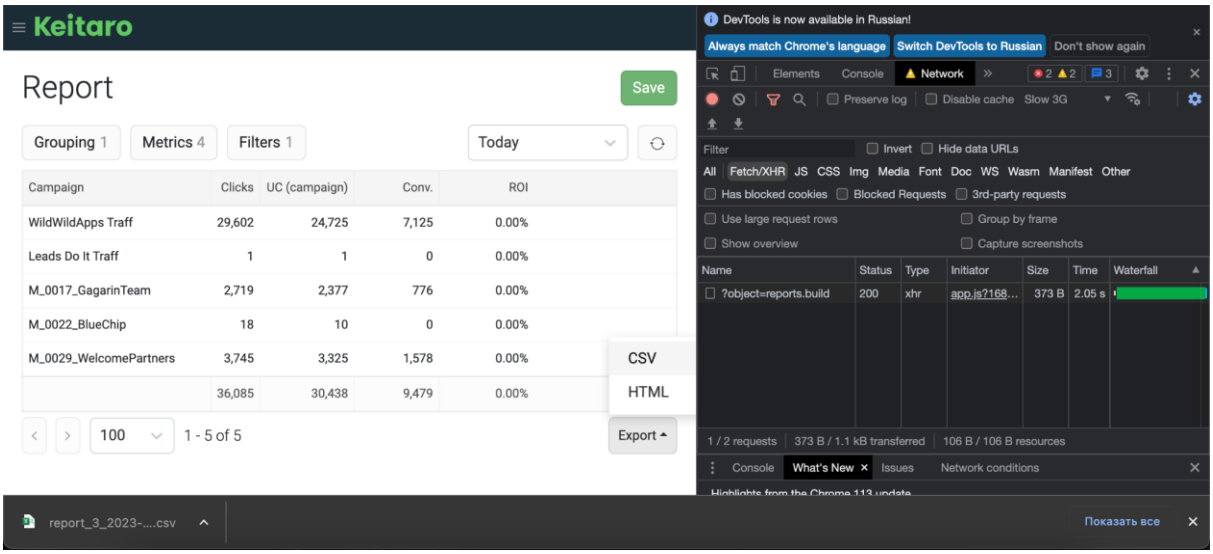


Рис. 3.11.

3. При використанні цього механізму результат щоразу відрізнявся, як і в пункті описанному вище.

4. Запит на завантаження даних у графічному інтерфейсі щоразу повертає у response всі коректні статистичні дані, тож я вирішила використовувати саме цей метод для отримання даних. Конвертувала C-URL, скопійований з консолі, за допомогою сервісу https://curlconverter.com/ у python, таким чином зформувався коректний запит до Keitaro.

3.5. Перші набутки цього інструменту:

Після розробки даного інструменту та аналізу понад 10- мобільних додатків вивела наступну закономірність, яка вказує на те, що при швидкому зростанні об’ємів трафіку, на понад 1,5-1,7 разів за добу, рівень конверту з реєстрації в депозит різко знижується, що свідчить про недосконаліть роботи штучного інтелекту Facebook, який мусить підбирати аудиторію, яка з

найбільшою ймовірністю внесе депозит, при заданні цілі рекламної оптимізації - “Покупка в середині додатку”.

3.6. Приклади

Тож можемо зробити висновок з рис 3.12, рис 3.13, що для зменшення фінансових маркетингових витрат і збільшення маржинальності бізнесу слід поступово нарощувати об’єми трафіку, а у випадку надання мобільних додатків в аренду великій кількості команд, слід збільшувати кількість мобільних додатків пропорційно до кількості команд. Саме такі дії стануть причиною збільшення валового прибутку і відповідно маржинальності бізнесу

Рис. 3.12.

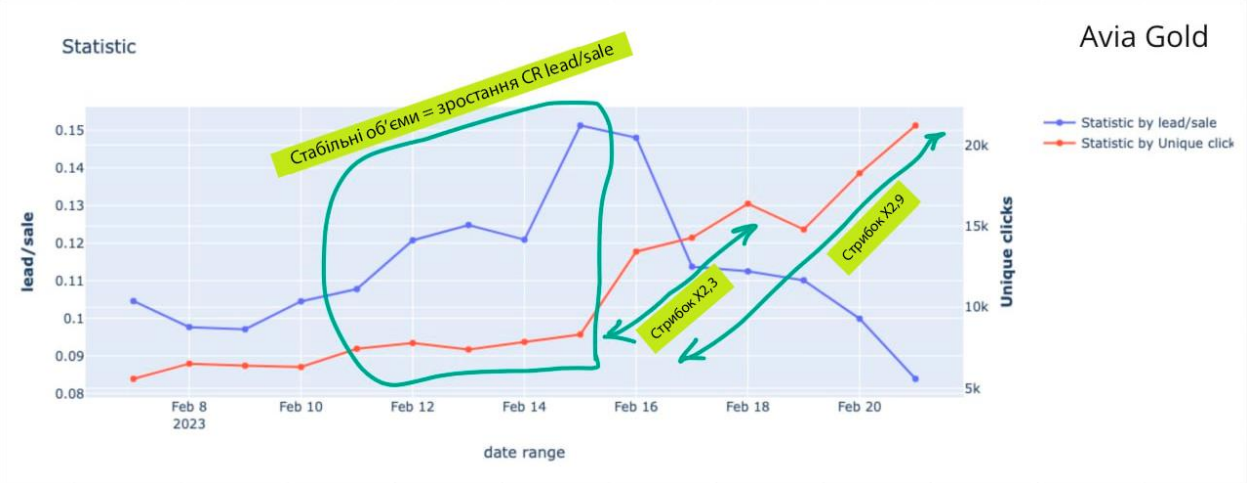


Рис. 3.13.

3.7. Збір та перелив аудиторії

Як було зазначено в пункті 4 другого розділу, для переливу аудиторії дуже важливо першочергово виділити саме ту когорту користувачів, яка буде:

1. Мапитися у мережі Facebook
2. При процессі мапінгу Facebook буде розуміти і давати максимально правильну когорту користувачів відносно саме наших очікуваннь.

Таким чином, першочерговим завданням для нас стає, саме виділення правильної когорти користувачів. рис. 3.14.

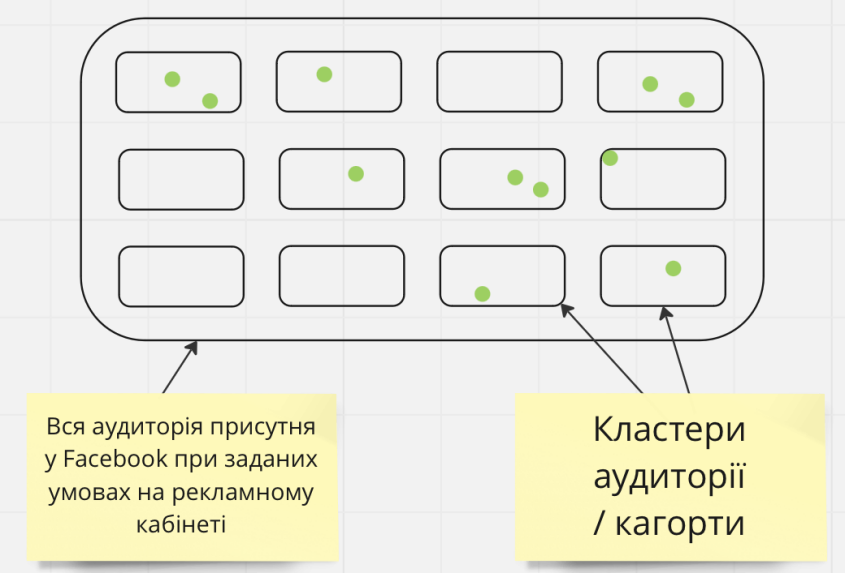


Рис. 3.14.

На рис 3.14. - механіка запуску рекламного кабінету, який просуває додаток без аудиторії

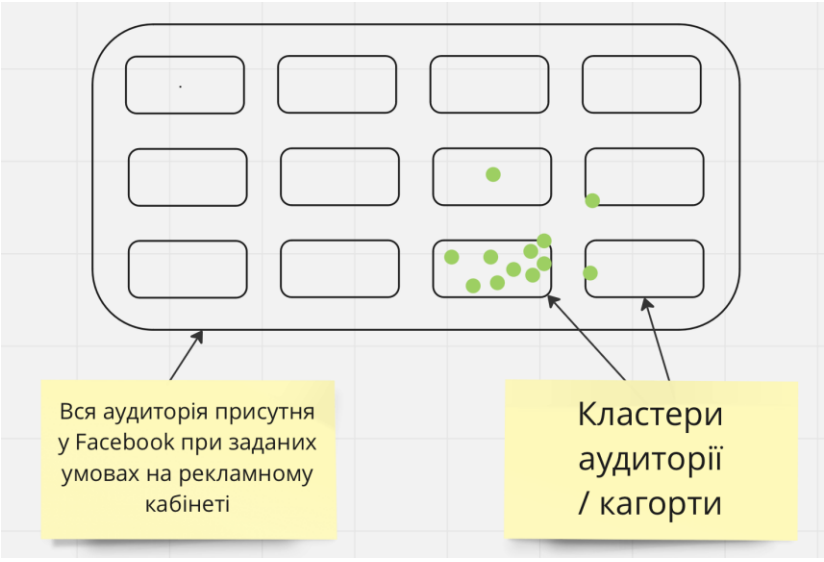


Рис. 3.15.

На рис 3.15. - механіка запуску рекламного кабінету, який просуває додаток з перелитою аудиторією.

Тож основною задачею стає підбір максимально правильної та чіткої аудиторії для переливу та подальшої оптимізації самої рекламної мережі.

Найбільший казус сфери Affiliate маркетингу!

Рекламодавці не віддають бази найліпших гравців та їх gadid, щоб ці дані не потрапили до конкурентів і ті не налаштували рекламу на їх гравців, тож навіть медіа баїнгові компанії та компанії по розробці додатків не мають інформації про тих гравців, яких ті ж казино й вимагають.

Тож в подальшому будемо використовувати одну з моделей нейромереж для створення потрібних сегментів аудиторії.

3.8. Механіка збору інформації про весь трафік, яка частково дублює дані Facebook-у:

Так як інформація про завершення реєстрації чи депозиту в додатку спочатку відбивається у рекламодавця та у нас в Keitaro, то всю інформацію про події одразу передаємо у Facebook;

Таким чином, якщо користувач із gadid_1 виконав реєстрацію, то ми отримуємо цю інформацію і записуємо в TDS а також передаємо у Facebook, відповідно якщо користувач із gadid_n виконав реєстрацію і депозит то відповідно до користувача із gadid_1 ми зберігаємо цю інформацію і дублюємо її в рекламну мережу. Приклад - рис. 3.16.

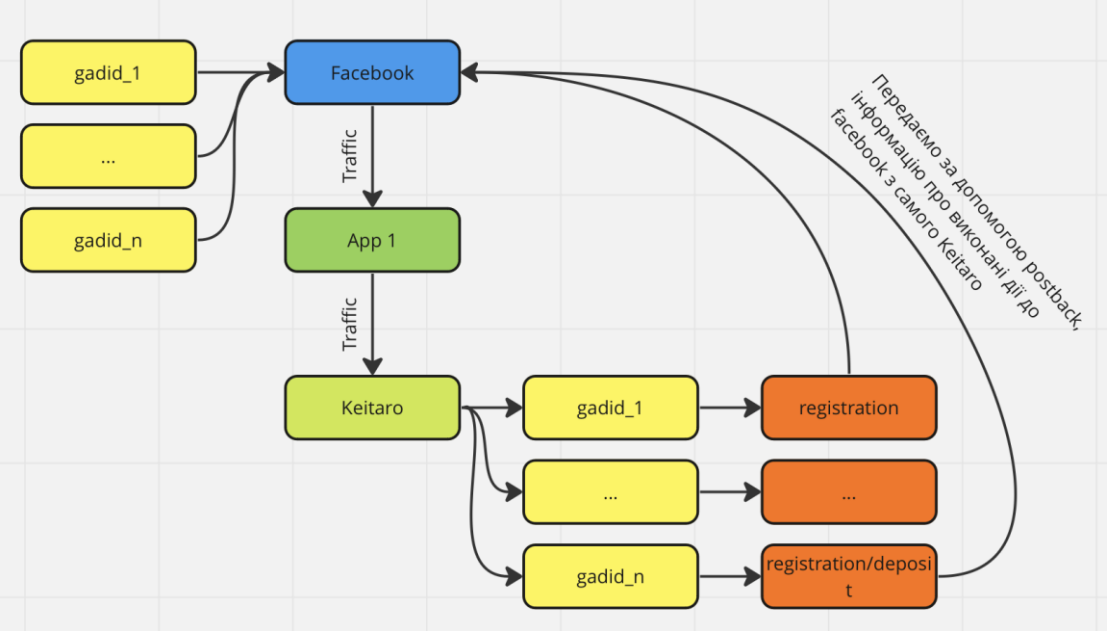


Рис. 3.16.

Отримуючи ці дані і маючи в базі даних статистичні на поведінкові фактори про кожного користувача, нейронна мережа facebook вже повноцінно класифікує і знаходить залежності серед поведінкових факторів та виконаних подій в додатку, таким чином виокремлює для рекламних кабінетів кластери чи суб кластери аудиторії.

У Facebook вказано, що чим більше трафіку пройде через рекламну компанію, тим більше вона оптимізується, це додатково підтверджує вище наведену інформацію.

З нашої ж сторони, у Keitaro зберігається тільки інформація про завершені події, країну та мову користувача, версію андроїд-пристрою. Жодної інформації про поведінку чи іншу класифікацію. А розуміючи, що на додаток постійно запускають нові рекламні кабінети, кожний з яких щоразу надає нову аудиторію, робимо висновок, що всю аудиторію переносити немає сенсу.

3.9. Чому не переносити тільки депозити?

Що ж стосується перенесення тільки депозитів, введемо новий термін, який щоразу вимагає рекламодавець, цей термін - “якість трафіку”.
Дана вимога говорить про наступне: якщо медіабаєр (людина яка рекламує казино), приводить гравця у казино по CPA-моделі, то за цього гравця казино виплачує фіксовану суму грошей, наприклад: 2000 грн за гравця з України, якщо той вніс мінімальний депозит 200 грн. Математика у цьому випадку ніби грає проти казино на -1800 грн, проте економіка казино базується якраз на повторних депозитах і високій активності гравця, які якраз і називаються якістю трафіку. Цей показник також має математичне відображення у двох термінах: NGR та GGR.

З статистичних даних, отриманих від рекламодавців: FavBet, Cosmo Lot, Slot City, First Casino, Ice Casino та інших, понад 80% користувачів з трафіку, який приводять медіа баєри є неякісним, тож для того, щоб медіа баєри давали якісний “App Event Optimization” трафік, на додатку має бути правильна аудиторія, яка формується далеко не з усіх користувачів, які зробили депозит.

3.10. Мікс джерел чи лише Facebook?

З вищезазначеного випливає, якщо трафік з будь-якої системи агрегується в одній і тій самій системі, яка називається Keitaro - то можливо проводити безліч експериментів, спрямованих на перетин різних аудиторій із різних джерел трафіку, чи будувати воронки ремаркетингу, проте це вже інша тема. На мою думку, це твердження є дійсно правдивим, так як класифікація однієї

рекламної мережі стосовно користувача може налічувати 20 показників, а інша вже 40 зовсім інших показників, хоча користувачів і об’єднує один ідентифікатор, проте поведінкові та статистичні характеристики завжди будуть різними. Тож далі не будемо міксувати аудиторію з різних джерел трафіку і будемо працювати лише з Facebook.

3.11. Збір даних для аналітики

Перед самим аналізом і сегментацією треба отримати датафрейм для аналізу, який в собі буде поєднувати групи з ідентифікаторів користувача та виконаних цільових подій, опис поведінкових факторів тих самих користувачів.

Дані будемо отримувати з двох сервісів:

1. Keitaro - система розподілення та контролю трафіку, з неї будемо отримувати співвідношення подій та ідентифікації користувачів. (рис. 3.17)

Sub ID 8	Клики	Лиды	Продажи	Доход
5ad554ab-0ddb-47a3-b61a-dfa1b85b0540	1	0	0	0,00 $
23835819-3442-4e5d-94b2-6417194e3dad	1	0	1	5,61 $
012c7f6a-bc98-40e6-966d-630b6e280beb	1	1	0	0,00 $
cd6305d7-c4b9-45c1-8182-8da8ce77dc4d	1	1	0	0,00 $
a91ade45-e9bc-449b-a9b0-bdc29fbb59fc	1	0	0	0,00 $
37141950-5fc1-4c5b-bbe0-0a08e090b6c4	1	1	0	0,00 $
	1 281 625	283 974	53 048	818 724,98 $

Рис. 3.17.

2. Onesignal - SDK для маршрутизації пуш-сповіщень та контролю поведінкових чинників користувача (рис. 3.18.):

a. Дата та час першої сесії
b. Кількість сесій користувача
c. Загальний час, витрачений у додатку
d. Дата та час останньої сесії

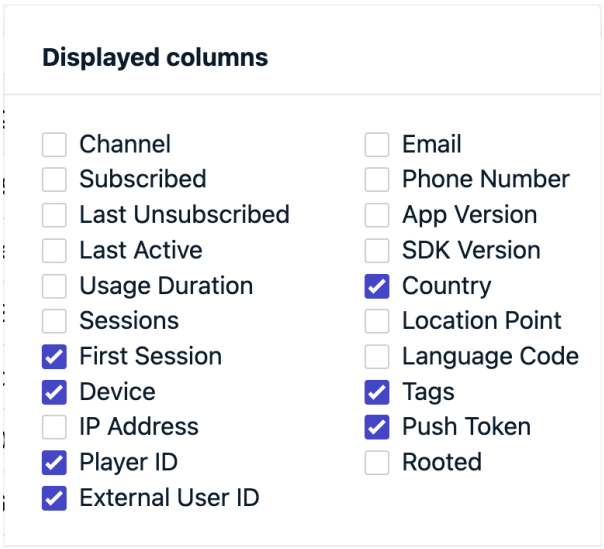


Рис. 3.18.

Далі поєднання інформації про кожного користувача відбувається завдяки поєднанню інформації про нього з 2х сервісів в 1 датафрейм, використовуючи унікальний ідентифікатор (Рис 3.19.), який називається gadid, у межах OneSignal має назву external_user_id, а в середині Keitaro, з яким ми взаємодіємо, зазначається як sub_id_8.

Gadid - це унікальний ідентифікатор кожного мобільного пристрою, який працює на базі ОС Android, цей ідентифікатор завжди можна змінити на новий у налаштуваннях, тоді старий id стане безкорисним.

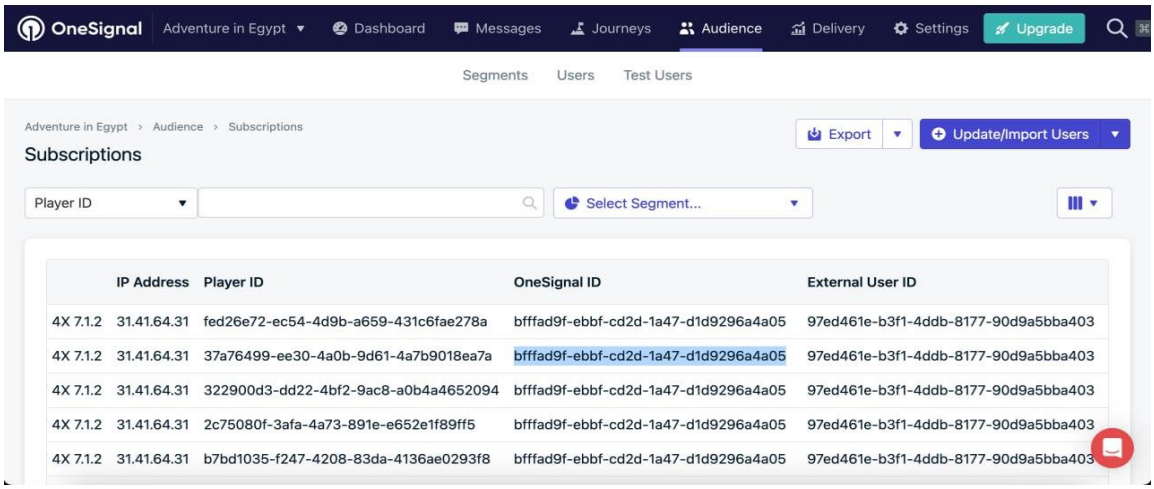


Рис 3.19.

Так як даний сервіс не потребує графічного інтерфейсу а на сервері запускається за допомогою команди nohup python3 main.py &, то даний сервіс будем розробляти як зовсім окремий проект.

3.12. Створення проекту

Загально проект розбиваємо на 2 частини.

1. collectors - частина проекту, яка буде відповідати за збір та поєднання даних.
2. predictors - побудова безпосереднього передбачення для кожного користувача про ймовірність повторного депозиту всередині додатку.

Також окремо винесено файл main.py, який будемо використовувати для запуску всього проекту. Команду запуску вказано вище. (рис. 3.20.)

