Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.492

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: "Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень"

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА
ВКБР.ІПЗ - 22.00.00.000 ІПЗ

Студент

ІПЗ-42 _____ /Владислав ГРОМЕНКО/

Науковий керівник

д.т.н, проф._____ /Віктор ШЕВЧЕНКО/

Консультант
з питань нормоконтролю

_____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри

д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

Рішенням Екзаменаційної комісії

випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії

професор, доктор техн. наук Вишнівський В.В

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО:

Завідувач кафедри програмних систем і технологій _____ (Олексій БИЧКОВ)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Громенко Владислав Віталійович_____

(прізвище, ім’я, по батькові)

1. **Тема випускної кваліфікаційної бакалаврської роботи** **Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень.**

керівник роботи Шевченко В.Л. д.т.н., професор_____

затверджена наказом вищого навчального закладу від „ 11 ”листопада 2020р. № 6

2. **Строк подання студентом роботи** 30.05.2021 р

3. **Вихідні дані до роботи** моделі, засновані на генетичних алгоритмах, вбудовані у програмне забезпечення для реклами, теоретичні концепції моделей для аналізу часових рядів _____

4. **Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)**

1. Аналіз наявних даних, для виявлення їх особливостей._____

2. Створення моделі для прогнозування часових рядів._____

3. Створення алгоритму для створення рекламного плану.

4. Реалізація ПЗ, яке реалізує попередні описані моделі та алгоритми.

5. **Перелік графічного матеріалу (з точним забезпеченням обов'язкових креслень)**

1. Графік автокореляції часового ряду (рис. 2.1. ст. 38)
2. Графік роботи моделі ARMA (рис 2.5, ст 43)
3. Схема роботи генетичного алгоритму (рис 2.10, ст. 53)
4. ER діаграма схеми даних БД (рис.3.2., ст. 58)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1. Постановка задачі автоматизації управління рекламною кампанією. Аналіз наявних даних та рішень.	Віктор ШЕВЧЕНКО	22.01.2021	22.01.2021
Розділ 2. Розробка математичної моделі для прогнозування часового ряду та методу прийняття рішень	Віктор ШЕВЧЕНКО	22.01.2021	22.01.2021
Розділ 3. Реалізація програмного забезпечення для автоматизції управління рекламною кампанією.	Віктор ШЕВЧЕНКО	22.01.2021	22.01.2021

7. **Дата видачі завдання** 13 жовтня 2020 р.

Керівник _____/Віктор ШЕВЧЕНКО

Завдання прийняв до виконання _____/Владислав ГРОМЕНКО

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Опис та формалізація задачі	17.10.2020 – 10.11.2020	Виконано
2	Вивчення існуючих методів	11.11.2020 – 15.12.2020	Виконано
3	Аналіз наявних даних	16.12.2020 – 10.01.2021	Виконано
4	Аналіз моделей часових рядів	11.01.2021 – 20.02.2021	Виконано
5	Проектування моделі для прогнозування часових рядів	21.02.2021 – 09.03.2021	Виконано
6	Проектування моделі прийняття рішень	10.03.2021- 29.03.2021	Виконано
7	Створення API для роботи з ПЗ	30.03.2021 – 25.04.2021	Виконано
8	Реалізація генетичного алгоритму	26.04.2021 – 5.05.2021	Виконано
9	Написання SAD для ПЗ	6.05.2021 – 10.05.2021	Виконано
10	Написання пояснювальної записки	11.05.2021 – 25.05.2021	Виконано

Студент – бакалавр _____/Владислав ГРОМЕНКО

Керівник роботи _____/Віктор ШЕВЧЕНКО

АНОТАЦІЯ (українською)

Випускна кваліфікаційна бакалаврська робота: 86 с., 21 рис., 14 табл., 29 джерел, 2 додат.,

Тема: Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень

Об’єкт дослідження: рекламні дані, пов’язані з предметною області нерухомості, представлені у вигляді реляційних структур даних та часових рядів.

Мета роботи: розробка формальних та програмних методів для автоматизації перебігу рекламної кампанії декількох продуктів, з обмеженими бюджетами.

Предмет дослідження: технології для роботи з даними у вигляди часових рядів, методи для прогнозування часових рядів з використанням машинного навчання, підхід до автоматизації прийняття рішень на основі генетичних алгоритмів.

Результати дослідження: досліджено можливість використання методів машинного навчання та генетичних алгоритмів до виділених задач автоматизації управління. Запропоновано програмний підхід до автоматизації управління рекламною кампанією об’єктів предметної області, пов’язаної з нерухомістю.

Висновок: в результаті досліджень було отримано підхід, що застосовується для автоматизації управління перебігу рекламних кампаній одночасно багатьох продуктів з різними бюджетами, та дозволяє оптимізувати витрати. Створено програмне забезпечення для моделювання та реалізації даного підходу.

БД, REST API, МАШИННЕ НАВЧАННЯ, ЧАСОВІ РЯДИ, ГЕНЕТИЧНИЙ АЛГОРИТМ.

АНОТАЦІЯ (російською)

Выпускная квалификационная бакалаврская работа: 86 с., 21 Рис., 14 Табл., 29 источников, 2 прилож.

Тема: Программное обеспечение автоматизации управления рекламной кампанией в условиях ресурсных ограничений

Объект исследования: рекламные данные предметной области недвижимости, представленные в виде реляционных структур данных.

Цель работы: разработка формальных и программных методов для автоматизации рекламной кампании нескольких продуктов, с ограниченными бюджетами.

Предмет исследования: технологии для работы с данными в виде временных рядов, методы для прогнозирования временных рядов с использованием машинного обучения, подход к автоматизации принятия решений на основе генетических алгоритмов.

Результаты исследования: исследована возможность использования методов машинного обучения и генетических алгоритмов к выделенным задач автоматизации управления. Предложено программный подход к автоматизации управления рекламной кампанией объектов предметной области, связанной с недвижимостью.

Вывод: в результате исследований было получено подход, который применим для автоматизации управления рекламных кампаний одновременно многих продуктов с различными бюджетами, и позволяет оптимизировать затраты. Создано программное обеспечение для моделирования и реализации данного подхода.

БД, REST API, МАШИННОЕ ОБУЧЕНИЕ, ВРЕМЕННЫЕ РЯДЫ, ГЕНЕТИЧЕСКИЙ АЛГОРИТМ.

АНОТАЦІЯ (англійською)

Final qualifying bachelor's work: 86 p., 21 Fig., 14 Tab., 29 Sources, 2 additions.

Topic: Advertising campaign management automation software in terms of resource constraints.

Research object: advertising data of the real estate subject area, presented in the form of relational data structures.

Purpose of work: development of formal and programmatic methods for automating an advertising campaign for several products, with limited budgets.

The subject of research: technologies for working with data in the form of time series, methods for forecasting time series using machine learning, an approach to automated decision-making based on genetic algorithms.

Results of the research: The possibility of using machine learning methods and genetic algorithms for the selected control automation problems were investigated. A programmatic approach to automating the management of an advertising company of objects in the subject area related to real estate is proposed.

Conclusion: As a result of the research, an approach was obtained that is applicable to automate the management of advertising campaigns simultaneously for many products with different budgets and allows you to optimize costs. The software has been created for modelling and implementing this approach.

DB, REST API, MACHINE LEARNING, TIME SERIES, GENETIC ALGORITHM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД - база даних.

СУБД – система управління базами даних.

TSDB – Time series database.

ОС – операційна система.

ПЗ – програмне забезпечення.

DTO – Data Transfer Object, певний клас, екземпляри якого використовуються для передачі даних між класами, компонентами програми.

Трафік – кількість користувачів певного веб-ресурсу.

ВСТУП

Актуальність роботи

У сучасному світі будь-який створений продукт потребує реклами, для того щоб кінцевий користувач дізнався та купив його. При цьому, для продуктів, що рекламують кілька різних товарів задача управління кампанією кожного може бути досить нетривіальна, адже зазвичай існує кілька рекламних каналів (наприклад банери на сайтах партнерах, преміальне розміщення в розділах власного сайту, реклама в соціальних мережах і тп.), кожен зі своїм обмеженням рекламних переходив, та також у окремих товарів є власний рекламний бюджет, за який не треба виходити.

Тож, з зростанням кількості активних користувачів Інтернету зростає і потреба в якісному управлінні рекламою.

Порівняння роботи з відомими розв’язаннями проблеми

Створення дієвого методу для управління рекламою є комплексною задачею, яку можна поділити на декілька пунктів:

1. Побудова інфраструктури для видачі рекламного контенту.

2. Оптимізація показу реклами певного об’єкту/товару, таргетинг.

3. Оптимізація планування рекламних ресурсів.

У всіх трьох напрямках проводяться активні дослідження та розвиток. Наприклад існують роботи з оптимізації показу реклами з використанням генетичних алгоритмів [1] та алгоритму стохастичного навчання ранжируванню [2].

Окрім дослідницьких робіт також існують готові продукти, які певною мірою вирішують цю велику комплексну задачу, наприклад Google Ad Manager, HubSpot Ad Tracking, та інші. Найбільш точно вони вирішують проблему з побудовою інфраструктури та автоматизації реклами, а щодо

проблем оптимізації, важко сказати які рішення вони використовують, адже ця інформація не знаходиться у вільному доступі.

У даній роботі розглядається одна з можливих реалізацій частини інфраструктури рекламної мережі, та найбільший акцент зроблений на побудові методу оптимізація планування рекламних ресурсів, з використання генетичного алгоритму для прийняття рішень, та методів машинного навчання для аналізу рекламних даних, з урахуванням обмежень, у вигляді різних рекламних бюджетів продуктів.

Мета і задачі дослідження

Метою бакалаврської роботи є покращення якості перебігу рекламної кампанії за рахунок автоматизації планування та зменшення залученості людини в процес планування. Досліджуваний метод повинен створити рекламний план для множини продуктів, у якому вказується, чи потрібно рекламувати продукт в певний день на певному рекламному каналі. Цей план базується на історичних даних по перебігу рекламних кампаній для кожного продукту по кожному рекламному каналу. Особливими вимогами є те що, план має враховувати рекламний бюджет окремих продуктів, а також обмежену кількість рекламних переходів, які може видати кожен рекламний канал.

Досягнення мети включало розв’язання таких задач:

1. Аналіз наявних рішень серед БД для часових рядів та вибір оптимальної.

2. Аналіз та проектування моделей, заснованих на статистичних методах, для прогнозування часових рядів.

3. Проектування системи управління рекламою на основі генетичного алгоритму, для створення рекламного плану.

4. Реалізація ПЗ для управління рекламною кампанією.

Об’єктом дослідження є процес перебігу рекламної кампанії одночасно декількох продуктів по певній множині каналів, з обмеженими бюджетами.