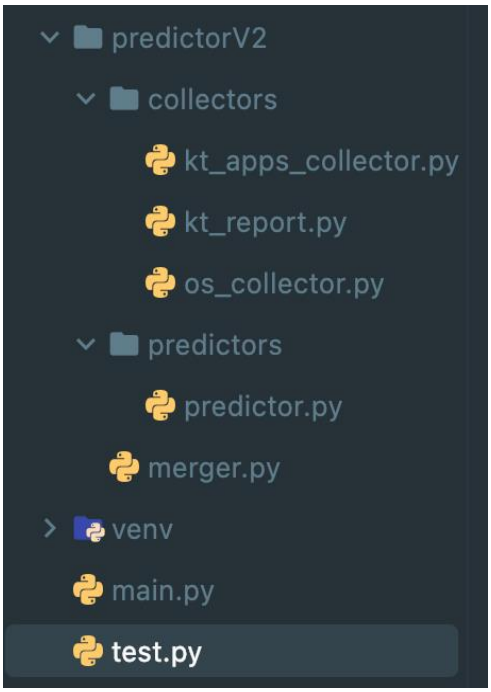


Рис. 3.20. Структура самого проекту

Отримання даних з Keitaro.

Процес отримання даних розібємо на 2 файли.

1. kt_apps_collector.py - сервісний файл, який будемо використовувати у тілі запиту, в якому в обов'язковому порядку будемо вказувати ідентифікатор додатку у Google Play, адже саме цей параметр буде важливим для аналітики, щоб не перемішувати аудиторію з різних додатків. Також у якості фільтру зазначимо, що нас цікавлять юзери саме з Facebook; (рис.3.21)

```
{
    'name': 'source',
    'operator': 'CONTAINS',
    'expression': 'acebook',
},
```

Рис. 3.21.

Далі в групінгу та метриках вказуємо потрібні параметри, зазначені вище.

2. kt_report.py - файл, який формує запити за пів року до сервісу Keitaro з кроком у 3 дні, так як єдиний можливий спосіб отримання даних - це авторизація через api і звернення до едпоінтів які використовуються на фронті разом з авторизацією через кукі. Враховуючи навантаженість сервісу, при запиті за весь період сервіс віддає 500 помилку, тим самим унеможливлює отримання даних одним запитом за весь період. Також, враховуючи, що для аналізу використовувалося понад 100 додатків і понад 5 мільйонів користувачів, я передбачила помилку “SSL Error” при отриманні даних за певні періоди дат, задля того, щоб скрипт пропрацював дійсно весь об’єм даних.

На рис. 3.22. зображенний даний фрагмент коду.

Після отримання всіх даних - повертаємо зібраний повний респонс для подальшого використання.

```
temp_date_from = self.date_from.replace(minute=0, hour=0, second=0, microsecond=0)
temp_date_to = self.date_to.replace(minute=0, hour=0, second=0, microsecond=0)
# timedelta_days = 3
while temp_date_from < temp_date_to:
    date_from = temp_date_from.strftime('%Y-%m-%d %H:%M')
    date_to = (temp_date_from + timedelta(days=timedelta_days)).strftime('%Y-%m-%d %H:%M')
    logger.debug(f'get rows from {date_from} to {date_to}')
    try:
        json_data = self._get_json_data(date_from, date_to)
    except requests.exceptions.SSLError:
        time.sleep(10)
        logger.warning('SSL error, sleep 10 seconds and try again')
        try:
            json_data = self._get_json_data(date_from, date_to)
        except requests.exceptions.SSLError:
            logger.error('SSL error again, skip this date')
            continue
    rows.extend(json_data)
    # new_generator = (_ for _ in json_data)
    # generator = itertools.chain(generator, new_generator)
    temp_date_from += timedelta(days=timedelta_days)
# return generator
return rows
```

Рис. 3.22.

Отримання даних з OneSignal

Робота з OneSignal стала дещо легшою, адже у випадку з ним, API дійсно працює коректно, за винятком швидкості генерації результатів у високонавантажених додатках.

Тож для отримання даних, нам знадобляться API ключі для кожного додатку, їх ідентифікатори в системі та ідентифікатори в Google Play Market для подальшої обробки. Ці дані зберігаються в пуш-системі компанії, тож при зверненні на едпоінт: http://wildwildpush.com/apps/all/?key=secret_key_value - отримуємо json response у вигляді списку з словників, вкладених у словник. (рис. 3.23.)


{"data": [{"name": "Adventure in Egypt", "onesignal_id": "58055375-7ff1-43b1-aa74-8f9419dc9403", "api_key": "ZTUwZGM0MWQtOTE4MC00YTlkLTg3NTktNmE4N2VmYTNiZDUz", "bundle_name": "com.naturalmotion.customstreetrace"}, {"name": "Adventure in Olympus", "onesignal_id": "615329bc-884d-4415-8179-31fc86f4da2a", "api_key": "NTIxY2RmY2UtZmFjYS00NGFkLWI1YjktOTlmYzdlMmJkYzVm", "bundle_name": "com.techvision.restau"}, {"name": "Adventures and Gifts", "onesignal_id": "eb705c76-2453-4664-b069-f118232422cd", "api_key": "ZmNmNGNhNmItOTU4Yi00ZDcxLTg4NDItNzc2NzFmZTMwYjdi", "bundle_name": "com.maroiegrwlk.unp"}, {"name": "Afrodita Power 2", "onesignal_id": "5fcc59bc-eb5d-4195-a050-9104bd31fc53", "api_key": "ODVkZDM2YzgtNTk1Mi00NmJhLTk0MTUtNDRmYmQxMmE2MWU1", "bundle_name": "com.pikpo"}, {"name": "Annubis Curse", "onesignal_id": "8680dc35-90c4-45c3-ab6f-0fb463714a50", "api_key": "ZmJhY2ViNjEtYTZiZS00MmVkLWIxZDQtODRjNWQxM2VjMjY0", "bundle_name": "com.weplaydots.twodotsandroi"}, {"name": "Anubis Gold", "onesignal_id": "cb1415ff-2999-48a5-aa43-bb1aad74351a", "api_key": "OWM2M2EzMmMtNWU3ZS00ZTVjLThkZTktYTM1MmFiNTY5Zjk4", "bundle_name": "com.outfit7.mytalkingangel"}, {"name": "Avia Gifts", "onesignal_id": "d3cfff14-4249-4603-99cc-5b59f591976f", "api_key": "ZDBlOTM4MTYtODZkNC00NWE1LWFlYjQtMGI3NjczYjZiMWY4", "bundle_name": "com.playkot.adve"}, {"name": "Avia gold", "onesignal_id": "ecaf2867-8641-4c86-aa2e-ecf3f4a8f4a8", "api_key": "MmM1ODdiMWYtZTg2NC00MTA0LWI2OTEtNjE0M2U2NGRmNzAz", "bundle_name": "com.hyperbeard.adorableho"}, {"name": "Aztec Wheel", "onesignal_id": "e1aa9155-d24c-4030-92c0-1fb43c265b1d", "api_key": "ZDg2ZmZlY2UtNzA2Yi00NDc1LTk5ODQtNDM4NTJiNGQ1Mjk5", "bundle_name": "com.boomblitgames.ArcheryClubTournamen"}, {"name": "Candy Land", "onesignal_id": "1312aeb2-155e-4cd1-b9f6-9b27851ad708", "api_key": "NTE3ZDhhZWUtOTY5OC00YzRjLTg2NjMtZjU0ZmM5ZmVkOTE2", "bundle_name": "com.and.riseofth"}, {"name": "Cave of Gold", "onesignal_id": "178c7bbb-bbf5-437c-a104-a381b07ae775", "api_key": "MjdmMDFmZmEtZmFiMC00MmIxLTg4YTktMzg3ZTE3ZDJhYWM0", "bundle_name": "com.craneballs.overkill"}, {"name": "Eastern Treasures", "onesignal_id": "30ba471d-2d06-4eab-8bad-4db2566d98df", "api_key": "MzE4MzJiYmItZGYzYi00Zjc5LWFjNGQtZDcyMGZiNGZkMjRm", "bundle_name": "com.mgs.snipe"}, {"name": "Gold of Zeus", "onesignal_id": "ef66738f-6b0b-4cc3-863d-9f6a5874126b", "api_key": "NmY1ZTZiY2UtODM5MS00ZTM5LWI2ZTctYWJmNmNiYWQwMGE1", "bundle_name": "com.bigbreakgames.w"}, {"name": "Fire Sand", "onesignal_id": "1705d592-b5ac-4299-8b92-d7699fle2ad6", "api_key": "ZWQyNWQ2NmEtMGUwNC00NmEyLTk4ODItYzhjYTFhNjk0ZWRl", "bundle_name": "com.whaleapp.blitz.rise.heroes.idle.rpg.battl"}, {"name": "Flash Wheel", "onesignal_id": "267b4cc1-e1cc-49b3-9d31-dc7411c72185", "api_key": "Y2Y2MWU2MWUtZDNiNS00ZDMlLThhMzYtNTc4ZjhlYjMyYzM4", "bundle_name": "com.hunt.royal"}, {"name": "Football World Teams", "onesignal_id": "d12e4504-71a0-4ab6-af5f-6093eb160af2", "api_key": "NTFlMWIzZGQtZTllZi00YjgxLWJiYWMtOGZmMTI3ZTIyNzU1", "bundle_name": "com.lyoomatch.jewelscrush.princessadventur"}, {"name": "Forest Secrets", "onesignal_id": "9dc818f2-1875-41e5-9f30-4a891e338681", "api_key": "MzBlZWE5NDMtZmZkMS00YjdiLWJhYzgtMDYxMzdkM2QzNTZk", "bundle_name": "com.atari.mobile.rctempi"}, {"name": "Fruit Boom", "onesignal_id": "c8c9782e-7465-422c-b694-2087adfc8375", "api_key": "ZTNmZDc3NzMtMzM4NS00YjVhLTllMzYtNmFiMTBhZWNkMzQz", "bundle_name": "io.voodoo.pa"}, {"name": "Fruit Gold", "onesignal_id": "08b7e819-08bf-4af2-91fc-708ed0e1c7f2", "api_key": "MTg1YjQxMjUtYzg5MC00OTIxLWIzYWMtZjEyMWY3ZDUwODg3", "bundle_name": "com.kabam.bullsey"}, {"name": "Fruityfall", "onesignal_id": "cfc1a6e3-0a2b-4759-aa46-9c836ce8636d", "api_key": "M2UwMGIyNDktM2ZkMS00YjM2LThhZjMtMzFjNmQ0ZjQyZGU5", "bundle_name": "dk.tactile.lilysga"}, {"name": "Gifts from Muerto", "onesignal_id": "06d76ba5-48cd-4503-8d03-5dc96d547a88", "api_key": "MmEyM2JhNDUtMGUxNy00MTdlLTgzODMtOTBmOGQzZjRjNWU3", "bundle_name": "com.hepagames.wheelevolutio"}, {"name": "Gifts From The Joker", "onesignal_id": "17belef4-17c3-4e6e-a09b-81b7969ad8d3", "api_key": "NDhlMDE2NTktMzk0Zi00Y2MwLThlYzYtNTExNjQlMzJhOTYw", "bundle_name": "org.duckdodo.baiko"}, {"name": "Gifts of Anubis", "onesignal_id": "2c4b4ae6-3524-470d-abdd-144dc0f975cd", "api_key": "YWUxZGZiZTktYjUwNy00MzY3LThmZjYtYTRhNDlmZDNmZDgx", "bundle_name": "com.kitka"}, {"name": "Gifts Roulette", "onesignal_id": "96d65e62-f87f-47f4-a201-1222fc176a50", "api_key": "OTQ2MWM5YjktZTZhZC00N2VjLWEyNmMtMWY1ZGE3NDMxNjA4", "bundle_name": "com.gameicreate.christmasfeve"}, {"name": "God of Luck", "onesignal_id": "8f5ff140-2523-4dee-a6b3-017b9f20bb74", "api_key": "MTkzYTExMTEtZDk4My00NzYyLWE0MDktOGRmY2E4OGUxMjFl", "bundle_name": "com.lokumgames.snookerelite3"}, {"name": "Gold Adventures", "onesignal_id": "7b60df74-8bc7-41e1-993b-09a53e8a490f", "api_key": "NWNiZTVjZmYtNjMzYi00NDQxLWFhOGQtOTU4OGI5MjBmMTk0", "bundle_name": "com.miniclip.agar.i"}, {"name": "Golden cave of fruits PM", "onesignal_id": "a27edf04-f900-4254-8cf6-1bb949af1ac8", "api_key": "OGZjZjQ2ZGMtZTg3Mi00NzAyLTk5NzUtNzBkOTg2ZTE5YTc1", "bundle_name": "com.find.out.hidden.object"}, {"name": "Golden Chance", "onesignal_id": "b0b0869b-60d4-4fbf-bc59-3f55933bdea5", "api_key": "NGFlZmE1ZTktOWFiZC00OWQwLWExMDItNzhiYjFjZWYwZTgy", "bundle_name": "cc.foresta"}, {"name": "Golden Clover", "onesignal_id": "21690a6d-f2c7-4e1e-8aaa-f2a3aca224df", "api_key": "OGM2YTIyNjYtODY5NS00YjIyLTg3MjgtNGNlYmQzZDc4ZTMy", "bundle_name": "com.efun.sed3.e"}, {"name": "Golden Cobra", "onesignal_id": "02308b78-c233-4371-8efa-527984b75858", "api_key":

Рис. 3.23.

Далі для кожного додатку формуємо запит на отримання даних від OneSignal у форматі посилання на завантаження gzip із вкладеним CSV файлом з даними. Після отримання файл треба завантажити, тут зустрічаємося

з вищезазначеною проблемою, про довгу генерацію файлу, тож передбачимо обробку даної проблеми наступним чином. (рис.3.24.)

```
def _get_bytes_csv_file(self, url):
    print(url)
    while True:
        r = requests.get(url)
        if '<Error>' in r.text:
            time.sleep(5)
            print('Trying')
        else:
            print('True')
            break
    return r.content
```

Рис. 3.24.

Після чого, розпаковуємо файл і передаємо на подальшу обробку.

Поєднання даних

Кожний з об'єктів з даними форматуємо у pandas DataFrame, після чого поєднуємо дані файлів, використовуючи однакові ідентифікатори та вбудований функціонал merge у pandas. В результаті отримуємо 1 великий DataFrame для кожного додатку, де кожен рядок описує як поведінкові фактори кожного користувача, так і виконані та невиконані дії.

Крім того, перед поєднанням даних, для кожного користувача, якщо він зробив депозит (sale), то у полі lead також проставляємо значення True, адже за воронкою продажу кожен юзер спочатку реєструється а вже після робить депозит, проте Keitaro влаштований таким чином, що при події sale, він переводить значення lead у позицію False. (рис. 3.25)

```
kt_data.loc[kt_data['is_sale'] == True, 'is_lead'] = True
```

Рис. 3.25

3.13. Аналіз, RandomForestRegressor

Для виділення потрібних нам когорти користувачів використаємо RandomForestRegressor.

RandomForestRegressor - це алгоритм машинного навчання, який використовується для вирішення задач регресії, тобто передбачення неперервного виходу. Він базується на методі випадкових лісів (Random Forest), який є ансамблевим методом.

Random Forest - це ансамбль рішень дерев рішень, де кожне дерево вирішує задачу класифікації або регресії, і результати об'єднуються шляхом голосування або усереднення для отримання кінцевого прогнозу. Дерева будуються шляхом вибору випадкового піднабору ознак і набору даних і забезпечують велику різноманітність у моделі.

RandomForestRegressor використовує Random Forest для вирішення задачі регресії. Під час навчання він будує багато дерев, використовуючи випадковий підбір ознак і даних, і потім усереднює (для регресії) прогнози кожного дерева для отримання кінцевого прогнозу.

Основні переваги використання RandomForestRegressor

1. Добре працює з великими наборами даних і високоімовірнісними ознаками.

2. Здатний автоматично управляти відсутніми даними і підтримує обробку відсутніх значень.

3. Забезпечує оцінку важливості ознак, що дозволяє виявити найбільш важливі ознаки у наборі даних.

4. Має хорошу стійкість до перенавчання і здатний підтримувати добру узагальнюючу здатність.

Випадки, коли варто використовувати RandomForestRegressor

1. Коли є великий набір даних, що містить числові властивості і/або категоріальні ознаки.

2. Коли робиться прогноз для неперервної змінної, наприклад, передбачити ціну нерухомості, дохід людини або вартість товару.

3. Коли включається важливість ознак у аналізі. RandomForestRegressor надає оцінку важливості ознак, що допомагає зрозуміти, які фактори найбільше впливають на вихідну змінну.

4. Коли в даних є викидами (outliers) або шум. RandomForestRegressor є менш чутливим до таких випадків і може дати більш стабільні прогнози.

Ці фактори якраз описують ситуацію, тож застосуємо саме цей метод. Його реалізація описана в файлі predictor.py. На виході отримуємо DataFrame, який описує ймовірність першого на повторного депозиту для кожного гравця у фреймі.

Так як в базі Keitaro зберігається інформація про сотні додатків і кожен з них є унікальним, аналізуємо кожен з них в окремому циклі, також врахуємо наступне: кожна країна на кожному додатку аналізується окремо, і у випадку якщо даних для аналізу не вистачає - таке співпадіння пропускаємо.

У якості поведінкових факторів для аналізу будемо використовувати наступні дані.

1. Кількість часу яка пройшла з моменту встановлення мобільного додатку
2. Кількість часу проведеного в середині додатку
3. Середня тривалість сесії гравця

Саме ці показники разом із деталізацією про реєстрації та депозити дають можливість правильного визначення ймовірності внесення повторного чи першого депозиту відносно поведінкових факторів.

3.14. Програмна розробка моделі

Так як аудиторії у кожній країні зовсім різні, окрім того, що аналізувати користувачів будемо по кожному додатку окремо, кожну країну будемо аналізувати також окремо. В свою чергу розуміємо, що маємо встановити мінімальне обмеження по кожній країні, тож задамо мінімальну кількість користувачів для аналізу - 500. Окрім цього, близько 10% користувачів знаходяться у базі із ідентифікатором 00000000-0000-0000-0000-000000000000, тож користувачів з таким ідентифікатором теж не будемо використовувати при подальшому аналізу.

Після того, побудувавши потрібні показники у коді, взявши за основу дані з OneSignal, аналізуємо їх задопомогою моделі RandomForestRegressor. (рис. 3.26)

```
    def _preprocess_data(self):

        self.country_df['last_active'] = pd.to_datetime(self.country_df['last_active'])
        self.country_df['created_at'] = pd.to_datetime(self.country_df['created_at'])
        self.country_df['time_since_install'] = (self.country_df['last_active'] - self.country_df['created_at']).dt.total_seconc
        self.country_df['average_session_duration'] = (self.country_df['playtime'] / self.country_df['session_count'])
        self.country_df = self.country_df[self.country_df['time_since_install'] >= 0]
        self.country_df = self.country_df[self.country_df['is_lead']]
        self.country_df = self.country_df[self.country_df['sub_id_12'] == self.target_sub_id_12]

    def _train_model(self):
        X = self.country_df[['time_since_install', 'playtime', 'average_session_duration']]
        y = self.country_df['is_sale']
        self.regressor = RandomForestRegressor(n_estimators=100, random_state=420)
        self.regressor.fit(X, y)

    def _predict(self):
        self.country_df['predicted_value'] = self.regressor.predict(self.country_df[['time_since_install', 'playtime', 'averag
```

Рис. 3.26

Результат роботи записуємо у csv файл predicted.csv.

Також через те, що робота цього скрипту зайняла понад 100 годин, у код додала логування за допомогою loguru, що дозволило у реальному часі відслідковувати поточний стан збору та аналізу аудиторії.

Результат роботи показані на рис. 3.27.

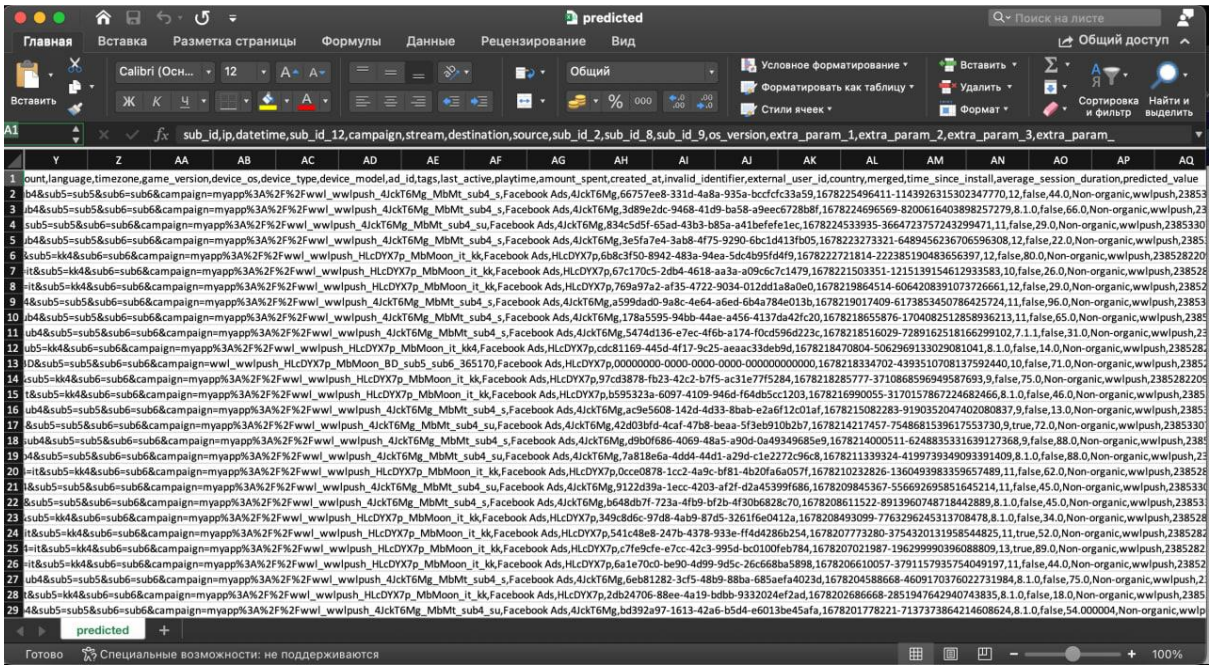


Рис. 3.27. Фрагмент фінального файлу після аналізу

Перевірка:

Для перевірки результативності роботи даного алгоритму звернулася до дуже близьких друзів з компанії BlueChip, які протягом декількох місяців збирали деталізацію по юзерах з Індії, проведених через додатки, які досліджувалися. Після чого була сформована вигрузка gadid за допомогою нейромережі. (рис 3.27.) Результат роботи був порівняний із даними наданими BlueChip. Співпадіння склало близько 80%+ при наданні random_state значення 420.

3.15. Скрипт переносу аудиторії

Для повноцінного функціонування даної моделі переносу аудиторії також передбачимо скрипт, як безпосередньо буде зчитувати csv файл із колонкою, яка буде містити потрібні gadid у facebook для необхідного додатку. Даний мікросервіс вбудуємо безпосередньо у дашборд, адже перелив аудиторії буде відбуватися на кожен новий додаток, що зумовлює потребу у UI-інтерфейсі.

Так як розробка буде відбуватися у межах Django, то діємо за стандартною структурою проекту, створюємо форми, html-розмітку, якою розширюємо index.html, а також утіліти для самого переливу аудиторії.

Даний функціонал розіб’ємо на 2 частини.

1. Спеціальна сторінка із формою, яка через метод POST буде отримувати csv файл і його кодову назву для панелі керування. Сама форма має наступний вигляд (рис. 3.28,3.29.)

```python
# Добавление аудитории форма
class AddAudience(forms.Form):
    dictionary = {
        'api_keys': {
            'class': "form-control",
            'id': "title_id",
            'placeholder': "aqw-erf-dsx\nqwe-rty-uio..."
        },

    }
    aud_name = forms.CharField(label="Назва аудиторії", max_length=50,
                               widget=forms.TextInput(attrs=dictionary['api_keys']))
    file = forms.FileField()
```

Рис. 3.28

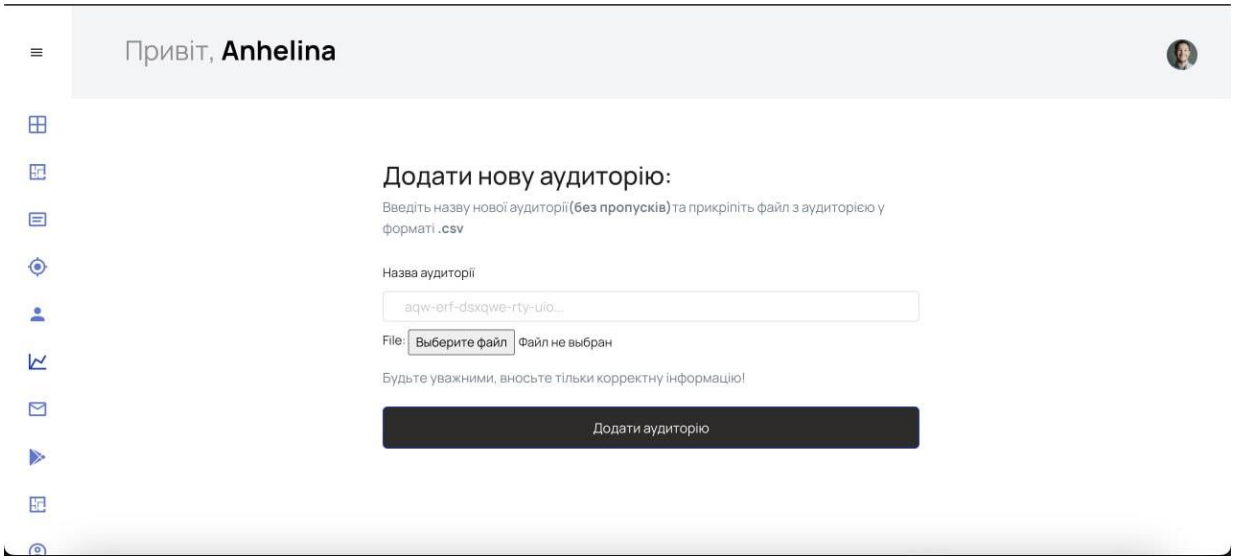


Рис. 3.29

2. Спеціальна форма, яка буде приймати в себе спеціальний id мобільного додатку, секретний api-токен, назву події (lead/sale) яку будемо переносити та дропдаун, у якому можна вибрати потрібну аудиторію за її назвою. (рис. 3.30)

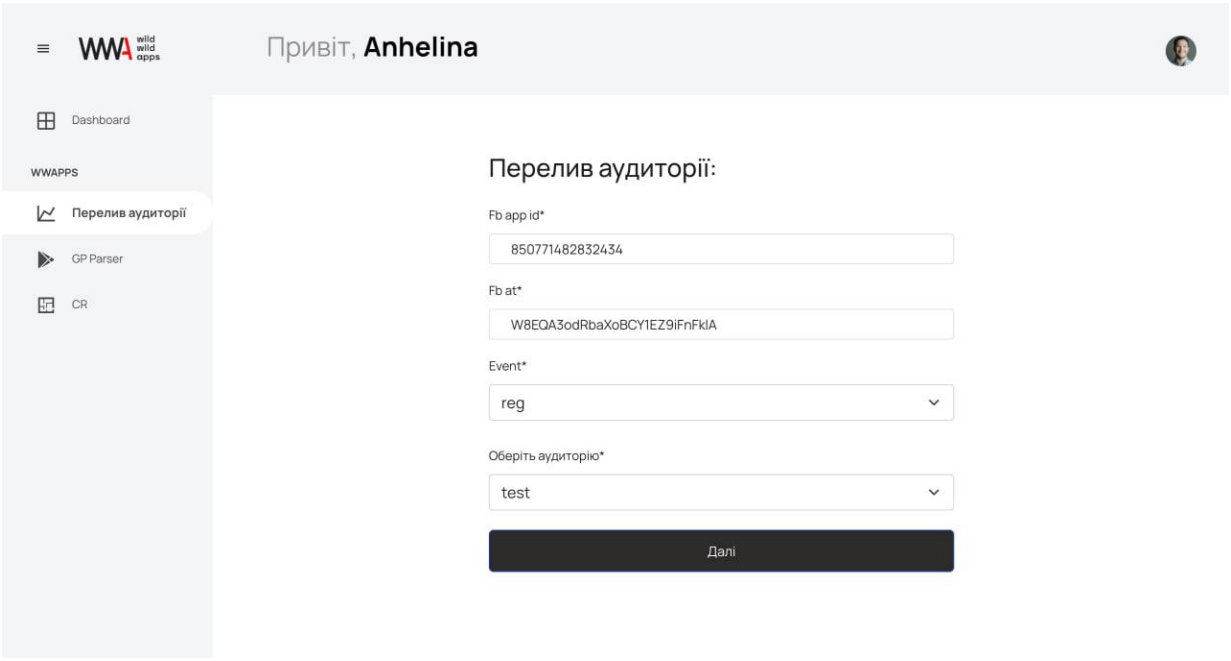


Рис. 3.30

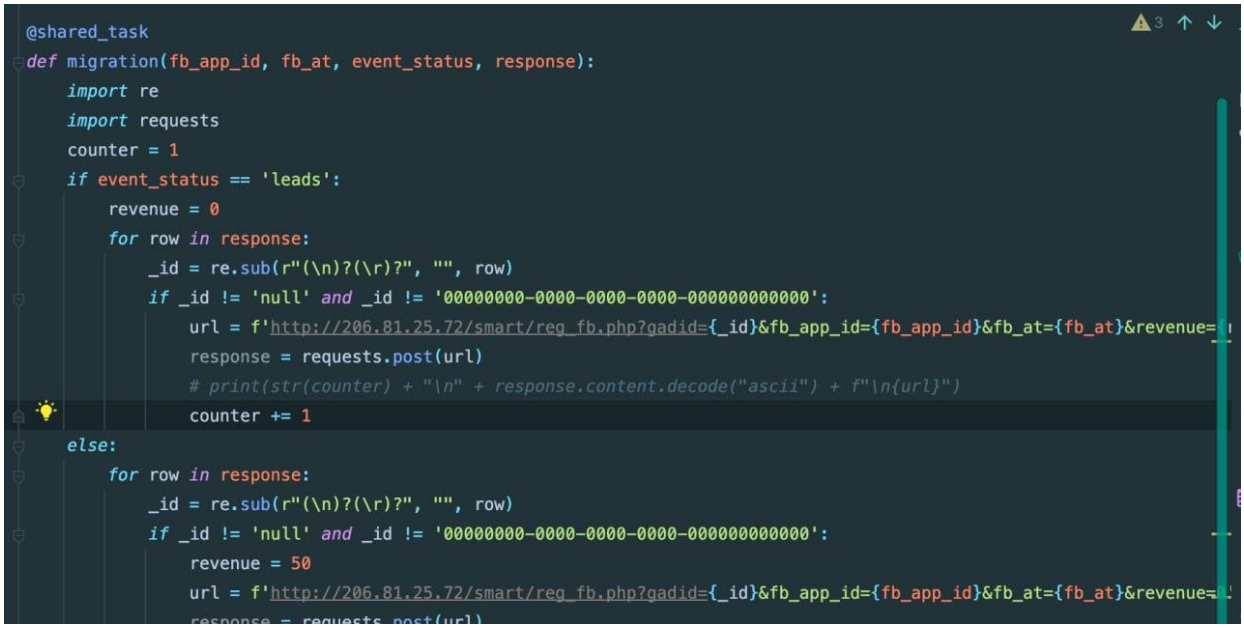
Після заповнення даних форми та відправлення POST запиту, як фоновий процес запускається redis для зберігання черги та celery для безпосереднього виконання процесу перенесення аудиторії, адже для перенесення кожного окремого gadid у новий додаток треба окремо звернутися до facebook по api, передати цілий пакет одразу - неможливо.

Для цього процесу підключаємо redis та celery, створюємо файл tasks, у якому прописуємо саму задачу, яка потрапляє на виконання як фоновий процес.

В процесі так само перевіряємо у файлі ідентифікатор на пустоту чи рівність 00000000-0000-0000-0000-000000000000, у випадку їх трапляння - ігноруємо.

Як бачимо на зображенні (рис.3.31), інформацію про кожного користувача передається на проміжний сервер, який вже в свою чергу отримуючи запит - передає ці дані по API на Facebook.

Після виконання даного скрипта наша аудиторія буде повністю перенесена на новий додаток, який в свою чергу буде готовим до запуску трафіку на нього.



```
@shared_task
def migration(fb_app_id, fb_at, event_status, response):
    import re
    import requests
    counter = 1
    if event_status == 'leads':
        revenue = 0
        for row in response:
            _id = re.sub(r"(\n)?(\r)?", "", row)
            if _id != 'null' and _id != '00000000-0000-0000-0000-000000000000':
                url = f'http://206.81.25.72/smart/reg_fb.php?gadid={_id}&fb_app_id={fb_app_id}&fb_at={fb_at}&revenue=
                response = requests.post(url)
                # print(str(counter) + "\n" + response.content.decode("ascii") + f"\n{url}")
                counter += 1
    else:
        for row in response:
            _id = re.sub(r"(\n)?(\r)?", "", row)
            if _id != 'null' and _id != '00000000-0000-0000-0000-000000000000':
                revenue = 50
                url = f'http://206.81.25.72/smart/reg_fb.php?gadid={_id}&fb_app_id={fb_app_id}&fb_at={fb_at}&revenue=
```

Рис. 3.31

3.16. GP Parser

Розподіл рейтингу мобільного додатку у Google Play може залежати від багатьох факторів, таких як якість додатку, користувацький досвід, рекламні кампанії та інші. Однак, я можу запропонувати загальний розподіл оцінок з урахуванням типових звичок користувачів:

1 зірка: 5%

2 зірки: 10%

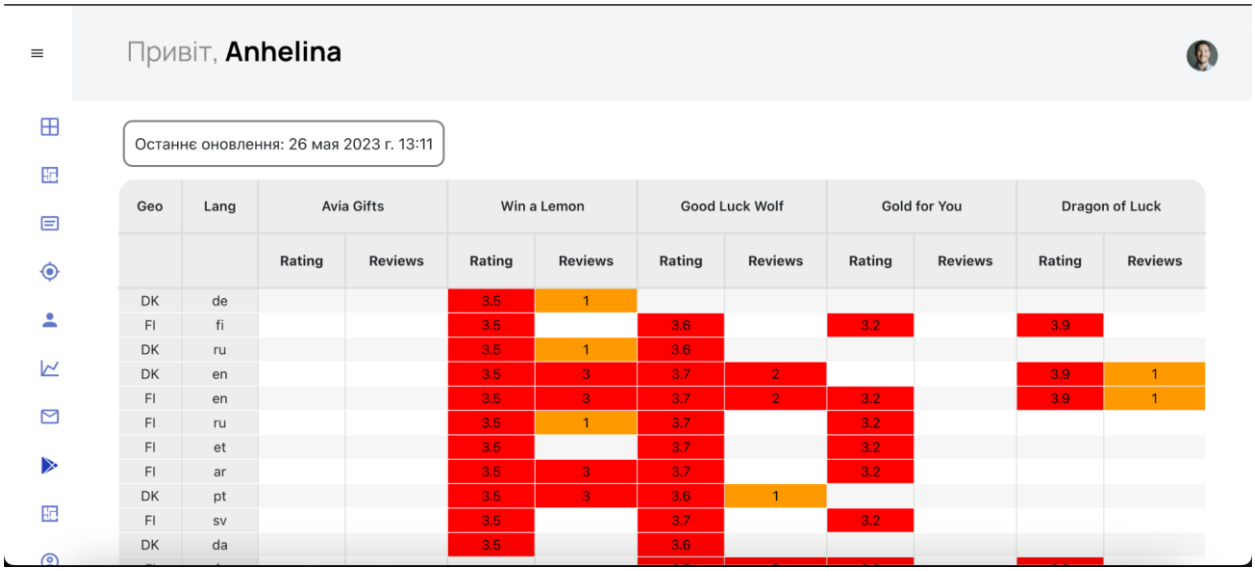
3 зірки: 20%

4 зірки: 30%

5 зірок: 35%

За таким розподілом, середня оцінка буде складати 3.8 із 5, саме така оцінка стане нижнім бар’єром для середньої оцінки.

Відповідно до оцінки очевидно, що наявність негативних відгуків буде мати негативний вплив на конверт та якість трафіку як і низький рейтинг. Тож для повноцінного якісного функціонування технічної сторони воронки розробимо інструмент (рис 3.32), який буде парсити рейтинги на відгуки для кожного мобільного додатку і залежно від оцінки змінювати колір.



Geo	Lang	Avia Gifts		Win a Lemon		Good Luck Wolf		Gold for You		Dragon of Luck	
		Rating	Reviews	Rating	Reviews	Rating	Reviews	Rating	Reviews	Rating	Reviews
DK	de			3.5	1						
FI	fi			3.5		3.6		3.2		3.9	
DK	ru			3.5	1	3.6					
DK	en			3.5	3	3.7	2			3.9	1
FI	en			3.5	3	3.7	2	3.2		3.9	1
FI	ru			3.5	1	3.7		3.2			
FI	et			3.5		3.7		3.2			
FI	ar			3.5	3	3.7		3.2			
DK	pt			3.5	3	3.6	1				
FI	sv			3.5		3.7		3.2			
DK	da			3.5		3.6					

Рис 3.32.

Саму систему розіб’ємо на 5 сторінок (рис 3.33.):

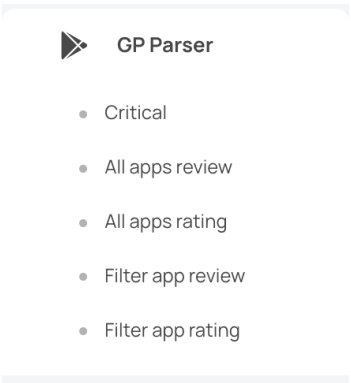


Рис 3.33.

1. Critical - перелік співпадінь країн та мов, на яких рейтинг нижчий за 4.4 та/або кількість негативних коментарів на першій сторінці >= 1
2. All apps review - аналог сторінки “Critical”, але відображає тільки показники кількості негативних відгуків.

3. All apps rating - аналог сторінки “Critical”, але відображає тільки показники рейтингу, нижчого за 4.5

4. Filter app review - аналог до “All apps review”, проте з дропдауну можна вибрати тільки один додаток і побачити інформацію про відгуки стосовно одного конкретного додатку.

5. Filter app rating - аналог до “Filter app review”, тільки інформація відображається стосовно низького рейтингу відповідно.

Розглядаємо дашборди на рис 3.34.

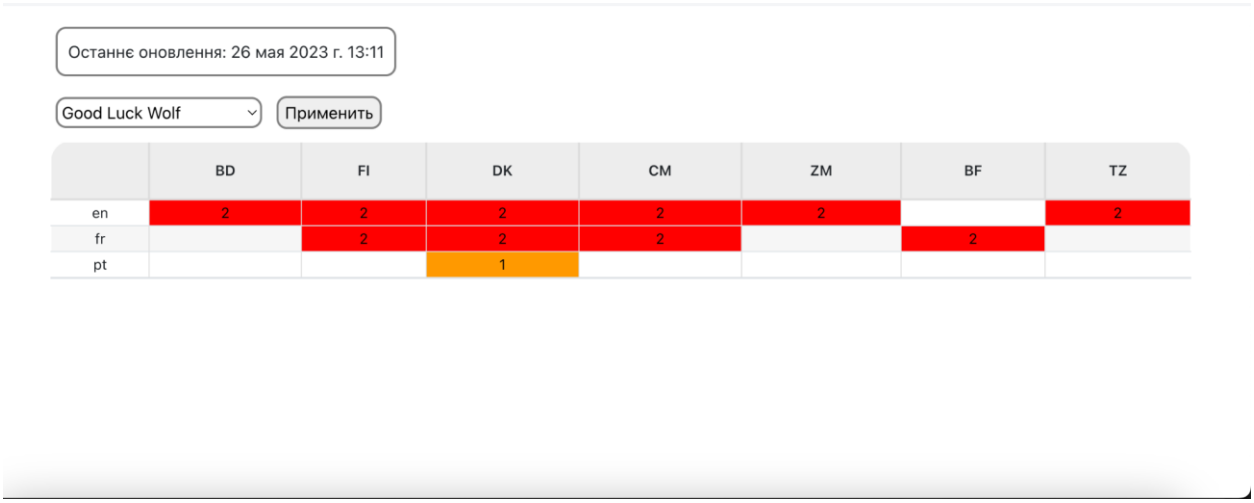


Рис 3.34.

Дашборди на рис 3.34. будуть використовуватися для моніторингу ситуації відділом накрутки компанії, за початок сигналізування про погану ситуацію була обрана оцінка 4.4, так як для накрутки рейтингу у випадку його падіння, експериментальним шляхом було виявлено, що треба від 48 до 96 годин, а за цей час оцінка може впасти нижче за 3.8. Принцип розробки (рис.3.35.):

```python
def get_bundle_id_by_bundle_name(bundle_name: str):
    """Получение id бандла по его названию"""
    bundle = Bundle.objects.filter(bundle_name=bundle_name).first()
    if bundle:
        return bundle.bundle_id
    logger.warning(f'Не удалось получить id по бандлу: {bundle_name}')
    return None


def get_lng_code_from_lng_name(lng_name: str):
    """Получение кода языка по его названию"""
    language = Language.objects.filter(lng_kt=lng_name).first()
    if not language:
        logger.warning(f'Не удалось получить lng_code по языку: {lng_name}')
        return lng_name
    return language.lng_code


def change_actual_from_bundle_data():
    """Удаление данных из таблицы BundleData"""
    BundleData.objects.filter(actual=True).update(actual=False)


def insert_into_bundle_data(bundle, bundle_id, geo, language, stars, count_bad_reviews):
    """Вставка данных в таблицу BundleData"""
    BundleData.objects.create(
        date_time=timezone.now() + timezone.timedelta(hours=3),
```

Рис 3.35.