Предметом дослідження є підхід до автоматизації рекламної кампанії, досліджується можливість використання методів машинного навчання та генетичного алгоритму.

Методи дослідження

Для створення предикативної моделі до наявних часових даних з предметної області, застосовуються методи регресійного аналізу та аналізу часових рядів. Для проектування БД з наявних даних використовується апарат реляційної алгебри. Для реалізації підходу до автоматизації рекламної кампанії управління використовуються метод, який базуються на генетичному алгоритмі.

Наукова новизна отриманих результатів

Наукова новизна роботи полягає у обґрунтуванні використання поєднання статистичних моделей прогнозування разом з генетичним алгоритмом для автоматизації перебігу рекламної кампанії:

1. Спроектовано метод для автоматизації управління рекламою, у якому прийняття рішень виконується генетичним алгоритмом.

2. Проведено аналіз статистичних моделей для прогнозування часових рядів у контексті прогнозування трафіку продуктів нерухомості.

Практичне значення одержаних результатів

Виконано програмну реалізацію зазначених моделей та методу для управління рекламними кампаніями. Одержана програмна реалізація

зазначеного підходу до управління рекламною кампанією використовує прогнозовані дані часових рядів у роботі генетичного алгоритму. Автоматичне прийняття рішень дозволяє зменшити залученість людини у керування рекламною кампанією.

Особистий внесок студента

Основним результатом є:

1. Запропонований метод до вирішення проблеми управління рекламною кампанією у предметній області пов’язаній з нерухомістю.

2. Приведена програмна реалізація підходу, яка на основі історичних даних у вигляді часових рядів, прогнозує перебіг компанії для кожного окремого продукту, та будує розклад, у який день та по якому рекламному каналі повинен рекламуватись продукт.

Апробація результатів випускної кваліфікаційної бакалаврської роботи

Результати бакалаврського дослідження були представлені на 8-мій Східно-Європейська конференція “Математичні та програмні технології Internet of Everything” (14.05.2021, Київ). Зб.матер., КНУ ім.Т.Шевченка с.17-18 pp.17-18, с.18-20 pp.18-20. [3-4]

Структура та обсяг роботи

Робота викладена на 86 сторінках друкованого тексту, який складається із вступу, 3 розділів, висновків, списку використаних джерел (28 найменувань). Робота містить 14 таблиць, 21 рисунок, 2 додатоки.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ РЕКЛАМНОЮ КАМПАНІЄЮ. АНАЛІЗ НАЯВНИХ ДАНИХ ТА РІШЕНЬ

1.1 Формалізація задачі

Основною задачею у даній роботі є зробити систему, яка буде створювати план на певний рекламний період, що показує коли і в якому каналі будуть рекламуватися певні продукти, за умови, що кожен продукт має свій бюджет (виходячи за рамки якого реклама по ньому перестає приносити прибуток) та кожен канал має обмеження трафіку, тобто ресурс для отримання реклами також обмежено. Формальну модель даної задачі можна представити у вигляді наступної системи рівнянь:

$$\sum_{ch_{id}=1}^{Ch} \sum_{d_{id}=1}^{D} \sum_{p_{id}=1}^{P} revenue_{ch_{id},\, d_{id}, p_{id}} \rightarrow max, \tag{1.1}$$

$$\sum_{ch_{id}=1}^{Ch} \sum_{d_{id}=1}^{D} revenue_{ch_{id},\, d_{id}, p_{id}} \leq budget_{p_{id}}, p_{id} = 1,2 \dots, P$$

$$\sum_{d_{id}=1}^{D} \sum_{p_{id}=1}^{P} productTraffic_{ch_{id},\, d_{id}, p_{id}} \leq channelTraffic_{ch_{id}},$$

$$ch_{id} = 1,2, \dots, Ch$$

де Ch, P, D – кількість доступних рекламних каналів, кількість продуктів, кількість днів у рекламному періоді відповідно, ch_{id}, d_{id}, p_{id} – індитифікатори певного продукту, рекламного каналу, дня відповідно, $revenue_{ch_{id},\, d_{id}, p_{id}}$ – прибуток з реклами продукту по рекламному каналу за день, $budget_{p_{id}}$ – бюджет реклами продукту на рекламний період, $productTraffic_{ch_{id},\, d_{id}, p_{id}}$ – трафік продукту у конкретний день та за

певними каналом , $channelTraffic_{ch_{id}}$ – сумарний трафік певного рекламного каналу.

З формули (1.1) видно що основна задача – отримати максимум прибутку з реклами, зважаючи на всі зазначенні обмеження. Однак варто зазначити, що насправді наперед заданою величинами є тільки кількість продуктів та каналів, дні в рекламному періоді, бюджет окремих продуктів. Такі величини як трафік, від якого в свою чергу напряму залежить прибуток, є величиною не реальною а прогнозованою, адже задача є створити план на майбутній період, по якому реальних даних очевидно немає. Зважаючи на дані обставини, можна записати формулу (1.1) в дещо зміненому вигляді:

(1.2)

$$\sum_{ch_{id}=1}^{Ch}\sum_{d_{id}=1}^{D}\sum_{p_{id}=1}^{P} revenue_{ch_{id},\,d_{id},p_{id}} \rightarrow max,$$

$$\sum_{ch_{id}=1}^{Ch}\sum_{d_{id}=1}^{D} revenue_{ch_{id},\,d_{id},p_{id}} \leq budget_{p_{id}}, p_{id} = 1,2 \dots, P$$

$$revenue_{ch_{id},\,d_{id},p_{id}} = predProdTraffic_{ch_{id},\,d_{id},p_{id}} * pricePerClick_{d_{id},p_{id}}$$

$$\sum_{d_{id}=1}^{D}\sum_{p_{id}=1}^{P} predProdTraffic_{ch_{id},\,d_{id},p_{id}} \leq predChannTraffic_{ch_{id}},$$

$$ch_{id} = 1,2, \dots, Ch$$

$$abs\left(realProdTraffic_{ch_{id},\,d_{id},p_{id}} - predProdTraffic_{ch_{id},\,d_{id},p_{id}}\right) \rightarrow min$$

$$abs\left(realChannTraffic_{ch_{id},\,d_{id},p_{id}} - predChannTraffic_{ch_{id},\,d_{id},p_{id}}\right) \rightarrow min$$

де Ch, P, D – кількість доступних рекламних каналів, кількість продуктів, кількість днів у рекламному періоді відповідно, ch_{id}, d_{id}, p_{id} – ідентифікатори певного продукту, рекламного каналу, дня відповідно, $revenue_{ch_{id},\,d_{id},p_{id}}$ – прогнозований прибуток з реклами продукту по рекламному каналу за день, що залежить від трафіку цього каналу та ціни,

$budget_{p_{id}}$ – бюджет реклами продукту на рекламний період, $predProdTraffic_{ch_{id},\, d_{id},p_{id}}$ – прогнозований трафік продукту у конкретний день та за певними каналом, $pricePerClick_{d_{id},p_{id}}$ – ціна одного рекламного переходу для певного продукту, $predChannTraffic_{ch_{id}}$ – прогнозований сумарний трафік певного рекламного каналу, $realProdTraffic_{ch_{id},\, d_{id},p_{id}}$ – реальний трафік продукту у конкретний день та за певними каналом, $realChannTraffic_{ch_{id}}$ – реальний сумарний трафік певного рекламного каналу.

Формула (1.2) є більш повною, в порівнянні з (1.1), адже в ній також беруться до уваги саме прогнозовані дані та те що вони повинні мати малу розбіжність з реальними даними.

1.2 Опис методу вирішення задачі

Загалом всю задачу по створенню методу можна розбити на 4 пункти:

1. Аналіз наявних даних, для виявлення їх особливостей.

2. Створення моделі для прогнозування часових рядів, яка на вхід отримує всі наявні історичні дані та на виході надає прогноз для зазначеного періоду

3. Створення алгоритму, який реалізує формальну формулу 1.2, з використанням даних, прогнозованих попередньою моделлю.

4. Написання ПЗ, яке реалізує попередні описані моделі та алгоритми, поєднує їх, та також надає інфраструктуру для збереження та обробки даних та забезпечує доступ користувача до результату роботи методу.

Для знаходження значень прогнозованих змінних формули 1.2 було вирішено використовувати статистичні методи для опису та прогнозування часових рядів на основі історичних даних трафіку. У загальному методі ці моделі повинні створюватись для кожного продукту та рекламного каналу

окрему, тож ця частина може бути замінена на інші моделі або системи для прогнозу даних.

Для реалізації частини системи, яка відповідальна за розробку плану було прийнято рішення розробити алгоритм на основі генетичного алгоритму. Головна причина використання генетичних алгоритмів – це їх зрозумілість та відносна гнучкість, завдяки якій можна швидко налаштовувати алгоритм, та ввести нові евристики, під нові бізнес-правила, які можуть змінюватись кожен новий період, через потреби бізнесу до відповідності умовам клієнтів. Також серед плюсів алгоритму є його здібність до розпаралелювання. Водночас серед мінусів такого підходу варто зазначити його теоретичну недоведену збіжність та також те, що при рівних умовах для нескладних задач генетичний алгоритм буде працювати гірше ніж стандартні алгоритми пошуку (проте не гірше випадкового пошуку).

До того як перейти до написання ПЗ було проаналізовано наявні рішення для TSDB [5], виявлено їх слабкі та сильні сторони, та виявлено яка найкраще зможе забезпечити роботу заявленого методу.

1.3 Вибір рішення для бази даних

Основними критеріями, які були враховані у цій роботі, під час вибору БД, є:

1. Природа наявних даних з предметної області. Більша кількість інформації, що розглянута у роботі представлена у вигляді часових рядів, але також наявна додаткова інформація, для налаштування рекламних компаній.

2. Здатність до масштабування, зважаючи на кількість даних. Часові ряди, зазвичай, мають великий розмір, тож щоб ПЗ було масштабованим перш за все БД повинна мати можливість масштабуватись.

3. Швидкість роботи, та здатність до оптимізації запитів, для прискорення роботи всього ПЗ.

1.3.1 Аналіз існуючих рішень БД, для роботи з часовими рядами

Виходячи з першого критерію вибору БД, які були описані вище, зрозуміло, що треба аналізувати БД, спеціалізовані для роботи та збереження саме часових рядів (Time series database), адже вони мають такі переваги над іншими:

1. Ефективне зберігання даних. За самою природою типу даних обробка може потребувати величезного обсягу пам’яті, управління яким може бути важким і дуже дорогим. Бази даних часових рядів мають інструментарій, який дає можливість агрегувати дані у заздалегідь визначені періоди часу та усунути певні потоки даних за необхідності та використовувати алгоритми стиснення, що оптимізують зберігання.