Даний мікросервіс так само вбудуємо в Django, і створимо як backend інтерфейс для керування сервісом, так і фронт для менеджерів, які будуть займатися накруткою рейтингів та відгуків.

Так як пропарсити понад 1000 сторінок під кожне завантаження сторінки майже неможливо, парсити дані будемо 1 раз на пів години, а результати парсингу будемо записувати в PostgreSQL за допомогою ORM від самого Django.

Механіка Google Play влаштована таким чином, що рейтинги і відгуки для кожної країни відображаються і рахуються окремо, крім того вони так само змінюються залежно від мови, тож знову ж таки звертаємося до Keitaro для отримання таких співпадінь країн з мовою, щоб там дійсно був трафік, знову ж таки для кожного окремого додатку.

Для встановлення відношення кодів мов keitaro та Google Play - створемо модель, яка буде зберігати ці відношення. З backend панелі дашборду це буде виглядати так: (рис.3.36.)

LNG CODE	LNG KT
ar	Arabic
ne	Nepali

Рис 3.36.

Також для самого парсингу створюємо ще одну модель, записи до якої будуть ставати частиною фонового процессу для парсингу.(рис. 3.37.)

```
class Bundle(models.Model):
    bundle_name = models.CharField(max_length=100)
    bundle_id = models.CharField(max_length=100)

    def __str__(self):
        return self.bundle_id
```

Рис 3.37.

За аналогією створимо модель, яка буде зберігати вже пропаршені дані, які будуть актуалізуватися кожні півгодини, тож параметр actual кожен раз буде приймати спочатку значення True, адже дані записуються одразу в момент парсингу, а при наступному, попередні будуть приймати значення False.(рис.3.38.)

```python
class BundleData(models.Model):
    date_time = models.DateTimeField()
    bundle_name = models.CharField(max_length=100)
    bundle_id = models.CharField(max_length=100, null=True)
    geo = models.CharField(max_length=5)
    lang = models.CharField(max_length=50, null=True)
    rating_score = models.FloatField(null=True)
    negative_reviews = models.PositiveIntegerField(null=True)
    actual = models.BooleanField()

    def __str__(self):
        return f'{self.bundle_name}_{self.rating_score}'
```

Рис 3.38.

Після цього будуємо та запускаємо знову ж таки фоновий процес, який буде виконуватися на базі Celery і актуалізувати дані в БД кожні 30 хвилин. (рис.3.39)

DATE TIME	BUNDLE NAME	BUNDLE ID	GEO	LANG	RATING SCORE	NEGATIVE REVIEWS	ACTUAL
9 июня 2023 г. 6:08	Avia Gifts	com.playkot.adve	PK	en	-	-	✓
9 июня 2023 г. 6:08	Avia Gifts	com.playkot.adve	RU	ru	3,2	3	✓
9 июня 2023 г. 6:08	Avia Gifts	com.playkot.adve	IT	it	3,7	1	✓
9 июня 2023 г. 6:08	Fruit Gain	eu.bandainamcoent.mytamagotchiforeve	BR	pt	-	-	✓

Рис 3.39.

Сам процес парсингу побудований наступним чином:

1. Робимо запит до Google Play з урахуванням id-додатку, країни та мови, після чого отримуємо response.
2. response розпаковуємо за допомогою BeautifulSoup
3. Після чого за допомогою regex знаходимо поточний рейтинг та виділяємо наявність негативних коментарів.

Вся механіка роботи описана у файлах, які знаходяться в папці services, адже ці файли є сервіс-файлами проекту.

3.17. Вік

Аудиторія для Тир 1, Тир 2 та Тир 3 може варіюватися за кількома факторами, такими як соціально-економічний статус, дохід, освіта, зацікавленість у конкретних продуктах або послугах та інші демографічні та поведінкові аспекти. Зробити висновки про платежеспособність аудиторії можна на основі досліджень та аналізу наявних даних.

Тир 1 аудиторія зазвичай складається з клієнтів з високим рівнем доходу та статусу. Це можуть бути люди з вищою освітою, високооплачуваними роботами або успішними бізнесменами. Вони мають можливість та готовність витрачати значну суму грошей на розкішні товари, послуги та інші ексклюзивні продукти. Висновок про їх платежеспособність базується на статистичних даних про їх доходи, витрати та споживчі звички.

Тир 2 аудиторія, зазвичай, належить до середнього соціально-економічного статусу. Це можуть бути люди зі стабільним доходом, середньою освітою та робочими місцями середньої оплати праці. Вони можуть мати обмежений бюджет, але все ж бажають мати доступ до якісних товарів та послуг. Висновок про їх платежеспособність може базуватися на аналізі їх фінансової стійкості, витрат та споживчої поведінки.

Тир 3 аудиторія зазвичай представлена людьми з нижчим рівнем доходу та статусу, низькооплачуваними роботами або незайнятими. Вони мають обмежений бюджет та звертають увагу на цінову доступність товарів та послуг. Однак, висновок про їх платежеспособність може бути зроблений на основі досліджень їх фінансового стану, споживчих звичок та здатності витрачати гроші на товари та послуги.

Для зроблення висновків про платежеспособність аудиторії Тир 1, Тир 2 та Тир 3 можна використовувати наступні джерела та методи:

1. Соціально-економічні дані: аналіз соціально-економічних показників, таких як середній рівень доходу, середні витрати на різні товари та послуги, демографічні характеристики та інші статистичні дані можуть дати загальне уявлення про платежеспособність різних аудиторій.
2. Маркетингові дослідження: проведення опитувань, фокус-груп та інших маркетингових досліджень може допомогти зрозуміти витрати та фінансові можливості різних сегментів аудиторії. Ці дослідження можуть розкрити їх споживчі пріоритети, вартість продуктів та послуг, які вони готові придбати.
3. Аналіз платіжних даних: використання аналітики платежів, таких як дані з кредитних карток, електронних платіжних систем та інших джерел, може дати уявлення про платежеспроможність клієнтів у різних категоріях.
4. Сегментація аудиторії: застосування методів сегментації аудиторії на основі демографічних, географічних та поведінкових характеристик може допомогти виділити більш платежеспроможні групи клієнтів.

Комбінація цих джерел та методів дослідження дозволить зробити висновки про те, яка аудиторія є найбільш платежеспроможною в контексті аудиторій Тир 1, Тир 2 та Тир 3. Наприклад, аналіз соціально-економічних даних може вказувати на те, що аудиторія Тир 1, яка має високий рівень доходу та статусу, зазвичай має більший потенціал для витрат на розкішні товари та ексклюзивні послуги.

Потрібно враховувати вік, який використовується при запуску на кожну окрему країну в моменті налаштування рекламної кампанії, адже досягнення певного соціального рівня, який стає комфортним для гри змінюється від країни до країни. З цього випливає, що цей фактор хоч і є впливовим на конверсію с встановлення додатку в депозит, проте він скоріше статичний, але різний для кожної країни.

Якщо ми говоримо про якість, то для кожної країни треба робити поглиблений аналіз відносно пунктів вказаних вище, враховуючи цей факт можна сильно впливати на якість.

3.18. Вага мобільного додатку

Безперечно вага мобільного додатку має бути мінімальною, адже у багатьох країнах досить повільний інтернет, що буде сприяти зменшенню кількості встановлення мобільного додатку і відповідно, негативно впливати на вартість кінцевого результату.

Умови зарахування користувача

Так як ми розглядаємо ситуацію із сторони медібаїнгової компанії, яка зазвичай працює за CPA-моделлю, тож маємо лише декілька ключових умов, які будуть дійсно мати сильний вплив на на конверт:

1. Значення мінімального депозиту
2. Сума виплати за CPA-моделью

Перший параметр сильно впливає на конверт, так як наприклад конверт в умовному Казахстані буде відрізнятися у декілька разів, якщо у одного рекламодавця значення мінімального депозиту буде рівне 1 долару, а у іншого 10 доларів, де мінімальний депозит - це мінімальна сума грошей, яку може внести користувач.

Сума виплати ж зрозуміло, що чим більша, тим більше підвищує рентабельність роботи.

3.19. Вплив дня тижня відсотку конверту з встановлення в депозит

Зробивши поденну вигрузку (рис 3.40.) статистики за 4 перші тижні квітня, можемо помітити середню закономірність рівня конверту з встановлення в депозит.

Day	Campaign	UC (campaign)	Sales	CR dep / install
Monday	Total	55622	16686	30%
Tuesday	Total	49340	17763	36%
Wednesday	Total	51410	18001	35%
Thursday	Total	63212	19599	31%
Friday	Total	65950	24395	37%
Saturday	Total	75430	27912	37%
Sunday	Total	71300	25684	36%

т.

Рис 3.40.

З огляду на вище наведенні дані, можна зробити висновок про найкращі дні для масштабування об’ємів реклами, що у свою чергу буде мати позитивний вплив на рівень конверту.

3.20. Воронки пуш-повідомлень

У сучасному світі мобільні додатки стали потужним інструментом для залучення та утримання користувачів. Для досягнення максимальної кількості депозитів в онлайн-казино, важливо використовувати ефективні стратегії залучення та утримання користувачів, включаючи воронки пуш повідомлень. Однак, успішність таких воронок залежить від ряду факторів, включаючи

події, що відбуваються на платформі Facebook, а також поведінкові звички користувачів щодо покупок мобільних додатків.

1. Наведено фактори, пов'язані з подіями на платформі Facebook.

1) Рекламна стратегія: Успішність воронок пуш повідомлень значно залежить від ефективності рекламної стратегії на платформі Facebook. Ретаргетингові кампанії, використання персоналізованих рекламних повідомлень та налагодження сегментування аудиторії можуть покращити результативність воронок.

2) Актуальність та цінність контенту: важливо, щоб повідомлення, які використовуються в воронках пуш повідомлень, мали актуальний та цінний контент. Контент повинен бути зорієнтований на потреби та інтереси користувачів, що допоможе залучити їхню увагу та спонукати до зроблення депозитів.

3) Взаємодія з користувачами: воронки пуш повідомлень повинні бути спрямовані на підтримку активної взаємодії з користувачами. Це може включати відправку персоналізованих повідомлень, відкриття можливості комунікації зі службою підтримки або надання ексклюзивних пропозицій та бонусів. Забезпечення відповідного рівня взаємодії може стимулювати користувачів до зроблення депозитів та залишення позитивних коментарів.

2. Наведено фактори, пов'язані з поведінковими звичками користувачів щодо покупок мобільних додатків.

1) Відомості про користувачів: збір і аналіз даних про користувачів, включаючи їхні покупки мобільних додатків, може допомогти визначити їхні вподобання та інтереси. Це дозволить налаштувати воронки пуш повідомлень, враховуючи ці вподобання та надавати користувачам персоналізовану інформацію, що може сприяти зробленню депозитів.

2) Сегментування користувачів: розбиття користувачів на сегменти на основі їхньої поведінки щодо покупок мобільних додатків може допомогти

встановити зв'язок між конкретними сегментами та їхньою реакцією на пуш повідомлення. Це дозволить визначити ефективні стратегії воронок пуш повідомлень для кожного сегменту та максимізувати кількість депозитів.

3) Стимулювання покупок: застосування методів стимулювання покупок, таких як пропозиції спеціальних промоакцій, бонусів або знижок, може значно збільшити ймовірність зроблення депозитів користувачами. Використання цих стратегій у воронках пуш повідомлень може залучити увагу та заохотити користувачів до активної участі в грі та зроблення депозитів.

Висновки до розділу 3

В цьому розділі ми приділили увагу побудові технічної воронки продажів для максимально ефективного досягнення цілей. Зробивши аналіз різних факторів, ми можемо зробити наступні висновки. Перш за все, для досягнення максимальної ефективності воронки, мобільний додаток повинен мати мінімальний розмір, щоб уникнути проблем з завантаженням та забезпечити швидкий доступ користувачів. Дизайн додатку також відіграє важливу роль і може бути унікальним або відповідати популярним стандартам на певній геолокації.

Додатково, для покращення воронки продажів важливо звернути увагу на рейтинги та відгуки. Рейтинг додатку повинен бути високим (більше 4 зірок), а негативних відгуків має бути мінімум. Можна використовувати мотивуючі коментарі для стимулювання реєстрації або покупки додатку. Для контролю негативних відгуків був створений продукт, що надає оперативну інформацію кожні 30 хвилин про рейтинги та кількість негативних коментарів.

Крім того, для формування цільової аудиторії в рамках воронки продажів варто використовувати перенос аудиторії з інших додатків. Вибираючи геолокацію та додаток, з якого буде перенесено аудиторію, можна залучити схожу цільову аудиторію, що сприятиме покращенню конверсії.

Застосування нейронних мереж допомагає виділити якісну аудиторію, яка відповідає потребам рекламодавця. При заливі трафіку варто уникати великих скачків трафіку, оскільки це може призвести до зниження конверсії і порушення оптимізації на платформі Facebook.

Важливо правильно настроювати пуш-повідомлення, щоб вони були яскравими і мотивували користувачів до цільових дій, таких як реєстрація або покупка додатку. Додатково, необхідно звернути увагу на платежеспроможний вік цільової аудиторії.

Загалом, використання технічних методів побудови воронки продажів у рамках Facebook трафіку є ефективним підходом для досягнення максимальної кількості депозитів. Аналізуючи фактори, які впливають на воронку, ми можемо зробити висновок, що максимально ефективна воронка вимагає мінімального розміру додатку, унікального або популярного дизайну, високого рейтингу та мінімуму негативних відгуків. Формування цільової аудиторії та правильна настройка пуш-повідомлень також є важливими аспектами.

Отже, зазначений флоу є ідеальним для технічної побудови воронки продажів на платформі Facebook, який дозволяє досягти максимальної ефективності та збільшити кількість депозитів.

4 ЕКОНОМІЧНА ЧАСТИНА. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

4.1 Постановка задачі проектування

У даній роботі метою є побудова і автоматизація технічної воронки для реалізації іздешевлення трафіку для просування онлайн казино. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по трафіку.

Технічні вимоги до програмного продукту є наступні:

- функціонування на пристороях із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидке функціонування та доступ до даних в реальному часі;
- можливість зручного масштабування та обслуговування;
- можливість зменшення витрат на впровадження програмного продукту;

4.2 Обґрунтування функцій програмного продукту

Головна функція F0 – розробка аналітики трафіку, для технічної воронки продажу, яка дозволяє аналізувати різні зміни ключових метрик ефективності при впливі на них. Беручи за основу цю функцію, можна виділити наступні:

F1 – вибір воронки.

F2 – якісний аналіз даних складових воронки.

F3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації.

Функція F1 : а) Через додаток б) Через пряме посилання.

Функція F2 : а) Використання базово-налаштованої технічної воронки. б) Аналітика та оптимізація роботи складових воронки.

Функція F3 : а) Використання шаблонних показників. б) Використання графічних показників.

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

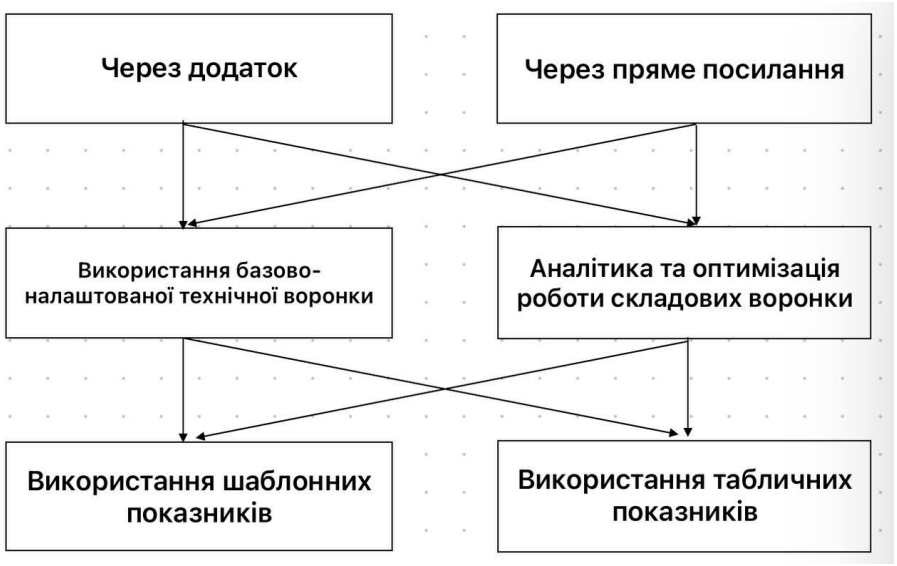


Рис 4.1 - Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантиів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Присутність ретеншину зі сторони медіа-байінгу.	Часті бани додатків.
	Б	Просте в налаштуванні, не потребує окремої компанії розробки додатків.	Не можливий збір аудиторії через брак інформації.
F2	*А*	Не потребує додаткових фінансови інвестицій та людського ресурсу.	Відсутність розширених аналітичних даних дл оптимізації або реструктуризації технічної воронки.
	Б	Зростання показників конверту з встановлення в депозит, якості трафік та зменшення маркетингових затрат.	Збільшення адміністративних витрат.
F3	*А*	Звичні для сприйняття на широкий загал ринку.	Відсутність можливостей кастомізацій метрик.
F3	*Б*	Зручні для візуального сприйняття та поденного аналізу.	Відсутність на графіках інформації про кількість завершених цільових подій.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1 *:*

Перевагу даємо прямому посиланню. Для спрощення роботи варіант А має бути відкинутий.

Функція F2 *:*

Допускається обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F3*:*

Допускається обрання обох варіантів. Найкраще використовувати обидва з них.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F1б – F2а – F3а

F1б – F2б – F3а

F1б – F2а – F3б

F1б – F2б – Fб

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

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

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

- *X1* – швидкість надання даних TDS системи;
- *X2* – об’єм кількості аналізуємих користувачів;
- *X3* – кількість додатків у PlayMarket;
- *X4* – показник конверту із встановлення в депозит.

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

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкість надання даних TDS системи	X1	response/c	50	10	1 сек.
Об’єм кількості аналізуємих користувачів	X2	одиниць	100000	1000000	10000000
Кількість додатків PlayMarket	X3	одиниць	1	20	50
Показник конверту встановлення в депозит	X4	відсотків	3%	8%	15%

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

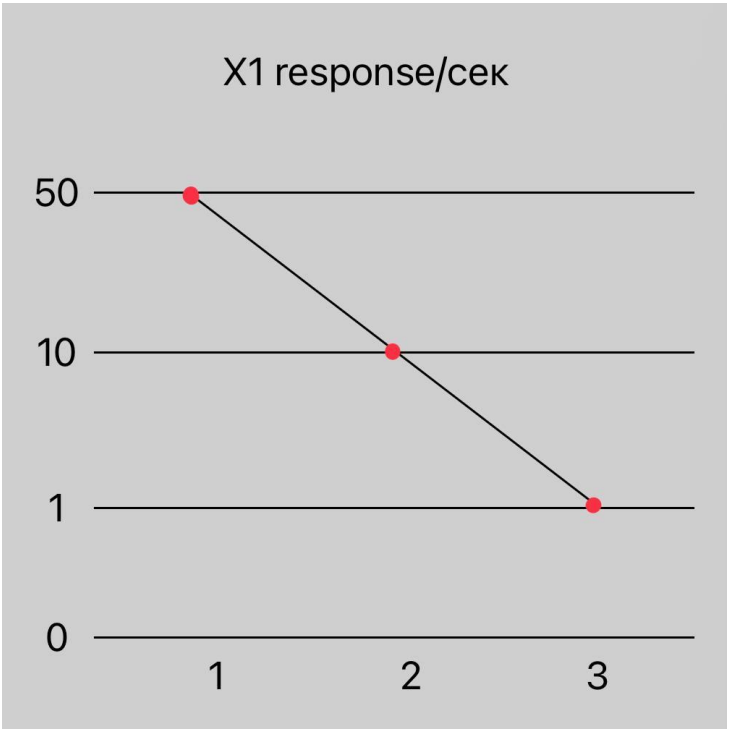


Рисунок 4.2 – X1, швидкодія мови програмування

Рисунок 4.3 – X2, об’єм пам’яті

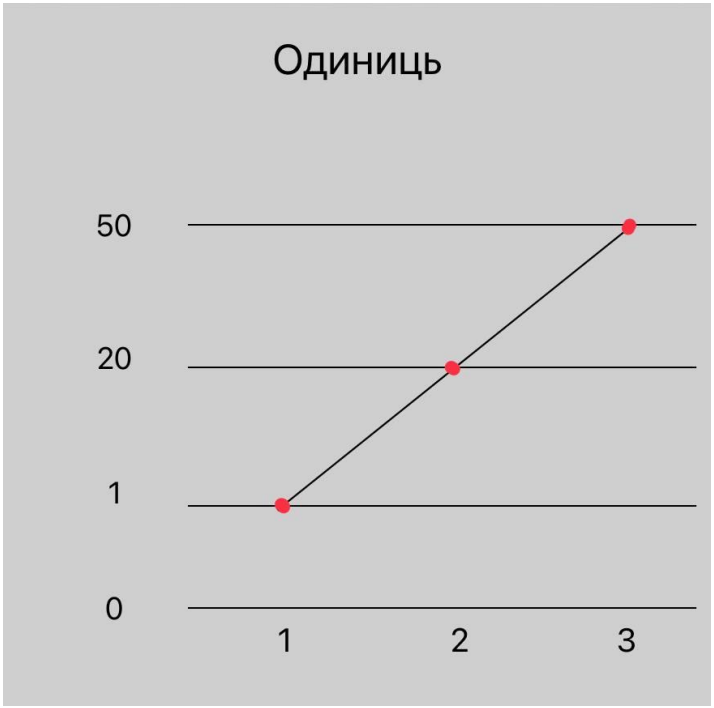


Рисунок 4.4 – X3, час попередньої обробки даних

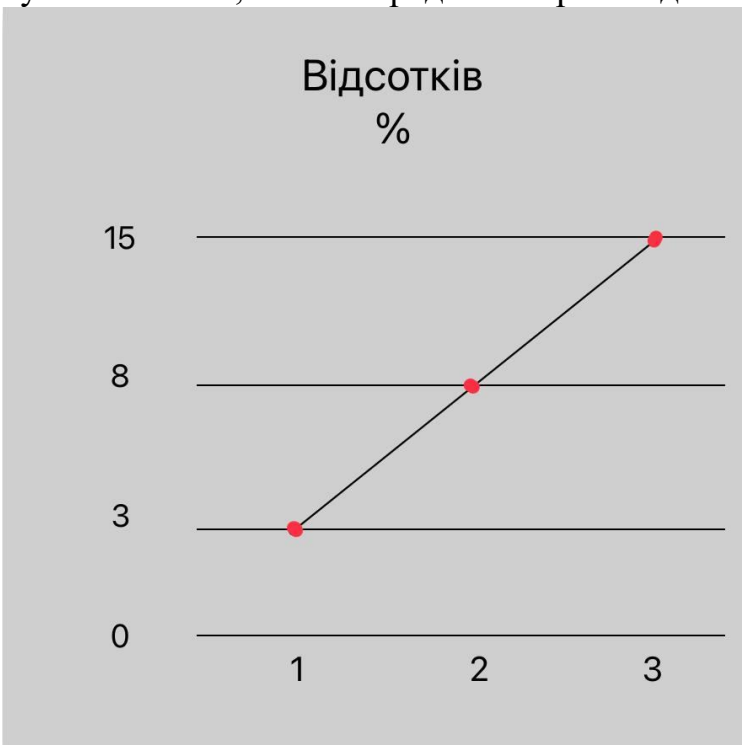


Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порів-няння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкість надання даних TDS системи	response /c	1	2	2	1	2	2	2	12	-5,5	30,25

Продовження таблиці 4.3.

X2	Об’єм кількості аналізуємих користувачів	штука	3	4	3	3	4	4	4	25	7,5	56,25
X3	Кількість додатків PlayMarket	штука	2	1	1	2	1	1	1	9	-8,5	72,25
X4	Показник конверту із встановлення в депозит	відсоток	4	3	4	4	3	3	3	24	6,5	42,25
	Разом		10	10	10	10	10	10	10	70	0	201

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \qquad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \qquad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 201. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 201}{7^2(4^3 - 4)} = 0{,}8204 > W_k = 0{,}82. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,82.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

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

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

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

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

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

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1	0,5	0,5	1,5	3,5	0,25	12,25	0,25	42,87	0,25
X2	0,5	1	1,5	0,5	3,5	0,25	12,25	0,25	42,87	0,25
X3	0,5	1,5	1	0,5	3,5	0,25	12,25	0,25	42,87	0,25
X4	1,5	0,5	0,5	1	3,5	0,25	12,25	0,25	42,87	0,25
Всього:					14	1	49	1	171,4	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об'єм кількості аналізуємих користувачів), *X3* (кількість додатків у PlayMarket) та *X4* показник конверту із встановлення в депозит) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість надання даних TDS системи) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$ – коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

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

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієн вагомості параметра	Коефіцієнт рівня якост
F1	Б	X1	0.7	5	0,25	1,25

Продовження таблиці 4.6.

F2	А	X2	1000000	10000000	0,25	2500000
	Б	X3	40	50	0,25	12,5
F3	А	X4	10	15	0,25	3,75
	Б	X4	1	15	0,25	3,75

За даними з таблиці 4.6 за формулою визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 2500005 ;$$

$$K_{K}2 = 17,5 .$$

$$K_{K}3 = 2500005 .$$

$$K_{K}4 = 17,5 .$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

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

Загальна трудомісткість обчислюється за формулою, де:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.12)$$

1) T_P – трудомісткість розробки ПП;
2) $K_П$ – поправочний коефіцієнт;
3) $K_{СК}$ – коефіцієнт на складність вхідної інформації;
4) K_M – коефіцієнт рівня мови програмування;
5) $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;
6) $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P = 23 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.5. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.85. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 23 \cdot 1.5 \cdot 0.85 = 29{,}325 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =18 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.6:

$$T_2 = 18 \cdot 0.9 \cdot 0.6 = 9.72 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (29{,}325 + 9.72 + 4.8 + 9.72) \cdot 8 = 428{,}52 \text{ людино-годин.}$$

$$T_{II} = (29{,}325 + 9.72 + 6.91 + 9.72) \cdot 8 = 445{,}4 \text{ людино-годин}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 15000 грн., один аналітик в області даних з окладом 23000. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (4.13)$$

, де М – місячний оклад працівників;

– кількість робочих днів тиждень;

– кількість робочих годин в день.

$$СЧ = \frac{15000 + 15000 + 23000}{3 \cdot 21 \cdot 8} = 105{,}16 \text{ грн.} \qquad (4.14)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot T_i \cdot К_Д, \qquad (4.15)$$

де $С_Ч$– величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 105.16 · 428,52 · 1.2 = 54075,80 грн.

II. $С_{ЗП}$ = 105.16 · 445,4 · 1.2 = 56205,92 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 54075,80 \cdot 0.22 = 11896,676$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 56205,92 \cdot 0.22 = 12365,30$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 15000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 15000 \cdot 0,2 = 36000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 36000 \cdot (1 + 0.2) = 43200 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 43200 \cdot 0,22 = 9504 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 3000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 3000 = 1050 \text{ грн.,}$$

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

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 3000 \cdot 0.08 = 336 \text{ грн.,}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

$$= 627,2 \text{ години,}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,25 \cdot 0,3 \cdot 4,79 = 225,32 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 3000 \cdot 0,67 = 2010 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = 43200 + 9504 + 1050 + 336 + 216,62 + 2010 = 56316,62 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 55770,62 / 627,2 = 88,91 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$\text{I. } С_М = 88,91 \cdot 428,52 = 38099,71 \text{ грн.}$$

$$\text{II. } С_М = 88,91 \cdot 445,4 = 39600,51 \text{ грн.}$$

Накладні витрати складають 44% від заробітної плати:

$$\text{I. } С_Н = 54075,80 \cdot 0,44 = 23793,35 \text{ грн.}$$

$$\text{II. } С_Н = 56205,92 \cdot 0,44 = 24730,60 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$\text{I. } С_{ПП} = 54075,80 + 11896,676 + 38099,71 + 23793,35 = 127865,53 \text{ грн.}$$

$$\text{II. } С_{ПП} = 56205,92 + 12365,30 + 39600,51 + 24730,60 = 132902,33 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР1} = 2500005 / 127865,53 = 19,5518292 \cdot 10^{-},$$

$$К_{ТЕР2} = 17,5 \ / 132902,33 = 0,00013168 \cdot 10^{-}.$$

$$К_{ТЕР3} = 2500005/ 127865,53 \ = 19,5518292 \cdot 10^{-}.$$

$$К_{ТЕР4} = 17,5 \ / 132902,33 = 0,00013168 \cdot 10^{-}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР13}= 19,5518292 \cdot 10^{-}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 6,851 \ \cdot \ 10^{-5}$.

Висновки до розділу 4

У четвертому розділі дипломної роботи було проведено детальний функціонально-вартісний аналіз розглянутого програмного продукту. Для цього були визначені та оцінені основні функції створеного продукту.

Процес функціонально-вартісного аналізу включав дослідження функціональності програмного комплексу, а також оцінку витрат, пов'язаних з розробкою та підтримкою програмного продукту. Результатом аналізу була здійснена оцінка основних функцій програмного продукту з точки зору їх значущості та вартості.

На основі проведеного аналізу було здійснено вибір варіанта реалізації програмного продукту.

ВИСНОВКИ

Ця дослідницька робота покращить розуміння трафіку з джерела Фейсбук у галузі геймблінгу та надасть підґрунтя для подальшого вдосконалення маркетингових стратегій та рекламних компаній. Результати аналізу можуть бути корисні для компаній, що працюють у цій сфері, допомагаючи їм залучати більше якісних користувачів та досягати більш високої конверсії.

Після проведення аналітики з технічною командою та командою мобільної розробки, завдяки побудованим інструментам аналітики та розробленим системам оптимізації, як технічної складової так і людського виробничого фактору, було виявлено ряд покращень, які напряму впливають на валовий прибуток та чистий прибуток, що є наслідками збільшення конверту та якості трафіку. При підвищенні якості, зростає рівень виплати за кожного користувача, а при зростанні конверту, відповідно, зменшується вартість за заохочення клієнту. Коливання об'ємів - до 20%. Перелив аудиторії - до 20% на конверт та до 15% на виплату за кожного гравця. Google Play Parser - від 3% до 20%, адже рейтинги та вігуки можуть суттєво відрізнятися.

Тож як бачимо, рентабельність від витрат може зрости на понад 40% завдяки вище зазначеним оптимізаціям. Додатково, кожен з інструментів має можливість покращення якості та швидкості роботи. Можливо вдосконалення всієї воронки додатковими мікросервісами в інших точках технічної воронки.

Заключно, ця робота підкреслює важливість постійного аналізу та вдосконалення маркетингових стратегій для досягнення успіху в сфері гемблінгу

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Smith J. Digital Marketing Strategies for the Gambling Industry. London: Marketing Insights Publishing. 2021.
2. Johnson R. The Ultimate Guide to Facebook Advertising. New York: McGraw-Hill Education. 2020.
3. Anderson P. Google Ads for Beginners: A Step-by-Step Guide to PPC Advertising. San Francisco: Anderson Publishing. 2019.
4. Lee S. App Store Optimization: A Comprehensive Guide to Boosting Your App's Visibility and Downloads. Boston: Lee Press. 2021.
5. Jones E. Search Engine Optimization Made Easy: A Practical Guide for Website Owners. Chicago: Jones Publications. 2020.
6. Brown K. Mobile Advertising: Supercharge Your Campaigns and Drive Results. Los Angeles: Brown Books. 2021.
7. Clark A. The Psychology of Online Gambling: A Comprehensive Analysis of Consumer Behavior. Oxford: Oxford University Press. 2022.
8. Martin D. The Impact of Social Media Advertising on the Gambling Industry. Sydney: Martin Publishing. 2021.
9. Wilson L. Ethical Considerations in Gambling Advertising: Balancing Business and Social Responsibility. Melbourne: Wilson Press 2022.
10. Affiliate Guard Dog - https://www.affiliateguarddog.com/
11. Affiliate Bible - https://www.affiliatebible.com/
12. GPWA (Gambling Portal Webmasters Association) - https://www.gpwa.org/
13. iGB Affiliate - https://www.igbaffiliate.com/
14. CalvinAyre.com - https://calvinayre.com/
15. AffiliateInsider - https://affiliateinsider.com/
16. “Палай” медіа - https://palai.media/

17. LC Work - https://logincasino.work/ua/

ДОДАТОК А
ЛІСТИНГ ПРОГРАМИ

kt_apps_collector.py

```
from predictorV2.collectors.kt_report import KTReport
from datetime import datetime, timedelta

import requests


class KtAppsReport(KTReport):

    def __init__(self, bundle_name: str, date_from: datetime =
datetime.now() - timedelta(days=1), date_to: datetime = datetime.now()):
        super().__init__(date_from, date_to)
        self.bundle_name = bundle_name

    def _get_json_data(self, date_from: str, date_to: str) -> list:
        json_data = {
            'range': {
                'interval': 'custom_date_range',
                'timezone': 'Europe/Kyiv',
                'from': date_from,
                'to': date_to,
            },
            'columns': [
                'sub_id',
                'ip',
                'datetime',
                'country',
                'sub_id_12',
                'campaign',
                'stream',
                'destination',
                'source',
                # 'sub_id_1',
                'sub_id_2',
                # 'sub_id_3',
                # 'sub_id_4',
                # 'sub_id_5',
                # 'sub_id_6',
                # 'sub_id_7',
                'sub_id_8',
                'sub_id_9',
                # 'sub_id_10',
                # 'sub_id_11',
                # 'sub_id_14',
                # 'sub_id_13',
```

```
            # 'sub_id_15',
            'os_version',
            'extra_param_1',
            'extra_param_2',
            'device_model',
            'extra_param_3',
            'extra_param_4',
            'external_id',
            'creative_id',
            'extra_param_6',
            'extra_param_7',
            'extra_param_8',
            'extra_param_9',
            'offer_id',
            'is_sale',
            'is_lead',
        ],
        'metrics': [],
        'grouping': [],
        'filters': [
            # {
            #     'name': 'sub_id_9',
            #     'operator': 'CONTAINS',
            #     'expression': '',
            # },
            # {
            #     'name': 'campaign_id',
            #     'operator': 'NOT_IN_LIST',
            #     'expression': [
            #         1,
            #         16,
            #     ],
            # },
            {
                'name': 'source',
                'operator': 'CONTAINS',
                'expression': 'acebook',
            },
            {
                'name': 'sub_id_12',
                'operator': 'EQUALS',
                'expression': self.bundle_name,
            },
        ],
        'sort': [
            {
                'name': 'datetime',
                'order': 'desc',
            },
        ],
        # 'limit': 250,
        'offset': 0,
    }
```

```
        response = requests.post(
            'https://domain.com/admin_api/v1/clicks/log',
            headers=self.headers,
            json=json_data,
        )
        try:
            return response.json().get('rows')
        except requests.exceptions.JSONDecodeError:
            print('JSONDecodeError при получении данных с kt')
            return []
```

kt_report.py

```
import time
from datetime import datetime, timedelta
import requests
from loguru import logger


class KTReport:

    def __init__(self, date_from: datetime, date_to: datetime):
        self.date_from = date_from
        self.date_to = date_to
        self.headers = {
            'accept': 'application/json',
            'Api-Key': 'api_key_value',
            'Content-Type': 'application/json',
        }

    def get_rows(self, timedelta_days: int = 3) -> list:
        rows = []
        # generator = (_ for _ in ())
        temp_date_from = self.date_from.replace(minute=0, hour=0, second=0,
microsecond=0)
        temp_date_to = self.date_to.replace(minute=0, hour=0, second=0,
microsecond=0)
        # timedelta_days = 3
        while temp_date_from < temp_date_to:
            date_from = temp_date_from.strftime('%Y-%m-%d %H:%M')
            date_to = (temp_date_from +
timedelta(days=timedelta_days)).strftime('%Y-%m-%d %H:%M')
            logger.debug(f'get rows from {date_from} to {date_to}')
            try:
                json_data = self._get_json_data(date_from, date_to)
            except requests.exceptions.SSLError:
                time.sleep(10)
                logger.warning('SSL error, sleep 10 seconds and try again')
                try:
                    json_data = self._get_json_data(date_from, date_to)
                except requests.exceptions.SSLError:
                    logger.error('SSL error again, skip this date')
                    continue
```

```
            rows.extend(json_data)
            # new_generator = (_ for _ in json_data)
            # generator = itertools.chain(generator, new_generator)
            temp_date_from += timedelta(days=timedelta_days)
        # return generator
        return rows

    def _get_json_data(self, date_from: str, date_to: str) -> list:
        pass
```

os_collector.py

```
from io import BytesIO
import pandas as pd

import requests
import time


class OneSignalMixin:
    @staticmethod
    def get_all_bundle_data():
        response =
requests.get('http://domain_os.com/apps/all/?key=api_key_value')
        return response.json().get('data')


class OneSignalAPI(OneSignalMixin):
    def __init__(self, app_id, auth_key):
        self.url =
f"https://onesignal.com/api/v1/players/csv_export?app_id={app_id}"
        self.headers = {
            "accept": "application/json",
            "Authorization": f"Basic {auth_key}",
            "content-type": "application/json"
        }
        self.payload = {
            "extra_fields": ["external_user_id", "country"],
            # "segment_name": "Subscribed Users", # если закомм - то єто за
все время

        }

    def get_dict_data_from_os(self):
        csv_file_url = self._get_csv_file_url()
        content = self._get_bytes_csv_file(csv_file_url)
        dict_data = self._process_gzip_file_to_dict(content)
        return dict_data

    def _get_csv_file_url(self):
        while True:
            response = requests.post(self.url, json=self.payload,
headers=self.headers)
```

```
            if 'errors' in response.text:
                print(response.text)
                time.sleep(5)
            else:
                print('True')
                break
        return response.json().get('csv_file_url')

    def _get_bytes_csv_file(self, url):
        print(url)
        while True:
            r = requests.get(url)
            if '<Error>' in r.text:
                time.sleep(5)
                print('Trying')
            else:
                print('True')
                break
        return r.content

    def _process_gzip_file_to_dict(self, content_data: bytes):
        byte_stream = BytesIO(content_data)
        data = pd.read_csv(byte_stream, compression='gzip')
        result = data.to_dict(orient='records')
        return result
```

os_collector.py

```
from sklearn.ensemble import RandomForestRegressor
from loguru import logger

import pandas as pd


class Predictor:
    def __init__(self, target_country_code: str, target_sub_id_12: str,
country_df: pd.DataFrame):
        self.target_country_code = target_country_code
        self.target_sub_id_12 = target_sub_id_12
        self.country_df = country_df

    def get_predict_df(self):
        try:
            self._preprocess_data()
            self._train_model()
            self._predict()

            self.country_df =
self.country_df[(self.country_df['predicted_value'] >= 0.0) &
(self.country_df['predicted_value'] <= 1.0)]
            return self.country_df
        except ValueError:
            logger.warning(f'No data for {self.country_df}')
```

```
            return pd.DataFrame()

    def _preprocess_data(self):

        self.country_df['last_active'] =
pd.to_datetime(self.country_df['last_active'])
        self.country_df['created_at'] =
pd.to_datetime(self.country_df['created_at'])
        self.country_df['time_since_install'] =
(self.country_df['last_active'] -
self.country_df['created_at']).dt.total_seconds()
        self.country_df['average_session_duration'] =
(self.country_df['playtime'] / self.country_df['session_count'])
        self.country_df =
self.country_df[self.country_df['time_since_install'] >= 0]
        self.country_df = self.country_df[self.country_df['is_lead']]
        self.country_df = self.country_df[self.country_df['sub_id_12'] ==
self.target_sub_id_12]

    def _train_model(self):
        X = self.country_df[['time_since_install', 'playtime',
'average_session_duration']]
        y = self.country_df['is_sale']
        self.regressor = RandomForestRegressor(n_estimators=100,
random_state=420)
        self.regressor.fit(X, y)

    def _predict(self):
        self.country_df['predicted_value'] =
self.regressor.predict(self.country_df[['time_since_install', 'playtime',
'average_session_duration']])
```

predictor.py

```
from sklearn.ensemble import RandomForestRegressor
from loguru import logger

import pandas as pd


class Predictor:
    def __init__(self, target_country_code: str, target_sub_id_12: str,
country_df: pd.DataFrame):
        self.target_country_code = target_country_code
        self.target_sub_id_12 = target_sub_id_12
        self.country_df = country_df

    def get_predict_df(self):
        try:
            self._preprocess_data()
            self._train_model()
            self._predict()
```

```
            self.country_df =
self.country_df[(self.country_df['predicted_value'] >= 0.0) &
(self.country_df['predicted_value'] <= 1.0)]
            return self.country_df
        except ValueError:
            logger.warning(f'No data for {self.country_df}')
            return pd.DataFrame()

    def _preprocess_data(self):

        self.country_df['last_active'] =
pd.to_datetime(self.country_df['last_active'])
        self.country_df['created_at'] =
pd.to_datetime(self.country_df['created_at'])
        self.country_df['time_since_install'] =
(self.country_df['last_active'] -
self.country_df['created_at']).dt.total_seconds()
        self.country_df['average_session_duration'] =
(self.country_df['playtime'] / self.country_df['session_count'])
        self.country_df =
self.country_df[self.country_df['time_since_install'] >= 0]
        self.country_df = self.country_df[self.country_df['is_lead']]
        self.country_df = self.country_df[self.country_df['sub_id_12'] ==
self.target_sub_id_12]

    def _train_model(self):
        X = self.country_df[['time_since_install', 'playtime',
'average_session_duration']]
        y = self.country_df['is_sale']
        self.regressor = RandomForestRegressor(n_estimators=100,
random_state=420)
        self.regressor.fit(X, y)

    def _predict(self):
        self.country_df['predicted_value'] =
self.regressor.predict(self.country_df[['time_since_install', 'playtime',
'average_session_duration']])
```