2. Швидкі запити даних. TSDB може також спростити запит та отримання даних на основі певних періодів. Наприклад, уявіть того, хто не пам’ятає назву книги, яку нещодавно читав, але знає, що читав її три місяці тому. Бази даних часових рядів можуть допомогти людині зрозуміти, якою була книга, не використовуючи купу пошукових символів. Використовуючи базу даних часових рядів, можливо швидко знайти інформацію на основі часових рамок – що дозволяє швидко шукати.

Тож весь наступний аналіз буде відбуватися саме для TSDB, а інші бази даних розглядатись не будуть. У якості БД, які варто перевірити, було взято рейтинг з наступного графіку та таблиці до нього.

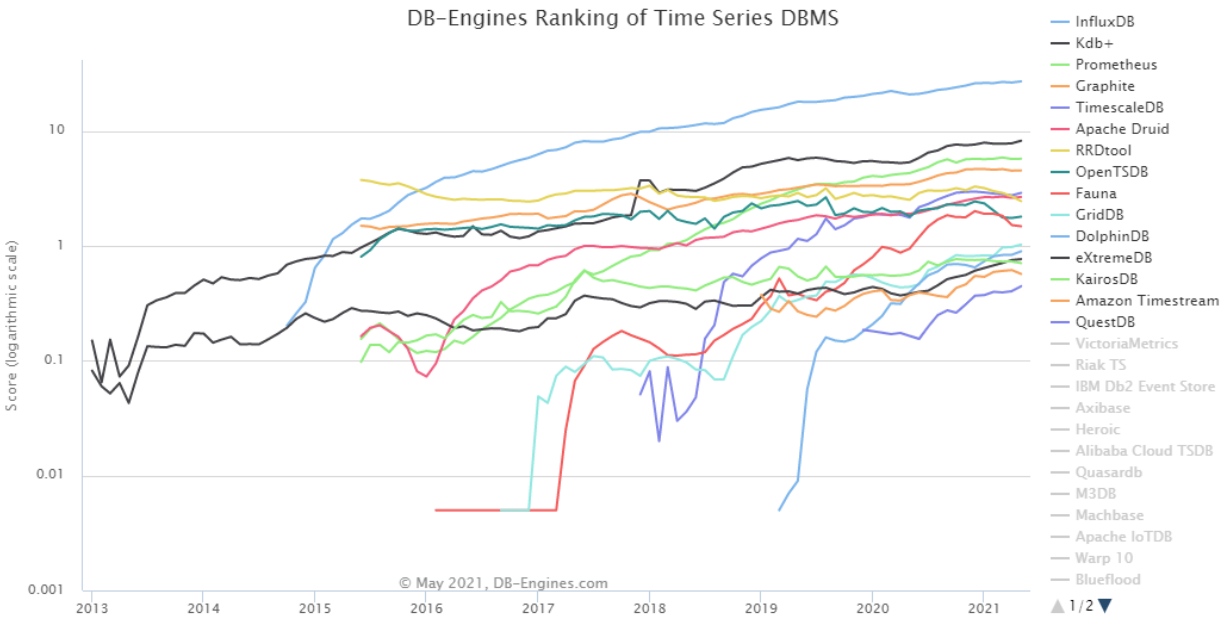


Рис. 1.1. Графік популярності TSDB за часом [6]

Rank			DBMS	Database Model	Score		
May 2021	Apr 2021	May 2020			May 2021	Apr 2021	May 2020
1.	1.	1.	InfluxDB	Time Series, Multi-model	27.17	+0.62	+6.25
2.	2.	2.	Kdb+	Time Series, Multi-model	8.26	+0.41	+2.89
3.	3.	3.	Prometheus	Time Series	5.76	+0.03	+1.45
4.	4.	4.	Graphite	Time Series	4.56	+0.03	+1.10
5.	5.	↑ 8.	TimescaleDB	Time Series, Multi-model	2.90	+0.13	+1.13

Рис. 1.2. Таблиця популярності різних TSDB

1.1.1.1 InfluxDB

InfluxDB [7] – це NoSQL [8] база даних з часовими рядами із відкритим кодом.

Основні плюси InfluxDB:

1. Легко налаштовувати, немає багатьох залежностей.

2. Спеціалізовано під часові ряди, може оптимізувати запити, знаючи структуру даних, та зберігає дані у порядку часу.

3. Має версію з відкритим кодом, яку можна використовувати для розробки.

4. Відсутність схеми даних, в сенсі того, що можна легко змінити формат даних, які зберігаються.

Незважаючи на плюси, мінуси використання InfluxDB випливають саме з них, це:

5. Відсутність схеми даних, в сенсі того, що немає додаткової перевірки на коректність даних, та складніше підтримувати відповідність ACID.

6. Версія з відкритим кодом має обмеження.

7. Спеціалізація під часові ряди робить неможливим використання БД для інших потреб, та також обмежує використання запитів на зміну та видалення даних (адже для часових рядів вони не часто зустрічаються).

1.1.1.2 Kdb+

KDB + [9]- це колонна база даних, яка використовується для великих обсяги даних, впорядкованих певним чином (наприклад, за часом).

Переваги Kdb+:

1. Розмір. За сучасними мірками KDB + має дуже малий розмір. Це лише один виконуваний файл розміром менше мегабайта і один невеликий текстовий файл з деякими системними функціями. Також, завдяки цьому інтерпретатор здатен повністю поміститись у кеш процесора, що може прискорити виконання.

2. Універсальність. БД не концентрується на якомусь певному домені, а, завдяки вбудованій мові, можна написати власну логіку, якої не вистачає.

3. Швидкість. Завдяки реляційній моделі даних, та оптимізації зберігання даних, а також векторній мові Q у Kdb+ досягнута значна швидкість виконання.

Основним недоліком Kdb+ є складність роботи з цією БД, та високий поріг входження, через те, що мова Q має незвичний та складний синтаксис, а QSQL (діалект SQL у Kdb+) вимагає від розробника самотужки оптимізувати різні запити, що може бути досить складною задачею.

1.1.1.3 Prometheus

Prometheus [10]– це рішення з відкритим кодом, яке використовується для моніторингу, розуміння статистичних даних з метричних даних та надсилання необхідних попереджень. Має у собі локальну базу даних часових рядів на диску, яка зберігає дані у спеціальному форматі на диску.

Переваги Prometheus:

1. Має багатовимірну модель, яка використовує ім’я метрики та пари ключ-значення для зберігання та запитів.
2. Має мову PromQL для запиту даних часових рядів.
3. Має потужні інструменти візуалізації даних.
4. Використовує pull mode HTTP для збору даних про часові ряди.

Основним недоліком Prometheus є те що його зазвичай використовують саме як систему моніторингу роботи ПЗ, адже у нього немає підтримки зберігання даних на тривалий період, тож як самостійне сховище даних, яке можна використати для нарощування бізнес-логіки над ним, Prometheus не є оптимальним рішенням.

1.1.1.4 Graphite

Graphite [11]– це засіб моніторингу, який призначений для роботи як на фізичному обладнанні так і в хмарній інфраструктурі. Зазвичай команди розробників використовують Graphite для відстеження продуктивності веб-сайтів, програм, додатків та мережевих серверів, адже дане рішення здатне полегшити зберігання, отримання, обмін та візуалізацію даних часових рядів.

Основною перевагою Graphite є його простота, адже дане рішення не надає якоїсь особливої, складної функціональності. Проте також дане рішення не є досить ефективним [12].

1.1.1.5 TimescaleDB

TimescaleDB [13] – це база даних часових рядів з відкритим кодом, оптимізована для швидкої вставки нових даних та виконання складних запитів. Основною особливістю цього рішення є те що воно повністю використовую SQL і відповідно є досить простим у використанні, звичним для більшості розробників, та має багату функціональність даної мови. Також це традиційна реляційна база даних, але яка здатна масштабуватися способами, які зазвичай використовують NoSQL БД.

Основні переваги TimescaleDB:

1. Легкість використання, яка досягається за допомогою повної інтеграції з звичним інтерфейсом мови SQL, який підтримується PostgreSQL, та легка інтеграція з усіма рішеннями, які підтримують PostgreSQL. А також, звичайно, додаванням спеціальних функцій, які корисні для роботи з часовими рядами.

2. Масштабованість, завдяки наступним особливостям:

2.1 Прозорість розділення часу та місця, яке дозволяє масштабувати рішення в рамках одного вузла та на кілька вузлів.

2.2 Висока швидкість запису даних (через наявність batching, in-memory індексів та інших особливостей).

2.3 Підтримка паралельних операцій між кількома серверами або чанками даних.

3. Надійність, основною причиною якої є побудова TimescaleDB над PostgreSQL, БД, яка розвивається вже більше 20 років, та є однією з

найпопулярніших серед розробників. З цього ж випливає й те що TimescaleDB повністю сумісна з всією екосистемою PostgreSQL.

Недоліки даної БД випливають з особливостей архітектури TimescaleDB, і основний це те що дане рішення використовує реляційну модель даних та, як результат, фіксовану схему даних, що може бути серйозним обмеженням, адже якщо зміняться мета дані часового ряду, то доведеться змінювати наявну схему (або створювати нову), та робити міграцію даних, що є коштовною операцією, особливо якщо система знаходиться під високим навантаженням.

1.3.2 Вибір БД та його обґрунтування

Зважаючи на результати аналізу різних БД, що був наведений у попередньому пункті, було виявлено, що Prometheus та Graphite, не будуть гарним вибором для побудови інфраструктури на їх основі, адже по-перше, зазвичай вони використовуються для моніторингу програмних систем, що не відповідає цілі даної роботи, а по-друге мають проблеми або з довгостроковим зберіганням даних, або мають слабші показники продуктивності, ніж інші розглянуті рішення.

Kdb+ навпаки має гарні характеристик продуктивності, та зберігання інформації, і той факт, що вона зазвичай застосовується у фінансових системах також слугує на користь вибору саме її, проте той факт, що ця БД є досить складною у використанні, має високий поріг входу та не займається оптимізацією запитів самотужки (а просто інтерпретує QSQL) ставе під сумнів доцільність її використання. Також вона не є безкоштовною, і немає відкритого коду, що робить її не найкращим вибором для даної роботи, зважаючи що всі її плюси можна побачити тільки на дійсно великий об’ємах даних.