merger.py

```
from sklearn.ensemble import RandomForestRegressor
from loguru import logger

import pandas as pd


class Predictor:
    def __init__(self, target_country_code: str, target_sub_id_12: str,
country_df: pd.DataFrame):
        self.target_country_code = target_country_code
        self.target_sub_id_12 = target_sub_id_12
        self.country_df = country_df
```

```
    def get_predict_df(self):
        try:
            self._preprocess_data()
            self._train_model()
            self._predict()

            self.country_df =
self.country_df[(self.country_df['predicted_value'] >= 0.0) &
(self.country_df['predicted_value'] <= 1.0)]
            return self.country_df
        except ValueError:
            logger.warning(f'No data for {self.country_df}')
            return pd.DataFrame()

    def _preprocess_data(self):

        self.country_df['last_active'] =
pd.to_datetime(self.country_df['last_active'])
        self.country_df['created_at'] =
pd.to_datetime(self.country_df['created_at'])
        self.country_df['time_since_install'] =
(self.country_df['last_active'] -
self.country_df['created_at']).dt.total_seconds()
        self.country_df['average_session_duration'] =
(self.country_df['playtime'] / self.country_df['session_count'])
        self.country_df =
self.country_df[self.country_df['time_since_install'] >= 0]
        self.country_df = self.country_df[self.country_df['is_lead']]
        self.country_df = self.country_df[self.country_df['sub_id_12'] ==
self.target_sub_id_12]

    def _train_model(self):
        X = self.country_df[['time_since_install', 'playtime',
'average_session_duration']]
        y = self.country_df['is_sale']
        self.regressor = RandomForestRegressor(n_estimators=100,
random_state=420)
        self.regressor.fit(X, y)

    def _predict(self):
        self.country_df['predicted_value'] =
self.regressor.predict(self.country_df[['time_since_install', 'playtime',
'average_session_duration']])
```

merger.py

```
from predictorV2.collectors.kt_apps_collector import KtAppsReport
from predictorV2.collectors.os_collector import OneSignalAPI
from predictorV2.predictors.predictor import Predictor
from predictorV2.merger import Merger

from datetime import datetime
from loguru import logger
```

```
import os.path as osp
import pandas as pd
import warnings
import json
import os


class Report:
    warnings.filterwarnings("ignore")

    def __init__(self, date_from: datetime, date_to: datetime):
        self.date_from = date_from
        self.date_to = date_to

    def create_data_in_csv(self):
        # if os.path.exists('predicted.csv'):
        #     os.remove('predicted.csv')
        bundles_data = OneSignalAPI.get_all_bundle_data()

        merger = Merger()

        for bundle in bundles_data:
            logger.debug(f'Get data from {bundle["bundle_name"]}')
            kt_apps = KtAppsReport(bundle_name=bundle['bundle_name'],
date_from=self.date_from, date_to=self.date_to)
            kt_apps_data = kt_apps.get_rows(timedelta_days=3)

            one_signal = OneSignalAPI(app_id=bundle['onesignal_id'],
auth_key=bundle['api_key'])
            os_data = one_signal.get_dict_data_from_os()
            if len(kt_apps_data) > 0 and len(os_data) > 0:
                logger.debug(f'Merge data for {bundle["bundle_name"]}')
                logger.debug(f'kt_apps_data: {len(kt_apps_data)}')
                logger.debug(f'os_data: {len(os_data)}')
                merge_data = merger.merge_kt_apps_with_os(kt_apps_data,
os_data)
                filtered_countries, merge_data_filtered =
self._get_filtered_countries(merge_data)
                concat_df = pd.DataFrame()
                for country in filtered_countries:
                    country_df =
merge_data_filtered[merge_data_filtered['country_code'] == country]
                    logger.debug(f'Predict country
{country}({len(country_df)}) for {bundle["bundle_name"]}')
                    predictor = Predictor(country, bundle['bundle_name'],
country_df)
                    concat_df = pd.concat([concat_df,
predictor.get_predict_df()])

                if not concat_df.empty:
                    logger.debug(f'Writing data for
{bundle["bundle_name"]}')
```

```
                    concat_df.to_csv(f'predicted.csv', index=False,
mode='a', header=not osp.isfile(f'predicted.csv'))
                else:
                    logger.debug(f'No writing data for
{bundle["bundle_name"]}')
                # concat_df.to_excel('test.xlsx')
            else:
                logger.debug(f'No merge data for {bundle["bundle_name"]}')
                logger.debug(f'kt_apps_data: {len(kt_apps_data)}')
                logger.debug(f'os_data: {len(os_data)}')

    @staticmethod
    def _get_filtered_countries(merge_data: pd.DataFrame, count: int = 500)
-> tuple[list[str], pd.DataFrame]:
        merge_data = merge_data[merge_data['external_user_id'] !=
"00000000-0000-0000-0000-000000000000"]
        count_by_country = merge_data['country_code'].value_counts()
        filtered_countries = count_by_country[count_by_country >=
count].index.tolist()

        merge_data_filtered =
merge_data[merge_data['country_code'].isin(filtered_countries)]
        return filtered_countries, merge_data_filtered


if __name__ == '__main__':
    r = Report(date_from=datetime(year=2022, month=11, day=24),
date_to=datetime(year=2023, month=5, day=24))
    r.create_data_in_csv()
```

wwldashboard > celery.py

```
import os
from celery import Celery
from celery.schedules import crontab

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'wwldashboard.settings')

app = Celery('wwldashboard')

app.config_from_object('django.conf:settings', namespace='CELERY')
app.conf.timezone = 'Europe/Kiev'

app.conf.beat_schedule = {
    'fill_data_to_db': {
        'task': 'gp_parser.tasks.fill_data_to_db',
        'schedule': crontab(minute=0, hour="*"),  # every hour at 0 minutes
    },
}


app.autodiscover_tasks()
```

audmigrations > views.py

```
import os
import pandas as pd
from itertools import chain

from threading import Thread

from django.shortcuts import render, redirect
from django.contrib import messages
from django.views import View

from .forms import AddAudience, CreateMigrationRequestForm
from .keitaro_aud import KTWorker

from .utils.helper import handle_uploaded_file
from .utils.orm_utils import create_audience
from .models import Audience
from . import tasks
from django.conf import settings


class CreateMigrationAudience(View):
    template_name = 'add_audience.html'
    form = AddAudience

    def get(self, request):
        context = {'form': self.form}
        return render(request, self.template_name, context)

    def post(self, request):
        form = self.form(request.POST, request.FILES)
        if form.is_valid():
            cleaned = form.cleaned_data
            handle_uploaded_file(request.FILES['file'])
            create_audience(cleaned['aud_name'], cleaned['file'].name)
            messages.success(request, 'Новая база успешно добавлена!')
        else:
            messages.error(request, 'Ошибка валидации')
        return redirect('add_new_audience')


class CreateMigrationQuery(View):
    template_name = 'add_migration_query.html'
    form = CreateMigrationRequestForm

    def get(self, request):
        context = {'form': self.form, 'audiences': Audience.objects.all()}
        return render(request, self.template_name, context)

    def post(self, request):
        form = self.form(request.POST)
        choosen_audience_id = request.POST['choosen_audience']
```

```
        if form.is_valid():
            cleaned = form.cleaned_data
            if cleaned['event'] == 'reg':
                event = 'leads'
            else:
                event = 'sales'
            file_name =
Audience.objects.get(id=choosen_audience_id).file_name
            df = pd.read_csv(os.path.join(settings.BASE_DIR,
f'audmigrations/static/upload/{file_name}'))
            user_data_list = list(chain.from_iterable(df.values.tolist()))
            tasks.migration.delay(fb_app_id=event,
fb_at=cleaned['fb_app_id'], event_status=cleaned['fb_at'],
response=user_data_list)

            messages.success(request, 'Задачи были отправлены в
обработку!')
        else:
            messages.error(request, 'Ошибка валидации')

        context = {'form': form}
        return render(request, self.template_name, context)
```

audmigrations > tasks.py

```
from celery import shared_task


@shared_task
def migration(fb_app_id, fb_at, event_status, response):
    import re
    import requests
    counter = 1
    if event_status == 'leads':
        revenue = 0
        for row in response:
            _id = re.sub(r"(\n)?(\r)?", "", row)
            if _id != 'null' and _id != '00000000-0000-0000-0000-
000000000000':
                url =
f'http://206.81.25.72/smart/reg_fb.php?gadid={_id}&fb_app_id={fb_app_id}&fb
_at={fb_at}&revenue={revenue}'
                response = requests.post(url)
                # print(str(counter) + "\n" +
response.content.decode("ascii") + f"\n{url}")
                counter += 1
    else:
        for row in response:
            _id = re.sub(r"(\n)?(\r)?", "", row)
            if _id != 'null' and _id != '00000000-0000-0000-0000-
000000000000':
                revenue = 50
```

```
                url =
f'http://206.81.25.72/smart/reg_fb.php?gadid={_id}&fb_app_id={fb_app_id}&fb
_at={fb_at}&revenue=0'
                response = requests.post(url)
                # print(str(counter) + "\n" +
response.content.decode("ascii") + f"\n{url}")
                url =
f'http://206.81.25.72/smart/pur_fb.php?gadid={_id}&fb_app_id={fb_app_id}&fb
_at={fb_at}&revenue={revenue}'
                response = requests.post(url)
                # print(str(counter) + "\n" +
response.content.decode("ascii") + f"\n{url}")
                counter += 1
```

audmigrations > models.py

```
from django.db import models


class Audience(models.Model):
    audience_name = models.CharField(max_length=50, verbose_name="Название
аудитории")
    file_name = models.FileField(upload_to='audmigrations/static/upload/',
verbose_name="Файл аудитории")

    def __str__(self):
        return self.audience_name
```

audmigrations > keitaro_aud.py

```
from datetime import datetime, timedelta
from django.conf import settings

import requests


class KTWorker:

    def create_resp(self, event_name, fb_app_id, fb_at, rows):
        from . import tasks
        tasks.migration.delay(fb_app_id=fb_app_id, fb_at=fb_at,
event_status=event_name, response=rows)
```

audmigrations > forms.py

```
from django import forms


# Добавление аудитории форма
class AddAudience(forms.Form):
    dictionary = {
        'api_keys': {
            'class': "form-control",
```

```
            'id': "title_id",
            'placeholder': "aqw-erf-dsx\nqwe-rty-uio..."
        },

    }
    aud_name = forms.CharField(label="Название аудитории", max_length=50,

widget=forms.TextInput(attrs=dictionary['api_keys']))
    file = forms.FileField()


class CreateMigrationRequestForm(forms.Form):
    dictionary = {
        'fb_app_id': {
            'class': "form-control",
            'id': "title_id",
            'placeholder': "Title"
        },
        'fb_at': {
            'class': "form-control",
            'id': "title_id",
            'placeholder': "Title"
        },

    }

    events = (
        ('reg', 'reg'),
        ('dep', 'dep')
    )

    fb_app_id = forms.CharField(required=True, max_length=35,
widget=forms.TextInput(attrs=dictionary['fb_app_id']))
    fb_at = forms.CharField(required=True, max_length=35,
widget=forms.TextInput(attrs=dictionary['fb_at']))
    event = forms.ChoiceField(choices=events, required=True)
```

crreview > views.py

```
from crreview.services.interactive_graph import PlotlyInteractiveGraph
from crreview.services.kt_statistics import KTStatistic
from crreview.services.kt_geo_data import KTGeoData
from django.shortcuts import render, redirect
from django.views import View


from datetime import datetime


class CrReview(View):

    def get(self, request):
        all_offers = KTStatistic.get_all_offers()
```

```
        all_geo_data = KTGeoData.get_all_geo_name()

        context = {
            'today': str(datetime.now().date()),
            'all_offers': all_offers.values(),
            'all_geo_data': all_geo_data[1:],
        }

        if request.session.get('date_from') and
request.session.get('date_to'):
            plot_div = PlotlyInteractiveGraph.create_simple_graph(
                date_from=request.session.get('date_from'),
                date_to=request.session.get('date_to'),

unique_clicks_data=request.session.get('unique_clicks_data'),
                inst_lead_data=request.session.get('inst_lead_data'),
                inst_sale_data=request.session.get('inst_sale_data'),
                lead_sale=request.session.get('lead_sale'),
            )

            context = {
                'today': str(datetime.now().date()),
                'all_offers': all_offers.values(),
                'all_geo_data': all_geo_data[1:],
                'plot_div': plot_div,
            }

            for key in ['date_from', 'date_to', 'unique_clicks_data',
'inst_lead_data', 'inst_sale_data', 'lead_sale']:
                del request.session[key]

        return render(request, 'crreview.html', context)

    def post(self, request):
        kt_statistic = KTStatistic(
            social=request.POST.get('social'),
            offer=request.POST.get('offer'),
            bundle_name=request.POST.get('bundle'),
            geo=request.POST.get('geo'),
            date_from=request.POST.get('date_from'),
            date_to=request.POST.get('date_to')
        )
        unique_clicks_data, inst_lead_data, inst_sale_data, lead_sale =
kt_statistic.get_statistic_by_metrix()

        request.session['date_from'] = request.POST.get('date_from')
        request.session['date_to'] = request.POST.get('date_to')
        request.session['unique_clicks_data'] = unique_clicks_data
        request.session['inst_lead_data'] = inst_lead_data
        request.session['inst_sale_data'] = inst_sale_data
        request.session['lead_sale'] = lead_sale

        return redirect('crreview')
```

crreview > services > interactive_graph.py

```
from crreview.services.kt_geo_data import KTGeoData


class TestKTGeoData:
    @classmethod
    def setup_class(cls):
        cls.geo_data = KTGeoData()

    def test_get_all_geo_name(self):
        all_geo_names = self.geo_data.get_all_geo_name()
        assert isinstance(all_geo_names, list)
        assert all(isinstance(name, str) for name in all_geo_names)

    def test_get_geo_code_by_geo_name(self):
        geo_name = 'United States'
        geo_code = self.geo_data.get_geo_code_by_geo_name(geo_name)
        assert isinstance(geo_code, str)
        assert geo_code == 'US'
```

crreview > services > kt_statistics.py

```
from wwldashboard.settings import keitaro_ktdash_api_key
from datetime import datetime, timedelta, date
from .kt_geo_data import KTGeoData
from loguru import logger
import requests


class KTStatistic:

    def __init__(self, social: str, offer: str, bundle_name: str, geo: str,
date_from: str, date_to: str):
        self.social = social
        self.offer = offer
        self.bundle_name = bundle_name
        self.geo = geo
        self.date_from = date_from
        self.date_to = date_to

    def get_statistic_by_metrix(self):
        date_from = datetime.strptime(self.date_from, '%Y-%m-%d').date()
        date_to = datetime.strptime(self.date_to, '%Y-%m-%d').date()

        unique_clicks_data = []
        inst_lead_data = []
        inst_sale_data = []
        lead_sale = []

        temp_date = date_from
        while temp_date != (date_to + timedelta(days=1)):
```

```
            statistic_from_kt = self._get_response(specific_day=temp_date)


unique_clicks_data.append(statistic_from_kt['campaign_unique_clicks'])
            try:
                result = (statistic_from_kt['leads'] +
statistic_from_kt['sales']) / statistic_from_kt[
                    'campaign_unique_clicks']
                inst_lead_data.append(result)
            except ZeroDivisionError:
                inst_lead_data.append(0)

            try:
                result = statistic_from_kt['sales'] /
statistic_from_kt['campaign_unique_clicks']
                inst_sale_data.append(result)
            except ZeroDivisionError:
                inst_sale_data.append(0)

            try:
                result = statistic_from_kt['sales'] /
(statistic_from_kt['leads'] + statistic_from_kt['sales'])
                lead_sale.append(result)
            except ZeroDivisionError:
                lead_sale.append(0)

            temp_date += timedelta(days=1)

        return unique_clicks_data, inst_lead_data, inst_sale_data,
lead_sale

    @staticmethod
    def _headers():
        headers = {
            'accept': 'application/json',
            'Api-Key': keitaro_ktdash_api_key,
            'Content-Type': 'application/json',
        }
        return headers

    def _json_data(self, specific_day: str):
        json_data = {
            'range': {
                'interval': 'custom_date_range',
                'timezone': 'Europe/Kyiv',
                'from': specific_day,
                'to': specific_day,
            },
            'columns': [],
            'metrics': [
                'campaign_unique_clicks',
                'leads',
                'sales',
```

```
            ],
            'grouping': [],
            'sort': [
                # {
                #     'name': 'country',
                #     'order': 'desc',
                # },
            ],
            'summary': True,
            'limit': None,
            'offset': 0,
        }

        if self.bundle_name == '':
            if self.geo == 'All geo' and self.offer == 'All offers':

self._update_all_geo_all_offers_without_bundle_name(json_data=json_data)

            elif self.geo != 'All geo' and self.offer == 'All offers':

self._update_specific_geo_with_all_offers_without_bundle_name(json_data=jso
n_data)

            elif self.geo != 'All geo' and self.offer != 'All offers':

self._update_specific_geo_with_specific_offer_without_bundle_name(json_data
=json_data)

            elif self.geo == 'All geo' and self.offer != 'All offers':

self._update_all_geo_specific_offer_without_bundle_name(json_data=json_data
)

        elif self.bundle_name != '':

            if self.geo == 'All geo' and self.offer == 'All offers':
                self._update_all_geo_all_offers(json_data=json_data)

            elif self.geo != 'All geo' and self.offer == 'All offers':

self._update_specific_geo_with_all_offers(json_data=json_data)

            elif self.geo != 'All geo' and self.offer != 'All offers':

self._update_specific_geo_with_specific_offer(json_data=json_data)

            elif self.geo == 'All geo' and self.offer != 'All offers':
                self._update_all_geo_specific_offer(json_data=json_data)
        return json_data

    def _update_all_geo_all_offers_without_bundle_name(self, json_data:
dict):
        if self.social == 'FB':
```

```
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social} without bundle name")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
```

```
                        ],
                    },
                ]
            })

    def _update_specific_geo_with_all_offers_without_bundle_name(self,
json_data: dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social} without bundle name")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
```

```
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })

    def _update_specific_geo_with_specific_offer_without_bundle_name(self,
json_data: dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social} without bundle name")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
```

```
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
```

```
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })

    def _update_all_geo_specific_offer_without_bundle_name(self, json_data:
dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social} without bundle name")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
```

```
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })

    def _update_all_geo_specific_offer(self, json_data: dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
```

```
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
```

```
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })

    def _update_all_geo_all_offers(self, json_data: dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
```

```
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    }, ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    }, ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    }, ]
            })

    def _update_specific_geo_with_all_offers(self, json_data: dict):
```

```
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
```

```
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    }, ]
            })

    def _update_specific_geo_with_specific_offer(self, json_data: dict):
        if self.social == 'FB':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
```

```
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'source',
                        'operator': 'CONTAINS',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'source',
                        'operator': 'NOT_CONTAIN',
                        'expression': 'acebook',
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
```

```
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })
        elif self.social == 'FB+UAC':
            logger.info(f"Collecting {self.offer} in {self.geo} by
{self.social}")
            json_data.update({
                'filters': [
                    {
                        'name': 'offer_id',
                        'operator': 'NOT_IN_LIST',
                        'expression': [
                            14,
                            17,
                            33,
                        ],
                    },
                    {
                        'name': 'sub_id_12',
                        'operator': 'EQUALS',
                        'expression': self.bundle_name,
                    },
                    {
                        'name': 'country',
                        'operator': 'EQUALS',
                        'expression':
self._get_geo_code_by_geo_name(self.geo),
                    },
                    {
                        'name': 'offer_id',
                        'operator': 'EQUALS',
                        'expression':
self._get_offer_id_by_offer_name(self.offer),
                    },
                ]
            })

    @staticmethod
    def get_all_offers():
        headers = {
            'Api-Key': keitaro_ktdash_api_key
        }

        response =
requests.get('https://wildwildktdash.com/admin_api/v1/offers',
headers=headers)

        return {offer['id']: offer['name'] for offer in response.json()}
```

```
    @staticmethod
    def _get_geo_code_by_geo_name(geo_name: str) -> str:
        kt_geo_data = KTGeoData()
        return kt_geo_data.get_geo_code_by_geo_name(geo_name=geo_name)

    def _get_offer_id_by_offer_name(self, offer_name: str) -> int:
        all_offers = self.get_all_offers().items()
        return next(offer_id for offer_id, offer_name_in_kt in all_offers
if offer_name_in_kt == offer_name)

    def _get_response(self, specific_day: date):
        url = 'https://domainkt.com/admin_api/v1/report/build'
        specific_day = specific_day.strftime('%Y-%m-%d')

        response = requests.post(
            url,
            headers=self._headers(),
            json=self._json_data(specific_day=specific_day)
        )

        return response.json()['summary']
```