Як результат, з-поміж 5 БД залишилось дві: InfluxDB та TimescaleDB. Незважаючи на те, що обидва ці рішення здавалися аналогічними, вони мають суттєві архітектурні відмінності: InfluxDB – NoSQL БД, яка використовує власні, нетипові, структури для зберігання даних, немає наперед визначеної схеми та навіть має свою власну мову для запитів, а в свою чергу TimescaleDB – є стандартною реляційною БД, яка лише оптимізує зберігання даних х стандартних таблиць, має наперед визначену схему даних, та використовує SQL для різних маніпуляцій над даними. Зважаючи на вищеозначені особливості TimescaleDB може бути більш пріоритетним через його стандартизованість, адже у контексті постійної розробки ПЗ, зазвичай такі рішення є більш надійними та дають додаткову швидкість розробки, проте цього не було достатньо для остаточного вибору, тож було прийнято рішення порівняти саме ці дві БД між собою більш детально.

Так як детальне порівняння двох БД є досить масштабною задачею, проведення у даній роботі проводилось на базі наявних аналізів, характеристик, показників, що були виявлені в інших роботах.

Перш за все було переглянуто порівняльну якісну характеристику цих двох БД, яка надається командою DB-Engines [14]. У табл. 1.1. наведена у скороченому варіанті, так як деякі параметри вже були розглянуті у даній роботі під час окремого аналізу кожної СУБД.

Таблиця 1.1

Порівняння характеристик InfluxDB та TimescaleDB.

Назва	InfluxDB	TimescaleDB
ОС	Linux, OS X	Linux, OS X, Windows
Типи	Числові дані та рядки	числа, рядки, булеві значення, масиви, JSON, геопросторові

		дані, валюти, двійкові дані, інші складні типи даних

Продовження табл. 1.1

Вторинні індекси	Ні	Так
SQL	SQL-подібна мова запитів	Так
API та інші методи доступу	HTTP API, JSON over UDP	ADO.NET, JDBC, native C library, ODBC, streaming API for large objects
Серверні скрипти	Ні	user defined functions, PL/pgSQL, PL/Tcl, PL/Perl, PL/Python, PL/Java, PL/PHP, PL/R, PL/Ruby, PL/Scheme, PL/Unix shell
Тригери	Ні	Так
Узгодженість		Сильна узгодженість
Зовнішні ключи	Ні	Так
Транзакції	Ні	ACID
Паралельність	Так	Так
Довговічність	Так	Так
Робота з користувачами	Керування правами через акаунти користувача	Детальні права доступу відповідно до SQL стандарту

Як видно з якісного порівняння в табл. 1.1 TimescaleDB має більш ширші можливості, як БД, так як підтримує більше функцій, механізмів, типів даних, а також дане рішення можна вважати більш гнучкою у налаштуванні безпеки ніж InfluxDB, адже TimescaleDB надає кращі можливості налаштування прав доступу для користувачів (за SQL стандартом).

Після порівняння якісних характеристик було зважено також дослідження показників продуктивності обох СУБД на однаковій множині

даних, яке було проведено командою Timescale [15]. Цей аналіз було використано у даній роботі, адже

1. Це одна з небагатьох робота у якій за багатьма характеристиками порівнюються саме InfluxDB та TimescaleDB.

2. В ній детально описано на яких даних та які характеристики перевіряються, з приведенням реальних числових показників.

З графіків/таблиць даних TimescaleDB показує себе краще за InfluxDb, на багатьох складних та великих запитах, проте може бути гіршою у більш простих та менш масштабних запитах. Також варто зазначити, що за результатами цього дослідження з збільшенням даних, значно зростає різниця в кількості затраченого місця на диску між TimescaleDB та InfluxDb.

Зважаючи на результати проведеного аналізу було прийнято рішення використовувати у подальшій роботі саме TimescaleDB, адже ця СУБД має такі переваги:

- SQL інтерфейс, та фундамент у вигляді PostgreSQL, що надає їй стабільності та легкості в управлінні.
- Можливість працювати не тільки з часовими рядами, завдяки реляційній моделі, що надає можливість зберігати у ній інші сутності.
- Має прийнятні показники продуктивності, у рамках даної задачі.

1.4 Аналіз часових рядів у наявних рекламних даних

1.4.1 Опис наявних даних

Першим кроком у аналізі рядів була побудова графіків цих рядів, для того щоб наочно можна було побачити дані та побудувати можливі гіпотези про них.

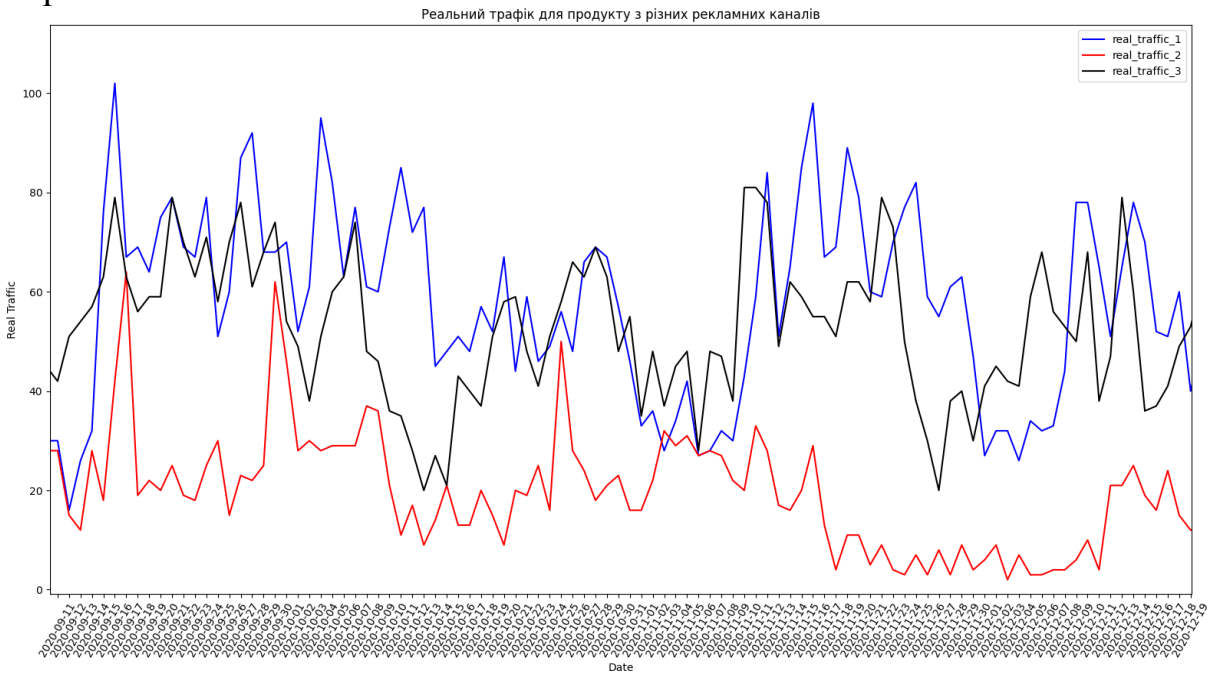


Рис. 1.3. Часовий ряд трафіку для конкретного продукту по 3 рекламним каналам

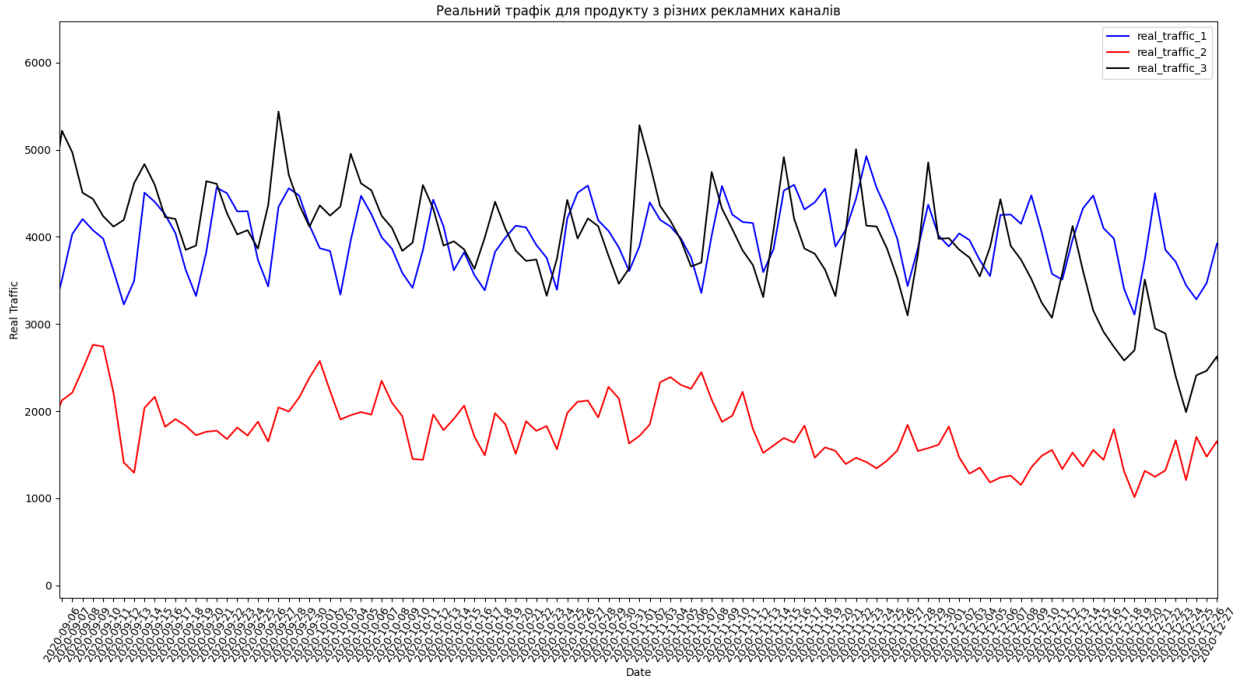


Рис. 1.4. Часовий ряд сумарного трафіку по 3 рекламним каналам

На рис. 1.3. та рис. 1.4. представлено графіки трафіку по 3 рекламним каналам для одного конкретного продукту та сумарного трафіку відповідно. Як видно з них трафік по наявних каналах немає чітко вираженого тренду, також він здається стаціонарним, адже має однаково середнє значення на всьому проміжку часу, та немає ярко виражених аномальних значень . Саме ці гіпотези і будуть перевірені.

1.4.2 Визначення характеристик рядів

Перш за все було визначено наскільки у часових рядах великий розкид даних, адже це може свідчити про наявність аномалій, та необхідність згладжування. Даний тест був проведений за допомогою коефіцієнта варіації. У табл. 1.2 містяться отримані результатів коефіцієнтів:

Таблиця 1.2

Коефіцієнти варіації часових рядів

Назва ряду	Рекламний канал 1	Рекламний канал 2	Рекламний канал 3
Сумарний трафік рекламного каналу	0.1	0.25	0.11
Трафік продукту (усередненого) у рекламному каналі	0.31	0.73	0.3

З табл. 1.2. видно, що часові ряди, які відносяться до сумарного трафіку не мають великого розкиду, та загалом можуть вважатись однорідними, що полегшує їх подальшу обробку. Для рядів же трафіку окремих продуктів, можливо варто застосовувати методи, які здатні їх привести до більш однорідного вигляду, адже незважаючи на те що величини все ще є невеликим, що свідчить про невеликий розкид, це дані

для усередненого рекламного продукту, хоча можуть траплятись рекламні продукти і з значними коефіцієнтами >1.

Наступним кроком в аналізі ряду було визначення чи є він стаціонарним. Для цього було проведено розширений тест Дікі-Фуллера (ADF) [16], який тестує нульову гіпотезу про те, що у авторегресійній моделі ряду наявний одиничний корінь, що свідчить про те що він нестаціонарний. Відповідно альтернативна гіпотеза – ряд є стаціонарним. У результаті було отримано наступні показники p-значень та ADF-статистики (див. табл. 1.3).

Таблиця 1.3

Показники ADF тесту

Назва ряду	Рекламний канал 1	Рекламний канал 2	Рекламний канал 3
Трафік рекламного каналу	ADF Statistic -7.4935644719422 p-value 4.4399303497e-11	ADF Statistic -4.013159376721185 p-value 0.0013431509439206	ADF Statistic: -6.060149492774983 5 p-value: 1.2180891447895e-07
Трафік продукту	ADF Statistic -5.9087827414042 p-value 2.669316620e-07	ADF Statistic -3.803354154095930 p-value 0.0028731834384446	ADF Statistic -6.049911448621101 p-value 1.2849616021286e-07

По результатах тесту можна сказати що наявні ряди є стаціонарними, що значно може покращити моделювання, та полегшити розробку самої моделі. Проте під час тестування виникали випадки (з деякими часовими рядами продуктів) коли ряд не є стаціонарним, але це можливо вирішити завдяки інтеграції часового ряду, наприклад в моделі ARIMA [17].

Останнім кроком в отримані характеристик рядів було виділення з них сезонної компоненти, яка може бути важливою для побудови моделей в майбутньому, було виділення компоненту сезонності з них. Для цього зазвичай вважають, що ряд складається з 3 компонентів: тренду, сезонності та шуму. Розрізняють адитивну модель, для якої справедлива наступна формула:

$$y_i = t_i + s_i + n_i \tag{1.3}$$

де y_i – значення ряду на і-му кроці, t_i, s_i, n_i –компоненту тренду, сезонності та шуму на і-му кроці відповідно. Формула мультиплікативної моделі має ті ж самі компоненти та наступну формулу:

$$y_i = t_i * s_i * n_i \tag{1.4}$$

Зазвичай першу використовують для лінійного зростання ряду, а другу – для експоненціального.

Для того щоб провести декомпозицію рядів за адитивною моделлю, у роботі було виконано наступні кроки (на прикладі часового ряду для сумарного трафіку одного з каналів, проте кроки є аналогічними для інших наявних рядів):

1. Виключено компонент тренду. Цей крок було зроблено за допомогою застосування до ряду лінійного фільтру ковзного середнього, за кількома можливими періодами. Тобто, якщо позначити період середнього як 2a, то формула цього фільтру для елементу t буде мати наступний вигляд:

$$\widehat{m_t} = \sum_{k=-a}^{a} \frac{x_{t+k}}{2a} \tag{1.5}$$

де x_t – значення ряду на кроці t, 2a – період для середнього.

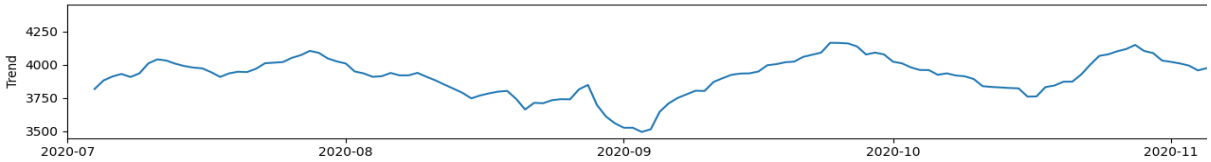


Рис. 1.5. Графік тренду часового рядку трафіку рекламного каналу

Як видно з формули, показник $\widehat{m_t}$ можливо отримати тільки починаючи з кроку a. У результаті було отримано наступний графік тренду (рис. 1.5.), який, як видно, коливається навколо одного значення.

2. Після отримання компоненту тренду, стало можливим його видалення з ряду, для того, що в результаті отримати сезонний компонент плюс помилку, з яким надалі можливо буде працювати, результатом є наступний рис. 1.6.

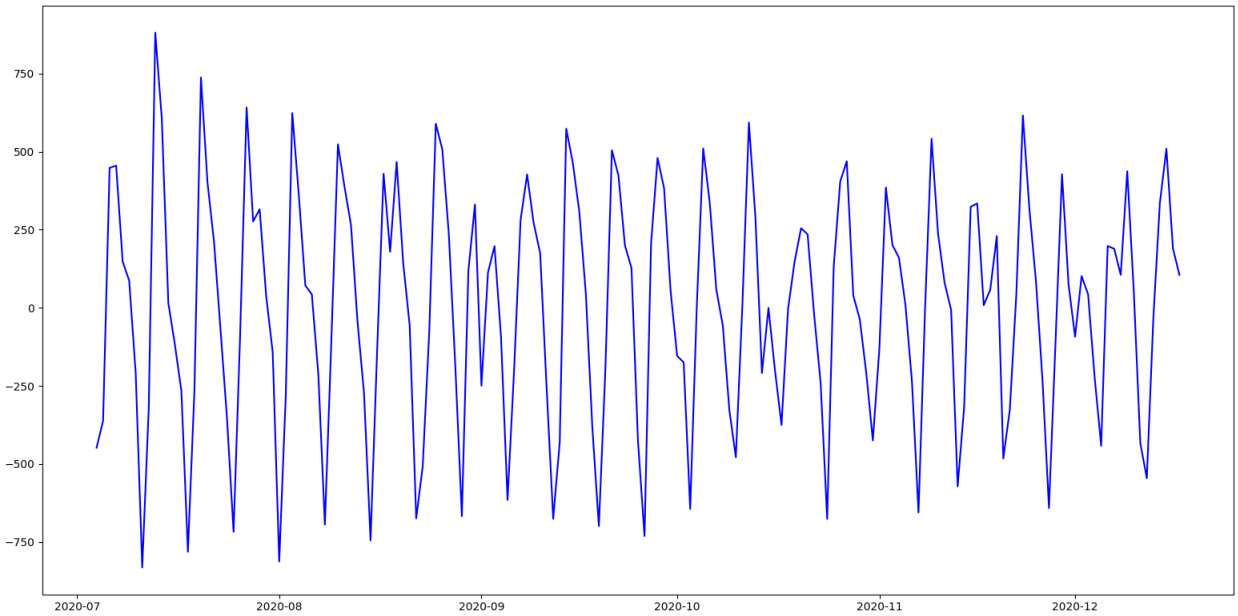


Рис. 1.6. Графік сезонності разом з помилкою трафіку рекламного каналу

3. З отриманого графіку можна виділити “чистий” компонент сезонності, який даст можливість повністю декомпозувати часовий ряд. Виділення компоненту сезонності відбувається за допомогою наступної формули

$$\widehat{s_t} = \sum_{k=0}^{[l/p]} \frac{x_{t\%p+k*p}}{[l/p]} \quad (2.4)$$

де l – довжина всього часового ряду, p – довжина періоду сезонності. Як видно з формули береться сердне значення по індексу $t\%p + k * p$, тобто елементів на однакових позиціях в рамках всіх періодів, та потім повторюється для всіх періодів.

У результаті було отримано наступний рис. 1.7. (для періоду 7 днів, нижче буде приведено пояснення чому було обрано саме його).

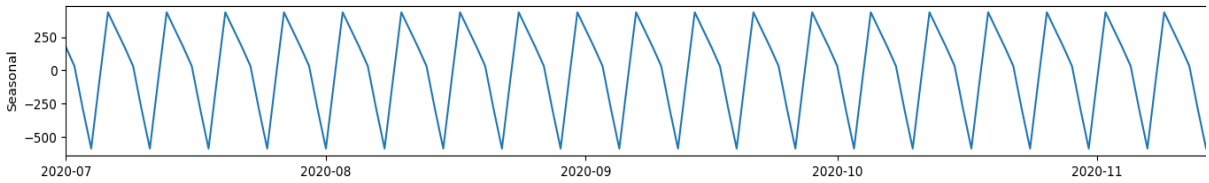


Рис. 1.7. Сезонний компонент трафіку рекламного каналу

Вище описані корки дали можливість отримати декомпозицію ряду на тренд та сезонність плюс помилка, проте при декомпозиції на кроці 3 видно, що періож треба задавати в декомпозицію заздалегіть, тож для того щоб правильно його обрати було прийнято рішення отримати середне значення помилки для всіх періодів сезонності на проміжку від 5 днів до 45. У результаті було виявлено, що найменшу середню помилку, надає

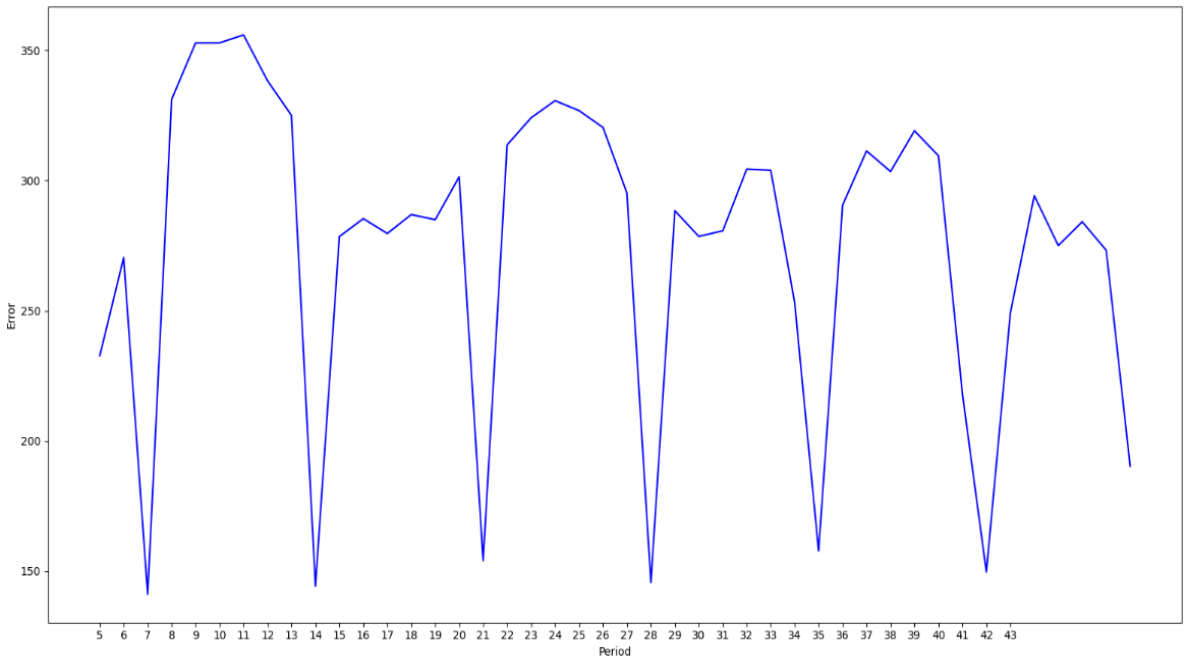


Рис. 1.8. Графік відношення середньої помилки до обраного періоду декомпозиції

декомпозиція з періодом у 7 днів (тобто у тиждень), що можна побачити з графіку на рис. 1.8.