gp_parser > views.py

```
from django.http import JsonResponse
from django.shortcuts import render
from django.views import View

from .services.readable_report import ReadableReport
from .utils.orm_utils import get_all_bundle_name,
get_last_update_data_in_db, get_all_actual_data_by_bundle_for_big_data
from django.core.serializers import serialize
import json

rr = ReadableReport()


class GPParserCriticalView(View):
    # critical
    def get(self, request):
        context = rr.get_all_app_reviews_or_ratings(option='all')
        last_update = get_last_update_data_in_db()
        context.update({
            'bundles': get_all_bundle_name(),
            'last_update': last_update
        })
        return render(request, 'critical.html', context=context)


class GPParserFilterRatingView(View):
    # filter-by-app-rating
    def get(self, request):
        last_update = get_last_update_data_in_db()
```

```
        bundle = request.GET.get('bundle')
        if bundle:
            context =
rr.get_filter_app_reviews_or_ratings(bundle_name=bundle, option='ratings')
        else:
            context = {}

        context.update({
            'selected_bundle': bundle,
            'bundles': get_all_bundle_name(),
            'last_update': last_update
        })
        return render(request, 'filter_app_rating.html', context=context)


class GPParserFilterReView(View):
    # filter-by-app-reviews
    def get(self, request):
        last_update = get_last_update_data_in_db()

        bundle = request.GET.get('bundle')
        if bundle:
            context =
rr.get_filter_app_reviews_or_ratings(bundle_name=bundle, option='reviews')
        else:
            context = {}

        context.update({
            'selected_bundle': bundle,
            'bundles': get_all_bundle_name(),
            'last_update': last_update
        })
        return render(request, 'filter_app_review.html', context=context)


class GPParserAllAppReView(View):
    # all-apps-reviews
    def get(self, request):
        # user_levels = get_user_level(request)
        last_update = get_last_update_data_in_db()
        context = rr.get_all_app_reviews_or_ratings(option='reviews')
        context.update({
            'last_update': last_update
        })
        return render(request, 'all_app_review.html', context=context)


class GPParserAllAppRatingView(View):
    # all-apps-rating
    def get(self, request):
        last_update = get_last_update_data_in_db()
        context = rr.get_all_app_reviews_or_ratings(option='ratings')
```

```
        context.update({
            'last_update': last_update
        })
        return render(request, 'all_app_rating.html', context=context)


def get_gp_parser_actual_data(request):
    # get_gp_parser_actual_data?bundle_id=com.lightdev.hoppinghead
    if request.method == 'GET':
        bundle_id = request.GET.get('bundle_id')
        date_from = request.GET.get('date_from')
        date_to = request.GET.get('date_to')
        bundle_data = get_all_actual_data_by_bundle_for_big_data(bundle_id)
        return JsonResponse({'result': bundle_data})
```

gp_parser > views.py

```
from .utils.orm_utils import change_actual_from_bundle_data,
insert_into_bundle_data, \
    get_all_bundles

from .services.traffic_report import TrafficReport
from .services.gp_parser import GPParser

from celery import shared_task
from loguru import logger


@shared_task
def fill_data_to_db():
    logger.info('Start task fill_data_to_db')
    tr = TrafficReport()
    bundles = get_all_bundles()
    filling_data = []

    for bundle in bundles:
        result = tr.get_geo_and_lng_for_bundle(bundle.bundle_id,
in_list=True)
        _filling_data(filling_data, bundle, result)

    for bundle in bundles:
        result = tr.get_geo_and_lng_for_bundle(bundle.bundle_id,
unique_clicks=600)
        _filling_data(filling_data, bundle, result)

    change_actual_from_bundle_data()
    for data in filling_data:
        insert_into_bundle_data(
            bundle=data['bundle'],
            bundle_id=data['bundle_id'],
            geo=data['geo'],
            language=data['language'],
            stars=data['stars'],
```

```
            count_bad_reviews=data['count_bad_reviews']
        )
    logger.info('End task fill_data_to_db')


def _filling_data(filling_data, bundle, result):
    for bgl in result:
        geo = bgl['country_code']
        language = bgl['language_code']
        gplay_parser = GPParser(app_id=bundle.bundle_id, geo=geo,
language=language)

        reviews = gplay_parser.get_app_reviews_parse()
        if reviews == 'ban':
            logger.warning(f'App {bundle.bundle_id} is banned in {geo} ->
{language}; continue loop')
            continue
        elif reviews:
            count_bad_reviews = len([star for star in reviews if star <=
3])
        else:
            count_bad_reviews = None

        stars = gplay_parser.get_app_stars()
        filling_data.append({
            'bundle_id': bundle.bundle_id,
            'bundle': bundle.bundle_name,
            'geo': geo,
            'language': language,
            'stars': stars,
            'count_bad_reviews': count_bad_reviews
        })
```

gp_parser > models.py

```
from django.db import models


class Language(models.Model):
    lng_kt = models.CharField(max_length=50)
    lng_code = models.CharField(max_length=50)

    def __str__(self):
        return self.lng_code


class Geo(models.Model):
    geo_code = models.CharField(max_length=5)
    geo_code_kt = models.CharField(max_length=5)

    def __str__(self):
        return self.geo_code
```

```
class Bundle(models.Model):
    bundle_name = models.CharField(max_length=100)
    bundle_id = models.CharField(max_length=100)

    def __str__(self):
        return self.bundle_id


class BundleData(models.Model):
    date_time = models.DateTimeField()
    bundle_name = models.CharField(max_length=100)
    bundle_id = models.CharField(max_length=100, null=True)
    geo = models.CharField(max_length=5)
    lang = models.CharField(max_length=50, null=True)
    rating_score = models.FloatField(null=True)
    negative_reviews = models.PositiveIntegerField(null=True)
    actual = models.BooleanField()

    def __str__(self):
        return f'{self.bundle_name}_{self.rating_score}'
```

gp_parser > utils > orm_utils.py

```
# from django.utils.timezone import make_aware
from django.utils import timezone
from ..models import Language, Bundle, BundleData
from datetime import datetime, timedelta

from loguru import logger


def get_all_bundle_name():

    bundles = Bundle.objects.all()
    return [bundle.bundle_name for bundle in bundles]


def get_all_bundles():

    return Bundle.objects.all()


def get_bundle_id_by_bundle_name(bundle_name: str):

    bundle = Bundle.objects.filter(bundle_name=bundle_name).first()
    if bundle:
        return bundle.bundle_id
    logger.warning(f'Не удалось получить id по бандлу: {bundle_name}')
    return None


def get_lng_code_from_lng_name(lng_name: str):
```

```
    language = Language.objects.filter(lng_kt=lng_name).first()
    if not language:
        logger.warning(f'Не удалось получить lng_code по языку:
{lng_name}')
        return lng_name
    return language.lng_code


def change_actual_from_bundle_data():

    BundleData.objects.filter(actual=True).update(actual=False)


def insert_into_bundle_data(bundle, bundle_id, geo, language, stars,
count_bad_reviews):

    BundleData.objects.create(
        date_time=timezone.now() + timezone.timedelta(hours=3),
        bundle_name=bundle,
        bundle_id=bundle_id,
        geo=geo,
        lang=language,
        rating_score=stars,
        negative_reviews=count_bad_reviews,
        actual=True
    )


def get_last_update_data_in_db():

    query = BundleData.objects.filter(actual=True).first()
    if query:
        return query.date_time
    return 'Нет данных'


def get_all_score_by_language_in_all_geo(language: str):

    query = BundleData.objects.filter(lang=language, actual=True)
    result = []
    for bundle in query:
        if bundle.rating_score is not None:
            result.append(bundle.rating_score)
        else:
            result.append(0)
    return result


def get_all_actual_data_by_bundle(bundle_name: str = None):

    all_actual_bundles = [bundle.bundle_name for bundle in
get_all_bundles()]
```

```
    if bundle_name:
        bundles_data = BundleData.objects.filter(actual=True,
bundle_name=bundle_name)
    else:
        bundles_data = BundleData.objects.filter(actual=True)

    actual_bundles = [bundle for bundle in bundles_data if
bundle.bundle_name in all_actual_bundles]
    return actual_bundles


# for big data collector
def get_all_actual_data_by_bundle_for_big_data(bundle_id: str, date_from:
str, date_to: str):

    bundle_data = BundleData.objects.filter(
        actual=True,
        bundle_id=bundle_id,
        date_time__range=(
            datetime.strptime(date_from, '%Y-%m-%d').date(),
            datetime.strptime(date_to, '%Y-%m-%d').date() +
timedelta(days=1)
        ),
    ).values()
    return list(bundle_data)
```

gp_parser > utils > element_spec.py

```
class ElementSpec:
    def __init__(self, review: list):
        self.review = review

    def get_value_by_indexes(self, indexes: list):
        return self._extract(self.review, indexes)

    def _extract(self, source: list, indexes: list):
        try:
            if len(indexes) == 1:
                return source[indexes[0]]
        except IndexError:
            return False
        return self._extract(source[indexes[0]], indexes[1::])
```

gp_parser > services > gp_parser.py

```
from ..utils.element_spec import ElementSpec
# from weblink.gp_parser.utils.element_spec import ElementSpec
from bs4 import BeautifulSoup
from datetime import datetime
from loguru import logger
```

```
import requests
import time
import json
import re


class GPParser:
    REVIEWS = re.compile(r"\)]}'\n\n([\s\S]+)")
    _proxies = {
        # 'http': 'http://MWT8KycW:McZGxQjZ@45.146.170.172:64084',
        'https': 'http://MWT8KycW:McZGxQjZ@45.146.170.172:64084',
    }

    def __init__(self, app_id: str, geo: str, language: str):
        self.app_id = app_id
        self.geo = geo
        self.language = language
        self.url =
f'https://play.google.com/store/apps/details?id={self.app_id}&gl={self.geo}
&hl={self.language}'
        self._soup = self._get_bundle_request()
        self._count_parse_reviews = 0

    def _get_bundle_request(self) -> BeautifulSoup | None:
        """
        Метод для запроса к id приложения в Play Market
        """
        headers = {
            'authority': 'play.google.com',
            'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
p,image/apng,'
                         '*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
            'accept-encoding': 'gzip, deflate, br',
            'accept-language': 'ru-UA,ru;q=0.9,en-US;q=0.8,en;q=0.7,uk-
UA;q=0.6,uk;q=0.5,ru-RU;q=0.4',
            'cache-control': 'max-age=0',
            'referer': 'https://play.google.com/',
            'sec-ch-ua': '"Not?A_Brand";v="8", "Chromium";v="108", "Google
Chrome";v="108"',
            'sec-ch-ua-arch': '"x86"',
            'sec-ch-ua-bitness': '"64"',
            'sec-ch-ua-full-version': '"108.0.5359.125"',
            'sec-ch-ua-full-version-list': 'Not?A_Brand";v="8.0.0.0",
"Chromium";v="108.0.5359.125", "Google '
                                          'Chrome";v="108.0.5359.125"',
            'sec-ch-ua-mobile': '?0',
            'sec-ch-ua-model': '""',
            'sec-ch-ua-platform': '"Windows"',
            'sec-ch-ua-platform-version': '"15.0.0"',
            'sec-ch-ua-wow64': '?0',
            'sec-fetch-dest': 'document',
            'sec-fetch-mode': 'navigate',
```

```
            'sec-fetch-site': 'same-origin',
            'sec-fetch-user': '?1',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) '
                          'Chrome/107.0.0.0 Safari/537.36',
        }

        for _ in range(6):
            try:
                response = requests.get(self.url, headers=headers,
proxies=self._proxies)
                if response.status_code == 200:
                    return BeautifulSoup(response.text, 'html.parser')
            except (ConnectionRefusedError, requests.exceptions.ProxyError,
requests.exceptions.ConnectTimeout):
                logger.warning("Ожидание GET")

                time.sleep(2)
        return None

    def _get_payload_data_for_reviews(self, count: int, stars: int = None,
pagination_token: str = None) -> bytes:
        """
        Метод формирования полезной нагрузки для получения отзывов
        """
        payload_for_first_page =
"f.req=%5B%5B%5B%22UsvDTd%22%2C%22%5Bnull%2Cnull%2C%5B2%2C{sort}%2C%5B{" \

"count}%2Cnull%2Cnull%5D%2Cnull%2C%5Bnull%2C{stars}%5D%5D%2C%5B%5C%22{app_i
d}%5C%22%" \

"2C7%5D%5D%22%2Cnull%2C%22generic%22%5D%5D%5D"
        payload_for_paginated_page =
"f.req=%5B%5B%5B%22UsvDTd%22%2C%22%5Bnull%2Cnull%2C%5B2%2C{sort}%2C%5B{" \

"count}%2Cnull%2C%5C%22{pagination_token}%5C%22%5D%2Cnull%2C%5Bnull%2C{" \

"stars}%5D%5D%2C%5B%5C%22{app_id}%5C%22%2C7%5D%5D%22%2Cnull%2C%22generic%22
%5D" \
                                      "%5D%5D"
        if pagination_token is not None:
            result = payload_for_paginated_page.format(
                app_id=self.app_id,
                sort=2,  # новые записи, 1 - по релевантности
                count=count,
                stars=stars,
                pagination_token=pagination_token,
            )
        else:
            result = payload_for_first_page.format(app_id=self.app_id,
sort=2, count=count, stars=stars)
        return result.encode()
```

```
    def _get_review_items_and_token(self, url: str, data: bytes) ->
list[list, str]:
        """Статическая функция для получения отзывов и токена

        :return: Десериализованный список данных с отзывами и токен

        """
        for _ in range(6):
            try:
                response = requests.post(url, data=data,
proxies=self._proxies,
                                         headers={"content-type":
"application/x-www-form-urlencoded"})
                if response.status_code == 200:
                    matches =
json.loads(self.REVIEWS.findall(response.text)[0])
                    reviews = json.loads(matches[0][2])[0]
                    token = json.loads(matches[0][2])[-1][-1]
                    return [reviews, token]
                elif response.status_code in [400, 404]:
                    raise TypeError
            except (ConnectionRefusedError, requests.exceptions.ProxyError,
requests.exceptions.ConnectTimeout):
                logger.warning("Ожидание POST")
                time.sleep(30)
        raise TypeError

    def get_app_stars(self) -> float | None:
        """Метод для получения оценки приложения в Play Market"""
        if self._soup:
            stars = self._soup.find('div', {'itemprop': 'starRating'})
            if stars is not None:
                try:
                    return float(stars.text.replace("star",
"").replace(',', '.'))
                except ValueError:  # todo убрать этот try
                    return None
            else:
                return None
        else:
            return None

    def get_app_reviews(self, count: int = 30, stars: int = None) ->
list[dict]:
        """
        Метод для получения отзывов
        """
        url =
f"https://play.google.com/_/PlayStoreUi/data/batchexecute?&gl={self.geo.low
er()}&hl={self.language}"
        # logger.info(f"Сформированная ссылка: {url}")
        # logger.info(f"Ссылка для парсинга: {self.url}")
```

```
        result = []
        token = None
        self._count_parse_reviews = count

        while True:
            if self._count_parse_reviews == 0:
                break
            data = self._get_payload_data_for_reviews(
                count=self._count_parse_reviews,
                stars="null" if stars is None else stars,
                pagination_token=token
            )
            try:
                items, token = self._get_review_items_and_token(url, data)

            except (TypeError, IndexError):
                # если закончились отзывы или нет отзывов
                break

            for review in items:
                es = ElementSpec(review)
                review_id = es.get_value_by_indexes([0])
                user_name = es.get_value_by_indexes([1, 0])
                user_image = es.get_value_by_indexes([1, 1, 3, 2])
                comment = es.get_value_by_indexes([4])
                stars = es.get_value_by_indexes([2])
                thumbs_up_count = es.get_value_by_indexes([6])
                review_created_version = es.get_value_by_indexes([10])
                at = datetime.fromtimestamp(es.get_value_by_indexes([5,
0]))
                result.append({"app_id": self.app_id, "review_id":
review_id, "user_name": user_name,
                               "user_image": user_image, "comment":
comment, "stars": stars, "at": at,
                               "thumbs_up_count": thumbs_up_count,
"review_created_version": review_created_version})
            self._count_parse_reviews = count - len(result)
        return result

    def get_app_reviews_parse(self):
        if self._soup:
            raw_result = self._soup.find_all('div', {'data-g-id':
'reviews'})
            raw_find = re.findall(r'</div><div class=".*?"><div aria-
label="(.*?)"', str(raw_result))
            stars = []
            for string in raw_find:
                count_min_stars = []
                for letter in string:
                    if letter.isdigit():
                        count_min_stars.append(int(letter))
                if len(count_min_stars) > 0:
                    stars.append(min(count_min_stars))
```

```
            # logger.debug(f'Приложение {self.app_id} имеет оценки: {stars}
по {self.geo}, {self.language}')
            logger.debug(self.url)
            return stars
        return 'ban'
```

gp_parser > services > readable_report.py

```
from typing import Literal

from ..utils.orm_utils import (
    get_all_bundle_name, get_bundle_id_by_bundle_name,
    get_all_actual_data_by_bundle
)
import pandas as pd


class ReadableReport:

    def get_filter_app_reviews_or_ratings(
            self, bundle_name: str,
            option: Literal["reviews", "ratings"] = "reviews") -> dict:
        """
        Возвращает контекст с данными для отображения в таблице в шаблоне
        """
        actual_data = get_all_actual_data_by_bundle(bundle_name)
        geo_codes, language_codes = self._get_all_geo_and_lng(actual_data,
delete_duplicates=True)
        df = pd.DataFrame(columns=geo_codes, index=language_codes)
        for bundle in actual_data:
            bundle_id = get_bundle_id_by_bundle_name(bundle.bundle_name)
            if bundle.lang != 'Неизвестно':
                url =
f'https://play.google.com/store/apps/details?id={bundle_id}&gl={bundle.geo}
&hl={bundle.lang}'
            else:
                url =
f'https://play.google.com/store/apps/details?id={bundle_id}&gl={bundle.geo}
'

            if option == 'reviews':
                if bundle.negative_reviews or bundle.negative_reviews == 0:
                    df.loc[bundle.lang, bundle.geo] =
[bundle.negative_reviews, url]
            elif option == 'ratings':
                if bundle.rating_score:
                    df.loc[bundle.lang, bundle.geo] = [bundle.rating_score,
url]

        df = df.dropna(axis=0, how='all')  # по оси x
        df = df.dropna(axis=1, how='all')  # по оси y
        df = df.fillna('')
        tuples_for_table = df.to_records()
```

```
        return {'tuples_for_table': tuples_for_table, 'geos': df.columns}

    def get_all_app_reviews_or_ratings(self, option: Literal['reviews',
'ratings', 'all']) -> dict:
        """
        Возвращает контекст с данными для отображения в таблице в шаблоне
        """
        bundles = get_all_bundle_name()
        actual_data = get_all_actual_data_by_bundle()
        geo_codes, language_codes = self._get_all_geo_and_lng(actual_data)
        multi_index = pd.MultiIndex.from_arrays([geo_codes,
language_codes]).drop_duplicates()
        if option == 'all':
            new_bundles = []
            for bundle in bundles:
                new_bundles.append(bundle + ' (rating)')
                new_bundles.append(bundle + ' (reviews)')
            df = pd.DataFrame(columns=new_bundles, index=multi_index)
        else:
            df = pd.DataFrame(columns=bundles, index=multi_index)

        for bundle in actual_data:
            bundle_id = get_bundle_id_by_bundle_name(bundle.bundle_name)
            if bundle.lang != 'Неизвестно':
                url =
f'https://play.google.com/store/apps/details?id={bundle_id}&gl={bundle.geo}
&hl={bundle.lang}'
            else:
                url =
f'https://play.google.com/store/apps/details?id={bundle_id}&gl={bundle.geo}
'

            if option == 'reviews':
                if bundle.negative_reviews or bundle.negative_reviews == 0:
                    df.loc[(bundle.geo, bundle.lang), bundle.bundle_name] =
[bundle.negative_reviews, url]
            elif option == 'ratings':
                if bundle.rating_score:
                    df.loc[(bundle.geo, bundle.lang), bundle.bundle_name] =
[bundle.rating_score, url]
            elif option == 'all':
                if bundle.rating_score and bundle.rating_score <= 4.4:
                    df.loc[(bundle.geo, bundle.lang), bundle.bundle_name +
' (rating)'] = [bundle.rating_score, url]
                if bundle.negative_reviews and bundle.negative_reviews > 0:
                    df.loc[(bundle.geo, bundle.lang), bundle.bundle_name +
' (reviews)'] = [bundle.negative_reviews, url]

        df = df.dropna(axis=0, how='all')  # по оси x
        # df = df.dropna(axis=1, how='all')  # по оси y
        df = df.fillna('')
        new_data_for_table = df.to_records()
```

```
        if option == 'all':
            new_data_for_table =
self._colorize_for_table_critical(new_data_for_table)

        headers = [['Rating', 'Reviews']] * len(bundles)
        context = {'tuples_for_table': new_data_for_table, 'bundles':
bundles, 'headers': headers}
        return context

    @staticmethod
    def _colorize_for_table_critical(tuples_for_table) -> list:
        new_data_for_table = []
        for data in tuples_for_table:
            temp = []
            for count, cell in enumerate(data, start=0):
                if count in [0, 1]:
                    temp.append(f'<td>{cell}</td>')

                elif count % 2 == 0:
                    # это рейтинг
                    if isinstance(cell, list):
                        url = cell[1]
                        cell = cell[0]
                        if cell:
                            if 4.1 < cell <= 4.4:
                                temp.append(f'<td class="orange-
background"><a style="color:black;" href="{url}"
target="_blank">{cell}</a></td>')
                            elif cell <= 4.1:
                                temp.append(f'<td class="red-background"><a
style="color:black;" href="{url}" target="_blank">{cell}</a></td>')
                        else:
                            temp.append(f'<td>{cell}</td>')
                    else:
                        temp.append(f'<td>{cell}</td>')
                else:
                    # это отзывы
                    if isinstance(cell, list):
                        url = cell[1]
                        cell = cell[0]
                        if cell:
                            if cell == 1:
                                temp.append(f'<td class="orange-
background"><a style="color:black;" href="{url}"
target="_blank">{cell}</a></td>')
                            else:
                                temp.append(f'<td class="red-background"><a
style="color:black;" href="{url}" target="_blank">{cell}</a></td>')
                        else:
                            temp.append(f'<td>{cell}</td>')
                    else:
                        temp.append(f'<td>{cell}</td>')
```

```
            new_data_for_table.append(temp.copy())
        return new_data_for_table

    @staticmethod
    def _get_all_geo_and_lng(actual_data, delete_duplicates: bool = False)
-> tuple:
        geos = []
        languages = []
        for bundle in actual_data:
            if delete_duplicates:
                if bundle.geo not in geos:
                    geos.append(bundle.geo)
                if bundle.lang not in languages:
                    languages.append(bundle.lang)
            else:
                geos.append(bundle.geo)
                languages.append(bundle.lang)

        return geos, languages
```

gp_parser > services > traffic_report.py

```
from pprint import pprint

from django.conf import settings

from ..utils.orm_utils import get_lng_code_from_lng_name

from datetime import datetime, timedelta

import requests


class TrafficReport:

    def __init__(self, date_from: datetime = datetime.now() -
timedelta(days=7), date_to: datetime = datetime.now()):
        self.date_from = date_from
        self.date_to = date_to

    def get_geo_and_lng_for_bundle(self, bundle: str, in_list: bool =
False, unique_clicks: int = 600) -> list:
        date_from = self.date_from.strftime('%Y-%m-%d')
        date_to = self.date_to.strftime('%Y-%m-%d')

        headers = {
            'accept': 'application/json',
            'Api-Key': settings.WWL_TRK_API_KEY_AUDM,
            'Content-Type': 'application/json',
        }
        params = {
            'object': 'reports.build',
        }
```