1.5 Висновки до розділу

1. Було описано задачу автоматизації перебігу рекламної кампанії в умовах ресурсних обмежень. Під ресурсними обмеженнями було визначено бюджет на рекламний період певного продукту (при прибутку більше цього бюджету реклама перестає приносити користь) та також обмеження трафіку рекламного каналу, тобто він може видати не більше певної кількості рекламних переходів.

2. Використовуючи описання задачі її було формалізовано, результатом чого стала система формул (1.2). Спираючись на дану формалізацію було описано метод вирішення задачі, який включає статистичні методи для визначення прогнозованих змінних, та генетичний алгоритм, який має створювати рекламний план для, врахування обмежень.

3. Для того щоб розпочати роботу з даними було зроблено аналіз наявних рішень для БД, серед найпопулярніших TSDB, для кожної приведено плюси та мінуси, та за результатом обрано TimscaleDB, через комбінацію потужності та зручності використання.

4. Було проведено попередній аналіз наявних часових рядів, який був потрібен через те, що наявні дані, під які налаштовуються рішення можуть мати значний вплив на подальші дії по програмній реалізації. За результатом аналізу було виявлено що більшість часових рядів можливо з певною точністю віднести до стаціонарних, з невеликим розкидом даних, хоча серед рядів зустрічаються і нестаціонарні ряди з великим коефіцієнтом варіації, тож при проектуванні моделі для прогнозування треба врахувати що для цих різних видів наявних даних варто використовувати різні

підходи. Також було виявлено що ряди загалом мають сезонність, яка кратна 7.

РОЗДІЛ 2
РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ ЧАСОВОГО РЯДУ ТА МЕТОДУ ПРИЙНЯТТЯ РІШЕНЬ

Як було зазначено вище у методи до рішення задачі існують дані, які мають бути прогнозовані з історичних даних, які є часовими рядами, тож у даній частини буде описано підхід, який використовувався для їх прогнозування, тобто створення моделі, яка могла б передбачати майбутні значення, на основні попередніх спостережень.

Кінцевим результатом цього розділу є готова модель, яка здатна з певною точністю прогнозувати рекламний трафік по різних каналах на наступний календарний місяць, на основі історичних даних за попередні місяці. Для цього було використано, проведений у розділі 1, аналіз наявних часових рядів, а також проведене порівняння існуючих статистичних моделей, та потім на основі отриманих результатів, визначено роботу моделі прогнозування, яка буде використана при побудові ПЗ.

2.1 Аналіз моделей часових рядів

Після отримання деяких знань про наявні часові ряди до них було застосовано моделі прогнозування, які базуються на машинному навчанні, для того щоб з наявних історичних даних трафіку можна було робити прогнози на наступний рекламний період по сумарному трафіку каналу (який можна вважати його обмеженням) та трафіку окремих продуктів. Під час аналізу якості та оцінки отриманих моделей брались до уваги наступні критерії:

1. Критерій Люнга-Бокса, або ж Q-тест Люнга-Бокса [18].

2. R^2 або коефіцієнт детермінації.[19]

3. Середня абсолютна помилка (MAE) . [20]

4. Графік моделі.

2.1.1 Модель MA

Модель MA [21] (ковзного середнього) – стандартний підхід, до моделювання часових рядів, яка передбачає, що поточне значення процесу залежить від певної кількості попередніх значень, та середнього значення по всьому процесу (часовому ряду). Це видно з формули для обрахунку x_t члена ряду:

$$x_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \cdots + \theta_k\varepsilon_{t-k} \quad (2.1)$$

де μ – середнє значення всього ряду, $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-k}$ – значення помилки (шуму) на певному за індексом, а $\theta_1, \theta_2, \ldots, \theta_k$ – параметри моделі, умовно кажучи вага помилки на певному попередньому спостереженні. Також у формули є параметр k – кількість попередніх спостережень, які беруться у розрахунок при побудові моделі. Як видно з опису моделі вона концептуально схожа га лінійну регресію, тож її результат завжди буде статичним.

У роботі для підбору параметру k, який слід вказати перш ніж запускати моделювання, було використано функцію автокореляції ряду, яка показує залежність ряду від нього самого, при зсуві на певну кількість кроків. Для цього було знайдено значення автокореляції для відхилень від 1 до 30, які можна представити у вигляді графіку на рис. 2.1.

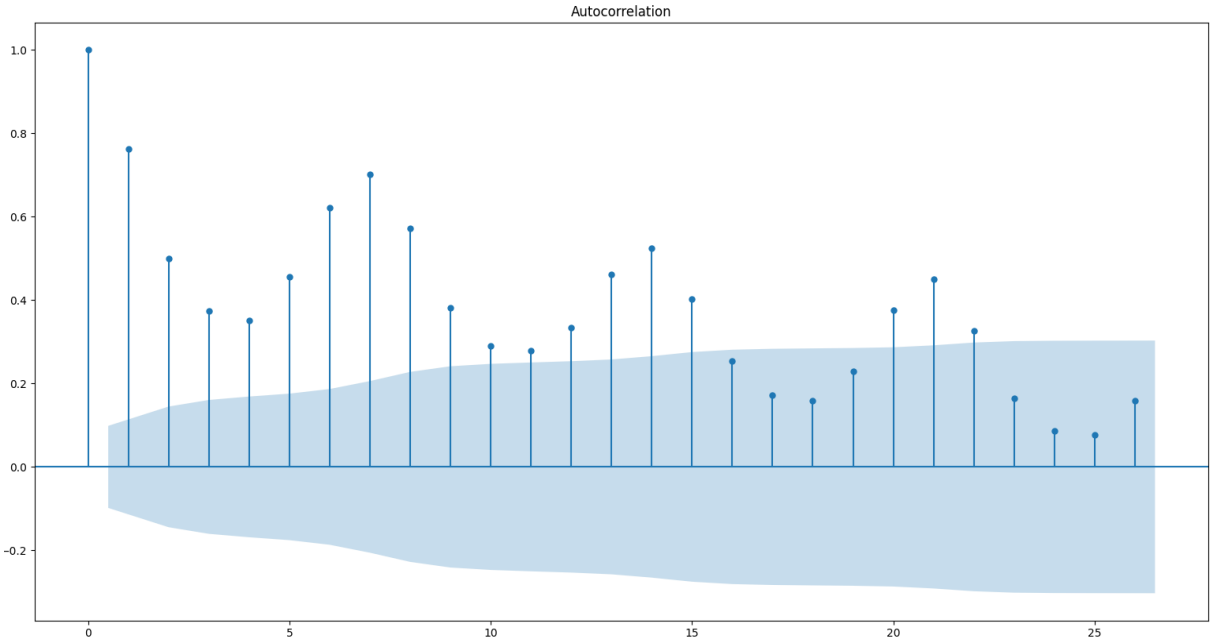


Рис. 2.1. Графік автокореляції часового ряду

У результаті, отримано модель, графік якої зображено на рис. 2.2., на якому синім кольором позначено реальне значення ряду, червоним – отримане моделлю, а також зверху знаходиться графік для тренувальний даних, а знизу – для тесту.

Як видно модель може повторювати певні загальні шаблони в часовому ряді, проте не робить цього належним чином, також після певного кроку, модель починає видавати у якості прогнозованого значення ряду його середнє значення. Оцінки ряду представлені у табл. 2.1.

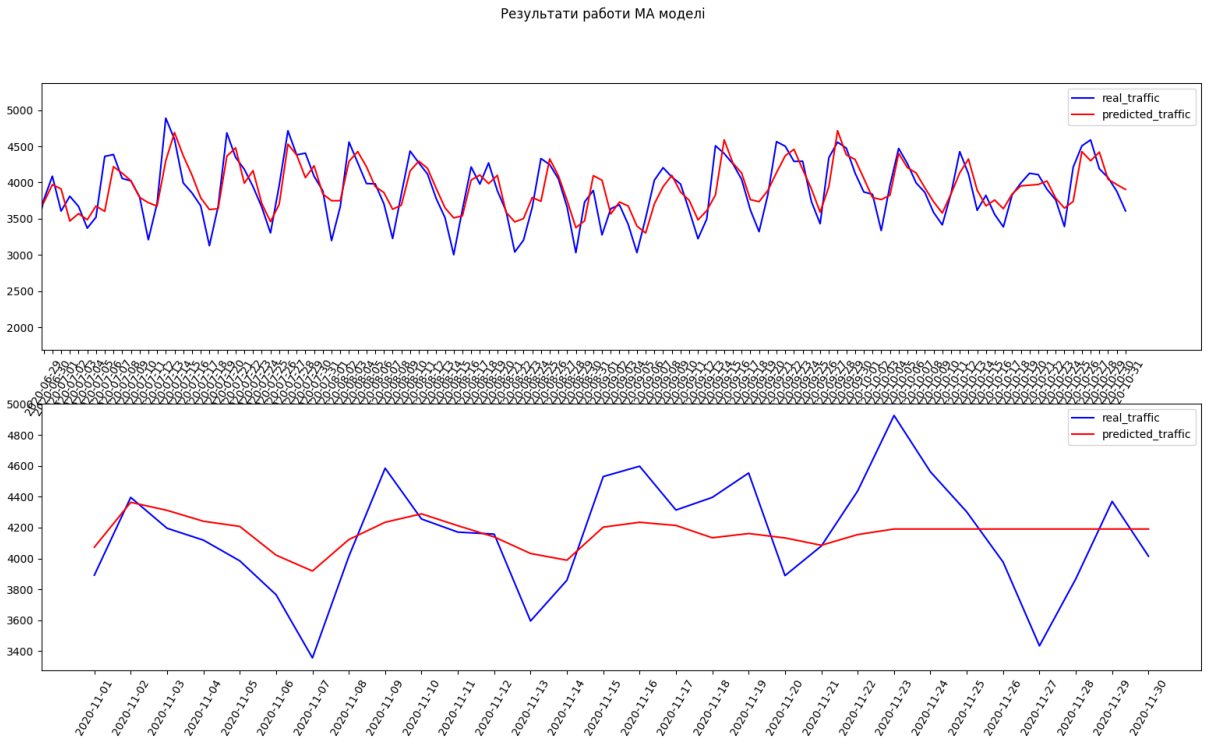


Рис. 2.2. Графік роботи MA моделі

Таблиця 2.1

Оцінка MA моделі

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.65	0.22
MAE	161.5	248.6
Критерій Люнга-Бокса	Починаючи з певного кроку p-value ~0.1 тож немає можливості сказати, що помилка моделі є «білим шумом».	

2.1.2 Модель AR

Модель AR [22] (авторегресійна модель) – модель часових рядів, у якій поточне значення ряду лінійно залежить від його попередніх значень. Зазвичай у моделі використовуються тільки певна кількість попередніх значень. Формула обрахунку поточного значення має наступний вигляд:

$$x_t = c + \sum_{i=1}^{p} a_i * x_{t-i} + \varepsilon_t, \quad (2.2)$$

де c – вільна константа, x_t – значення рядку на позиції t, a_i- параметри моделі, ε_t – помилка на кроці t. Також, як видно з формули, модель містить гіперпараметр p – який визначає, скільки попередніх спостережень впливають на поточне спостереження в моделі.

Для знаходження оптимального значення параметру p у роботі було застосовано функцію часткової автокореляції, яка схожа на функцію автокореляції, що застосовувалась до моделі MA, проте додатково для лагу p видаляє ще всі спостереження, між x_{t-p} та x_t, тобто не враховує кореляцію членів з меншими лагами. На рис. 2.3. представлено графік часткової автокореляції для часового ряду

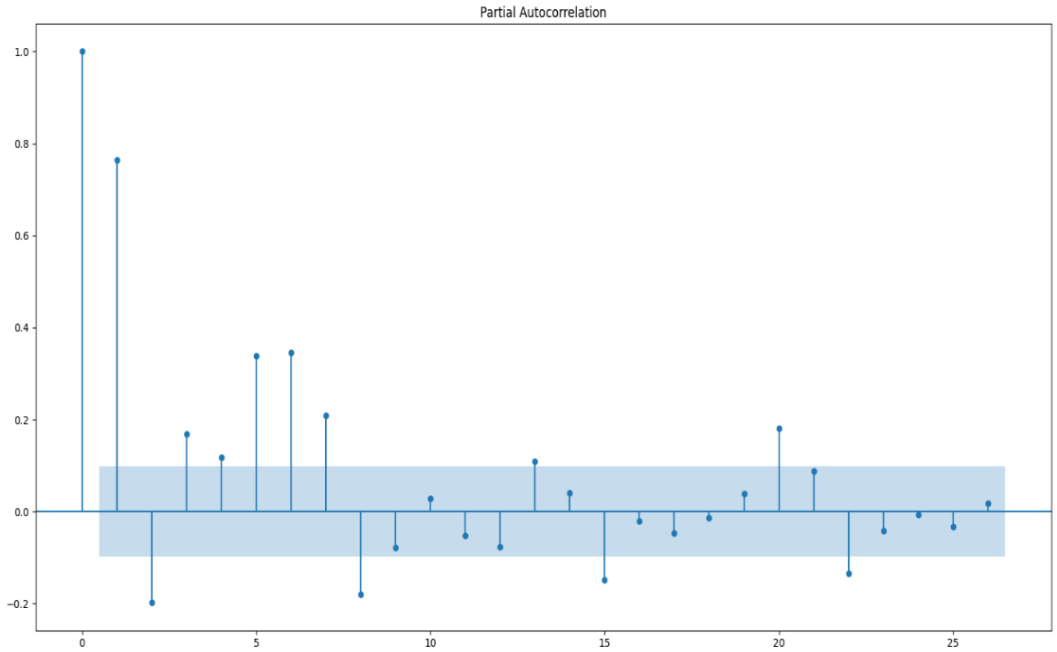


Рис. 2.3. Графік часткової автокореляції часового ряду

Сам алгоритм підбору оптимального значення з часткової автокореляції подібний до того, що використовується в MA. У результаті, отримано модель, графік якої зображено на рис. 2.4, де зверху знаходиться графік тренувальної вибірки, а знизу – тестовій.

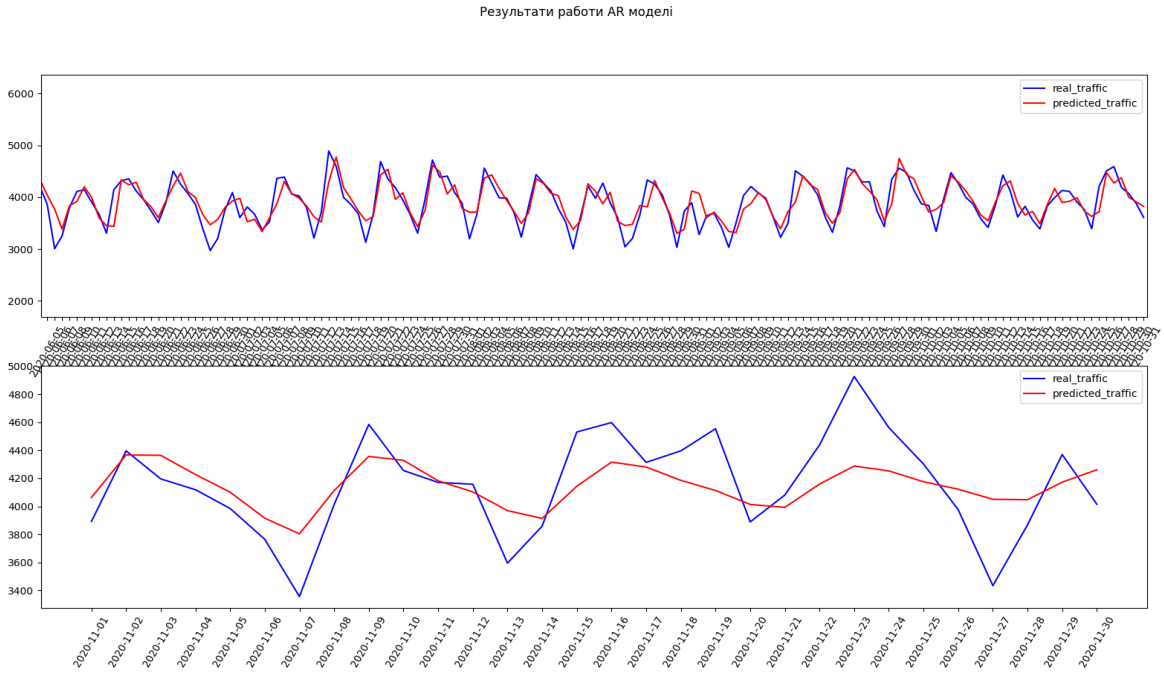


Рис. 2.4. Графік роботи AR моделі

Як видно з графіку (рис. 2.4) модель може добре повторювати реальний шаблон поведінки трафіку, проте чим далі ми намагаємось прогнозувати, тим гірше вона це робить, і поступово прямує до певної точки. Оцінки ряду представлені у табл. 2.2

Таблиця 2.2

Оцінка AR моделі

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.68	0.43
MAE	151	213.1
Критерій Люнга-Бокса	Мінімальне значення p-value для всіх параметрів автокореляції – 0.14, що не дає змоги відкинути гіпотезу про те що ряд відхилень (помилок) є «білим шумом»	

Можна зробити висновок, що модель є трохи кращою на тренувальній виборці, ніж MA (R^2 відрізняється на 0.03, та MAE на ~10), проте має суттєво кращі результати на тестовій виборці, більше того не так сильно втрачає якість за кількістю прогнозованих кроків, як MA.

2.1.3 Модель ARMA

Модель ARMA [23] (авторегресії – ковзного середнього) поєднує разом дві попередні моделі AR та MA. При такому комбінуванні AR використовується для знаходження залежності від попередніх значень ряду, а MA для знаходження залежності поточного відхилення від попередніх відхилень. Зрозуміло, що модель має два гіперпараметри p, q – як порядки частин AR та MA відповідно. Формула, яка описує модель, має наступний вигляд:

$$x_t = c + \varepsilon_t + \sum_{i=1}^{p} a_i * x_{t-i} + \sum_{j=1}^{q} \theta_j * \varepsilon_{t-j}, \quad (2.3)$$

де c – вільна константа, x_t – значення рядку на позиції t, a_i- параметри моделі AR, ε_t – помилка на кроці t, θ_j- параметри моделі MA.

Знаходження оптимальних значень параметрів p та q було зроблено таким самим чином як і знаходження p для AR та k для MA, та було описано у попередніх розділах. Результатом роботи моделі став графік, зображений на рис. 2.5.