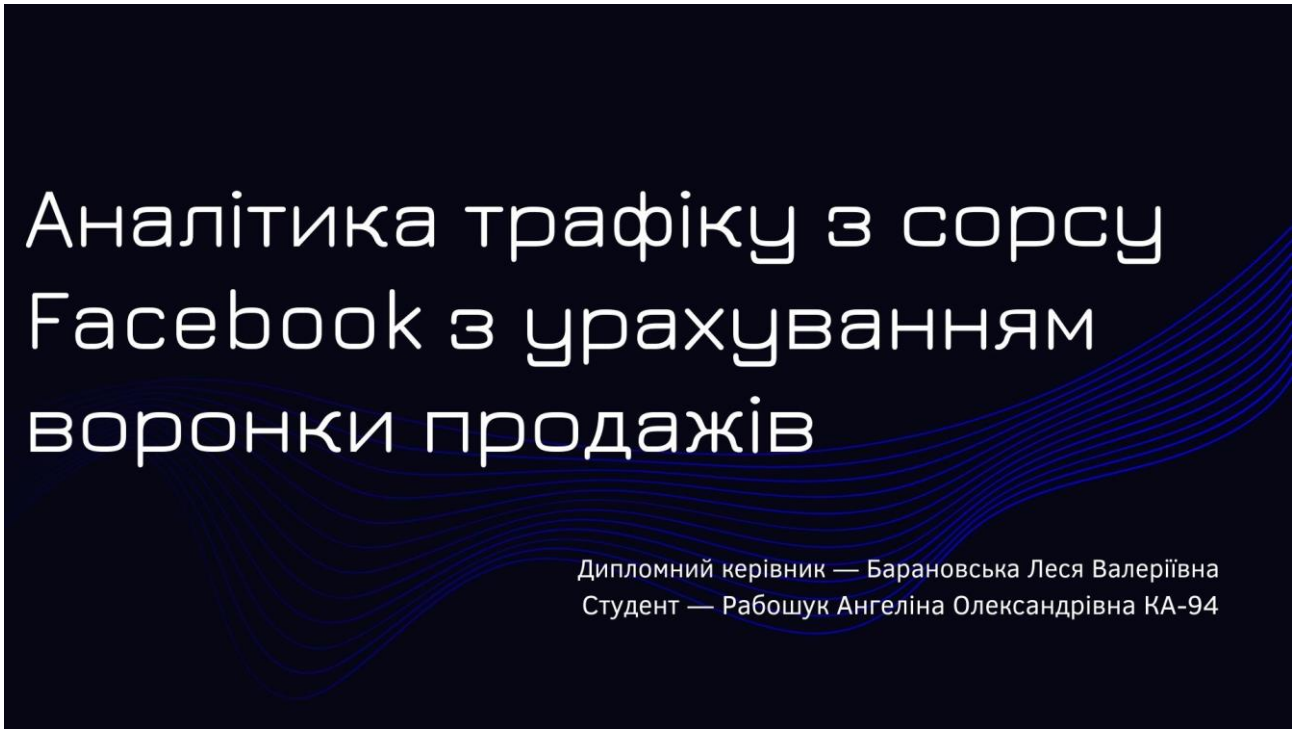
```
json_data = {
    'range': {
        'interval': 'custom_date_range',
        'timezone': 'Europe/Kyiv',
        'from': f'{date_from} 00:00',
        'to': f'{date_to} 23:59',
    },
    'columns': [],
    'metrics': [
        'campaign_unique_clicks',
    ],
    'grouping': [
        'country',
        'language',
    ],
    'filters': [
        {
            'name': 'sub_id_12',
            'operator': 'EQUALS',
            'expression': bundle,
        },
        {
            'name': 'country',
            'operator': 'IN_LIST' if in_list else 'NOT_IN_LIST',
            'expression': [
                'AT',
                'AE',
                'CA',
                'DK',
                'FI',
                'IE',
                'GL',
                'NZ',
                'NO',
                'CH',
                'AU',
                'BE',
                'FR',
                'DE',
                'NL',
                'GB',
            ],
        },
    ],
    'sort': [
        {
            'name': 'campaign_unique_clicks',
            'order': 'desc',
        },
    ],
    'summary': True,
    'limit': None,
```

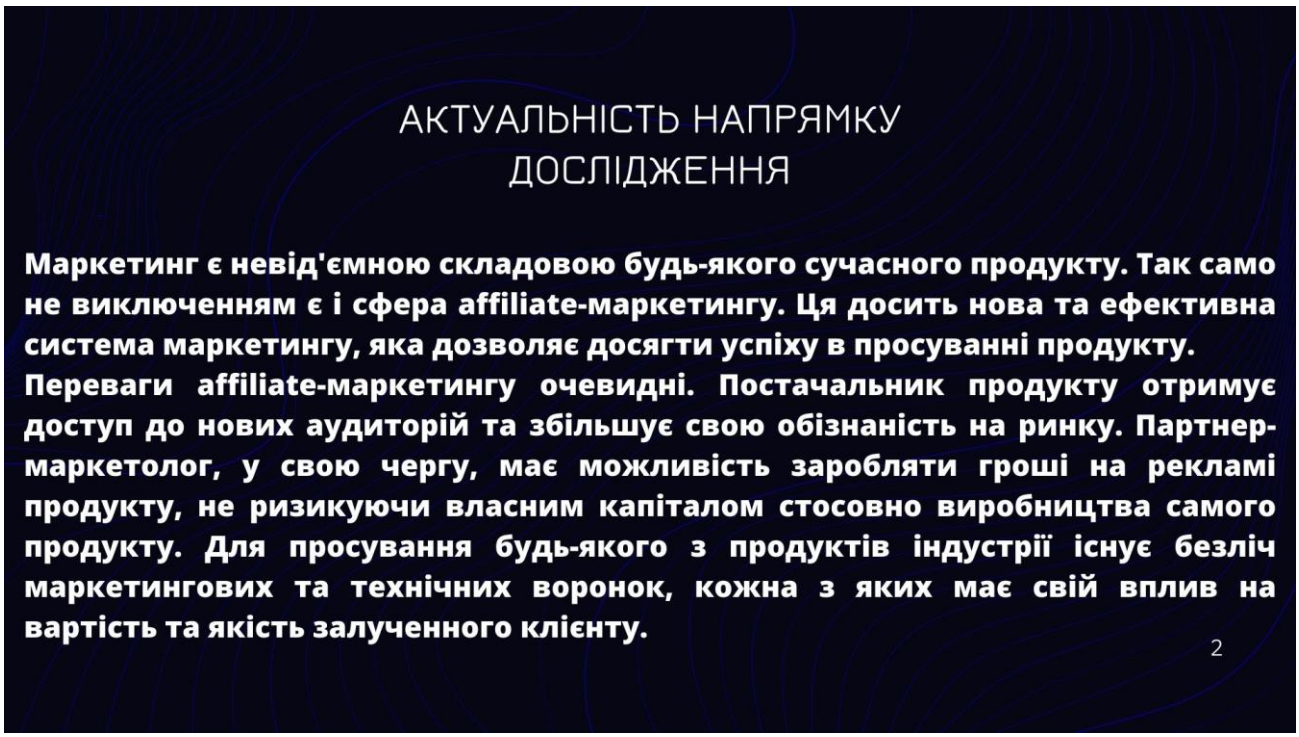
```
            'offset': 0,
        }
        response = requests.post(
            'https://domainkt.com/admin_api/v1/report/build',
            headers=headers,
            json=json_data,
        )
        print(response.text)

        if not in_list:
            rows_actual_data = []
            for row in response.json()['rows']:
                if row['campaign_unique_clicks'] >= unique_clicks:
                    language_code =
get_lng_code_from_lng_name(row['language'])
                    row.update({'language_code': language_code})
                    rows_actual_data.append(row)
        else:
            rows_actual_data = []
            for row in response.json()['rows']:
                language_code = get_lng_code_from_lng_name(row['language'])
                row.update({'language_code': language_code})
                rows_actual_data.append(row)
        return rows_actual_data
```

ДОДАТОК Б

ПРЕЗЕНТАЦІЯ





ПОСТАНОВКА ЗАДАЧІ

- **Мета - побудова і автоматизація технічної воронки для реалізації іздешевлення трафіку для просування онлайн казино.**
- **Об'єкт дослідження - застосування та автоматизація нейронних моделей для відслідковування якості трафіку і зниження витрат.**
- **Предмет дослідження - технічна воронка продажу з використанням штучного інтелекту. Побудова маршруту трафіку.**

3

ПОСТАНОВКА ЗАДАЧІ

Для досягнення цієї мети необхідно розв’язати наступні задачі:

- **Будування оптимальної системи проходження трафіку з мінімальною ціною і максимальною якістю для онлайн казино.**
- **Аналіз основних інфраструктурних елементів технічної воронки.**
- **Принцип аналізу - порівняння фінансових і статистичних показників на основі даних в TDS та даних від рекламодавців.**

4

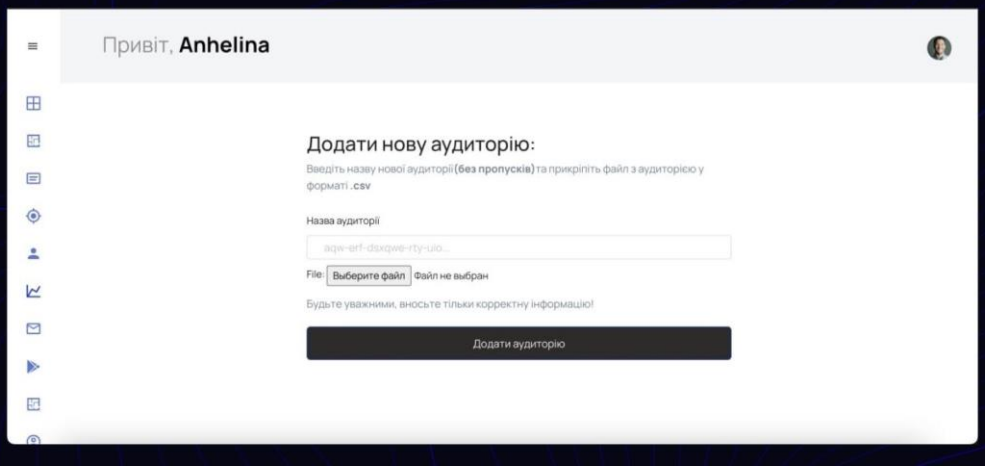
ПОСТАНОВКА ЗАДАЧІ

- **Аналіз та розробка технічних модернізацій з використанням автоматизацій та нейромереж для оптимізації роботи трафік системи.**
- **Розробка аналітичного інструменту і порівняння вихідних статистичних даних отриманих в результаті роботи.**
- **Визначити нові напрямки оптимізації, автоматизації та суміжних проектів/продуктів, визначених під час дослідження та розробки.**

5

БАЗА РОЗРОБКИ

1. **Для розробки обрана сукупність python + Django**
2. **В рамках збереження даних - база даних PostrgreSQL**
3. **UI реалізований на базі: HTML5, CSS3, bootstrap 4 та 5**



6

ІНСТРУМЕНТ ГРАФІЧНОЇ АНАЛІТИКИ

Для візуального аналізу ефективності та порівняння якості роботи воронки продажу - розроблено мікросервіс, який отримує дані по API та на їх основі будує графіки отриманих та розрахункових показників.

7

ШВИДКА ЗМІНА ОБСЯГУ ТРАФІКУ

Графічним методом дослідження виділено залежність різкого зростання об'ємів трафіку, спрямованого на 1 додаток, та спадання рівня конверсіонності пропорційно до об'ємів;

На базі отриманої інформації виділене перше покращення: "Для збільшення та стабілізації рівня конверту треба збільшити кількість додатків та рівномірно розподілити їх серед клієнтів. Ріст показників при використанні інформації >= 20%"

8

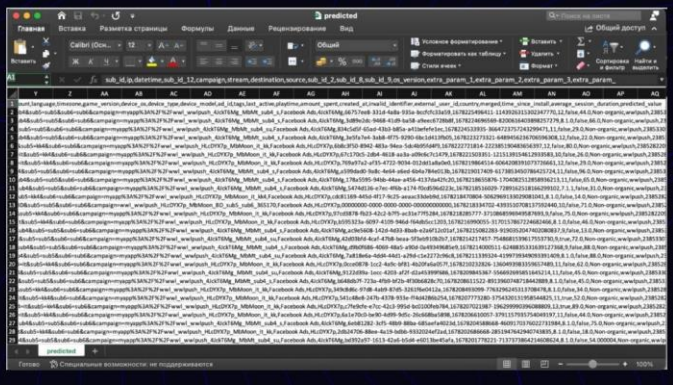
КЛАСТЕРІЗАЦІЯ FACEBOOK

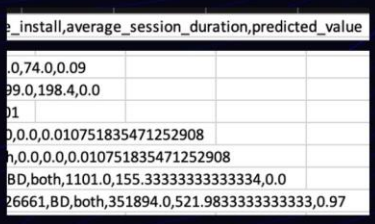
При запуску рекламної компанії Facebook може або давати зовсім рандомну аудиторію з будь-якого підкластера, загального кластеру аудиторії, зазначеного налаштуваннями рекламного кабінету, або саме той підкластер, у якому знаходиться саме наша цільова аудиторія.
Так як Facebook вміє мапити отримані події і кластеризувати користувачів за подіями, то зі сторони баїнгу потрібно виділити та надати Facebook-у максимально якісну аудиторію.

9

ОБРОБКА ТА АНАЛІЗ АУДИТОРІЇ

1. **Для аналізу поєднали дані з 2 сервісів: OneSignal та Keitaro**
2. **Отриманий DataFrame проаналізували за допомогою RandomForestRegressor**
3. **Для кожного юзера у комбінації додаток + країна отримали ймовірність повторного депозиту на основі поведінкових чинників**

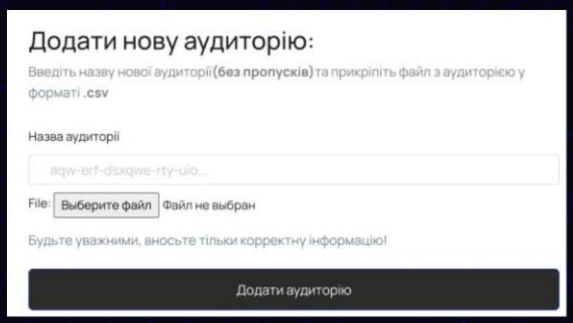


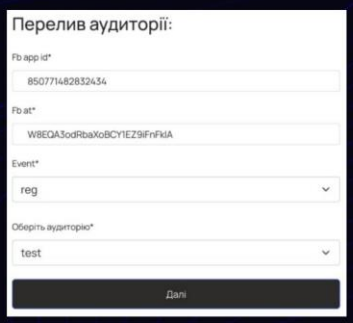


10

ПЕРЕЛИВ АУДИТОРІЇ – МЕХАНІКА

Отримана аудиторія зберігається у вигляді gadid в .csv файлі, в якому виділяються вже юзери з потрібною ймовірістю повтороного депозиту та підвантажуються.
Система, отримуючи всі необхідні дані про додаток у Facebook та файл з аудиторією, реалізує покроковий відстрій postback про кожну імітовану подію на проміжний сервер.





11

GOOGLE PLAY PARSER

Наявність негативних рейтингів та відгуків = втрата великої кількості гравців. Тож розроблений продукт - система моніторингу та вчасного сповіщення відділу роботи з додатками для покращення показників та зменьшення кількості вузьких горняток у воронці.

12

ПІДСУМОК ПОТЕНЦІЙНОГО ЗРОСТАННЯ ПОКАЗНИКІВ

Вплив попередніх досліджень та інструментів у цифрах:

1. **Коливання об'ємів - до 20%**
2. **Перелив аудиторії - до 20% на конверт та до 15% на виплату за кожного гравця**
3. **Google Play Parser - від 3% до 20%, адже рейтинги та відгуки можуть суттєво відрізнятися**

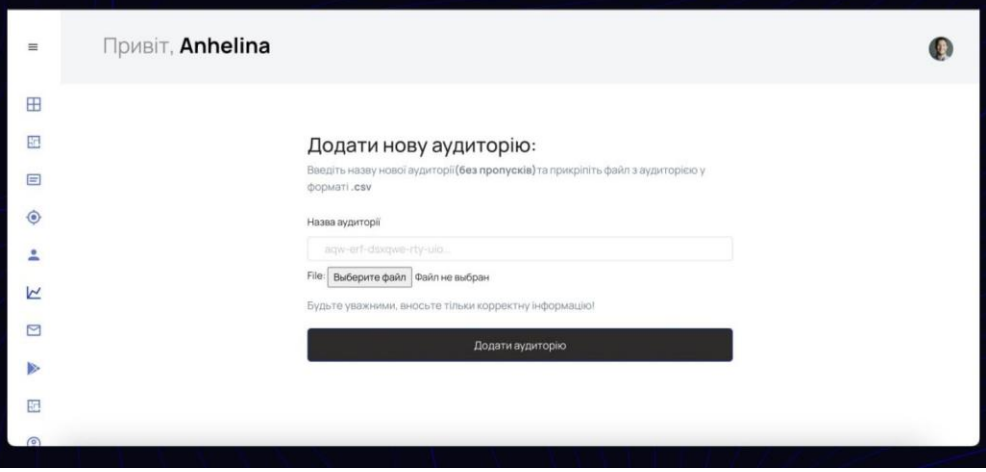
Інструменти та вплив на процеси:

1. **Google Play Parser - реалізація вчасного сповіщення про проблему.**
2. **CR Review - інструмент загального моніторингу якості роботи воронки.**

13

РЕЗУЛЬТАТИ РОБОТИ

Створено технічну воронку продажу на основі використання штучного інтелекту, що здатна відслідковувати якість трафіку і зниження витрат клієнтів.



14

ВИСНОВКИ

Проведено аналіз різних методологій реклами з точки зору контенту та технічної складової.

• Виявлено, що існує значна кількість таких методологій, що надають можливість ефективно комунікувати з аудиторією.

• Facebook є найбільш ефективним та цікавим інструментом реклами.

• На основі цього було реалізовано технічну воронку, яка повністю замінює спілкування з безліччю рекламодавців про надання баз, які вони не згодні надавати навіть під NDA, що надало безперечну перевагу серед інших компаній, які розробляють мобільні додатки.

• Було вирішено проблему можливісті збору не тільки загальної аудиторії, а найбажанішої рекламодавцем, пересилання цієї аудиторії на новий додаток, завдяки цьому отримання великої конкурентної переваги. Якість трафіку через ці додатки стала значно більшою і конверт кращим.

15

• Побудована в роботі технічна воронка є ідеальною для продажів на платформі Facebook.

ДЯКУЮ ЗА УВАГУ

РАБОШУК АНГЕЛІНА