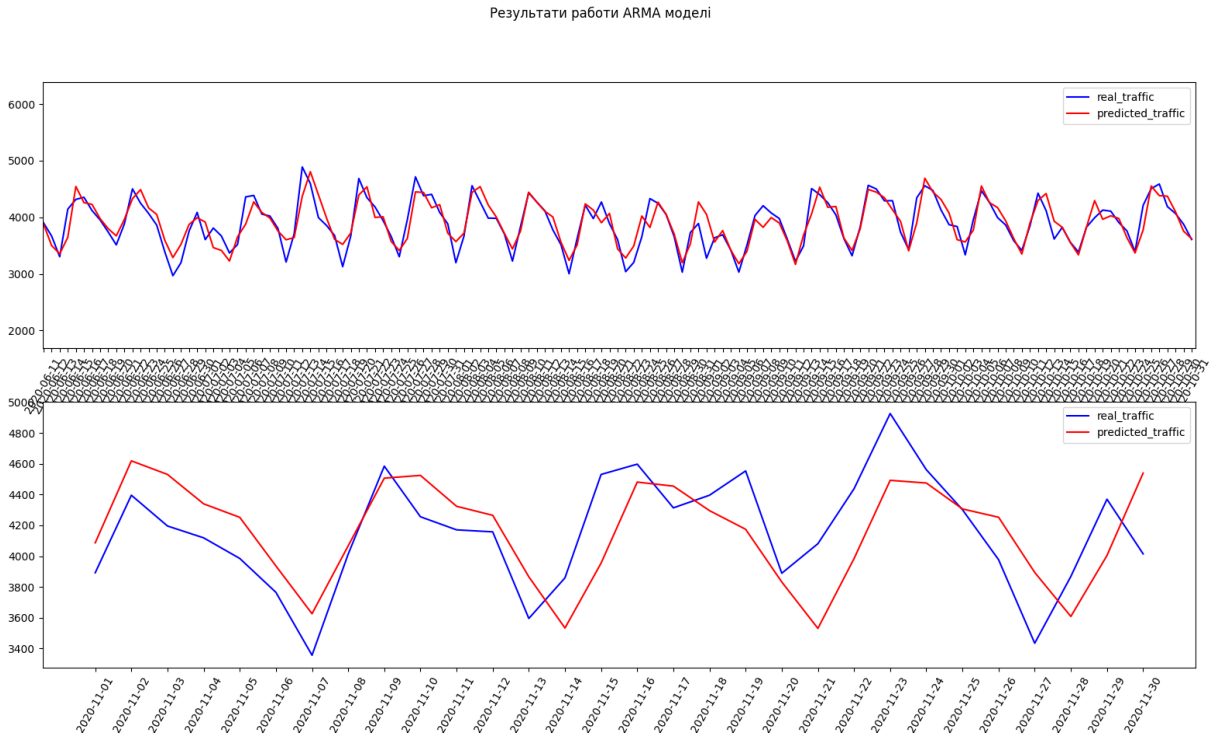


Рис. 2.5. Графік роботи моделі ARMA

Хоча на перший погляд здається що модель повторює реальні дані (навіть тестові) з хорошою точністю, провівши її оцінку, результати якої представлені у табл. 2.5, виявилось що модель насправді перенавчилася, адже вона істотно покращила результати на тренувальний виборці, але має при цьому має значно гірші результати на тестовій виборці, відносно попередніх моделей.

Таблиця 2.3

Оцінка ARMA моделі

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.75	0.28
MAE	124	257.6
Критерій Люнга-Бокса	Мінімальне значення p-value для всіх параметрів автокореляції – 0.9, завдяки чому можна стверджувати, що помилки є «білим шумом»	

Виходячи з цих оцінок було прийнято рішення для даної моделі, трохи змінити параметри (константи), для алгоритму підбору оптимальних значень p та q та за допомогою цього було отримано кращі результати, які представлені на графіку (рис. 2.6) та у табл. 2.4 нижче. Як бачимо модель має більш плавний вигляд, та хоча не досягає показників попередньої AR на тренувальній виборці, проте трохи покращує її на тестовій.

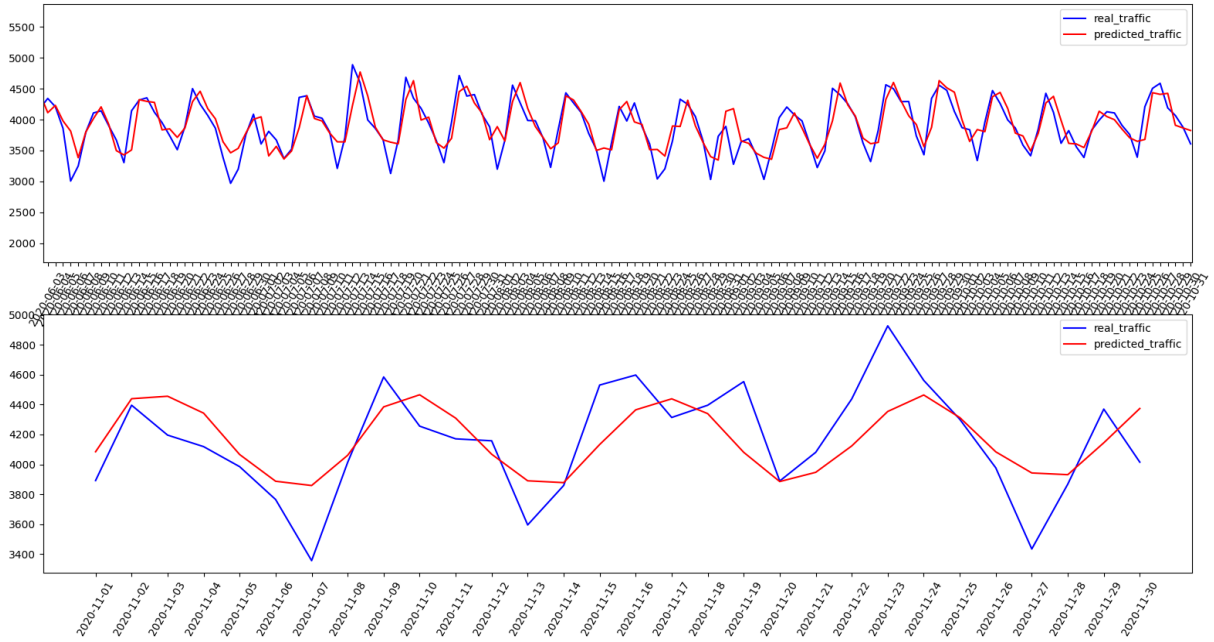


Рис. 2.6. Графік роботи моделі ARMA, друга версія

Таблиця 2.4

Оцінка ARMA моделі, друга версія

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.62	0.47
MAE	167.2	203.6
Критерій Люнга-Бокса	Мінімальне значення p-value для всіх параметрів автокореляції – 0.04, завдяки чому можна стверджувати, що з певного кроку «білий шум» все ж таки має якусь залежність	

2.1.4 Модель ARIMA

Модель ARIMA [17] (інтегрована авторегресія – ковзного середнього) – є розширенням моделі ARMA, яке можна застосовувати для опису та

прогнозування нестаціонарних рядів, роблячи їх стаціонарними за рахунок взяття різниці деякого порядку від наявного рядку (ця операція називається інтегруванням). Порядок інтегрування визначається гіперпараметром d моделі, тож її можна записати як ARIMA(p,d,q). Формула моделі має наступний вигляд:

$$\Delta^d x_t = c + \varepsilon_t + \sum_{i=1}^{p} a_i * \Delta^d x_{t-i} + \sum_{j=1}^{q} \theta_j * \varepsilon_{t-j}, \quad (2.4)$$

де c – вільна константа, x_t – значення рядку на позиції t, a_i – параметри моделі AR, ε_t – помилка на кроці t, θ_j – параметри моделі MA, Δ^d – оператор різниці з порядком d.

Так як під час аналізу виявилось, що деякі наявні часові ряди можуть бути нестаціонарними, то було прийнято рішення протестувати модель яка здатна працювати і з такими даними. При цьому було у якості порядку інтеграції було взято значення 1. Результатом цього став графік зображений на рис. 2.7.

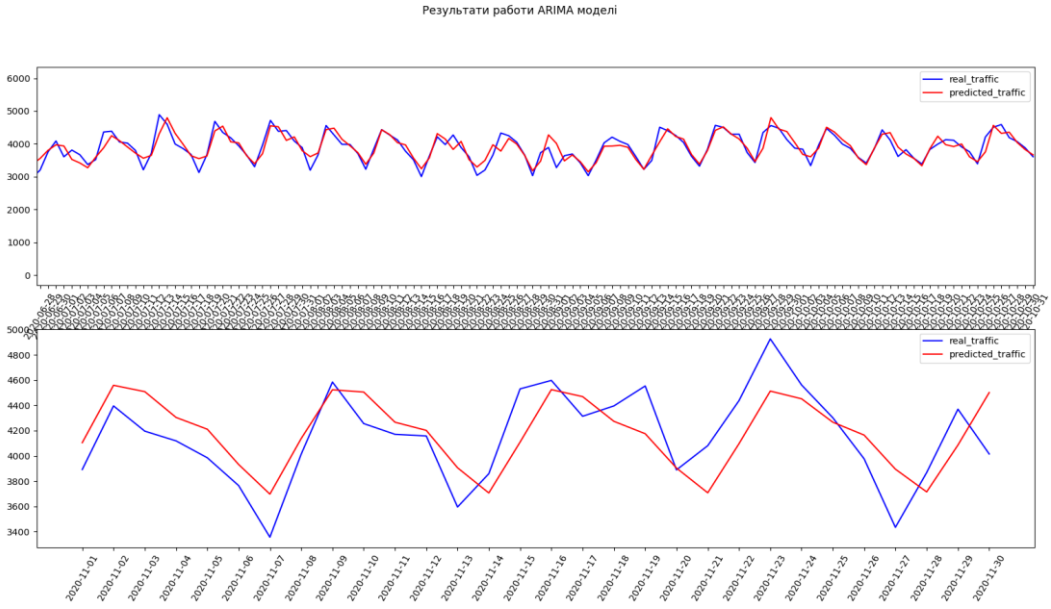


Рис. 2.7. Графік роботи моделі ARIMA

Таблиця 2.5

Оцінка ARIMA моделі

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.75	0.47
MAE	128.49	209.4
Критерій Люнга-Бокса	Мінімальне значення p-value для всіх параметрів автокореляції – 0.5, тож помилки моделі можна назвати «білим шумом».	

Можна зробити висновок, показує найкращі результати на тренувальний виборці, при цьому зберігаючи гарні показники на тестовій виборці, яка конкурує з попередніми моделями, враховуючи також більшу універсальність у роботі з не тільки стаціонарними рядами.

2.1.5 Модель SARIMA

Модель SARIMA [24] (сезонна інтегрована авторегресія – ковзного середнього) – покращення моделі ARIMA, розглянутої в попередньому розділі, яка окрім залежностей між даними на поточному кроці та попередніми, враховує також сезонну поведінку часових рядів. Сезонна частина моделі, схожа на не сезонну, проте включає в себе зсуви на певний сезонний період. Данна модель SARIMA(p,d,q)$(P,D,Q)_m$ має наступну формулу:

$$\Delta^d x_t = c + \varepsilon_t + \sum_{i=1}^{p} a_i * \Delta^d x_{t-i} + \sum_{j=1}^{q} \theta_j * \varepsilon_{t-j} + \sum_{k=1}^{P} \varphi_k * \sum_{l=0}^{p} a_i * \Delta^D x_{t-m*k-l} + \sum_{k=1}^{P} \omega_k * \sum_{l=0}^{p} \theta_j * \varepsilon_{t-m*k-l}, \quad (2.5)$$

де c – вільна константа, x_t – значення рядку на позиції t, a_i – параметри моделі AR (при чому $a_0 = 1$), ε_t – помилка на кроці t, θ_j – параметри моделі MA (при чому $\theta_0 = 1$), Δ^d – оператор різниці з порядком d, φ_k- параметри сезонної моделі для авторегресії, ω_k- параметри сезонної моделі для ковзного середнього, m – період сезонності.

Під час аналізу було визначено, що часові ряди мають період сезонності, кратний 7, тож треба було перевірити модель, яка здатна працювати з сезонними часовими рядами. Так як ця модель походить від

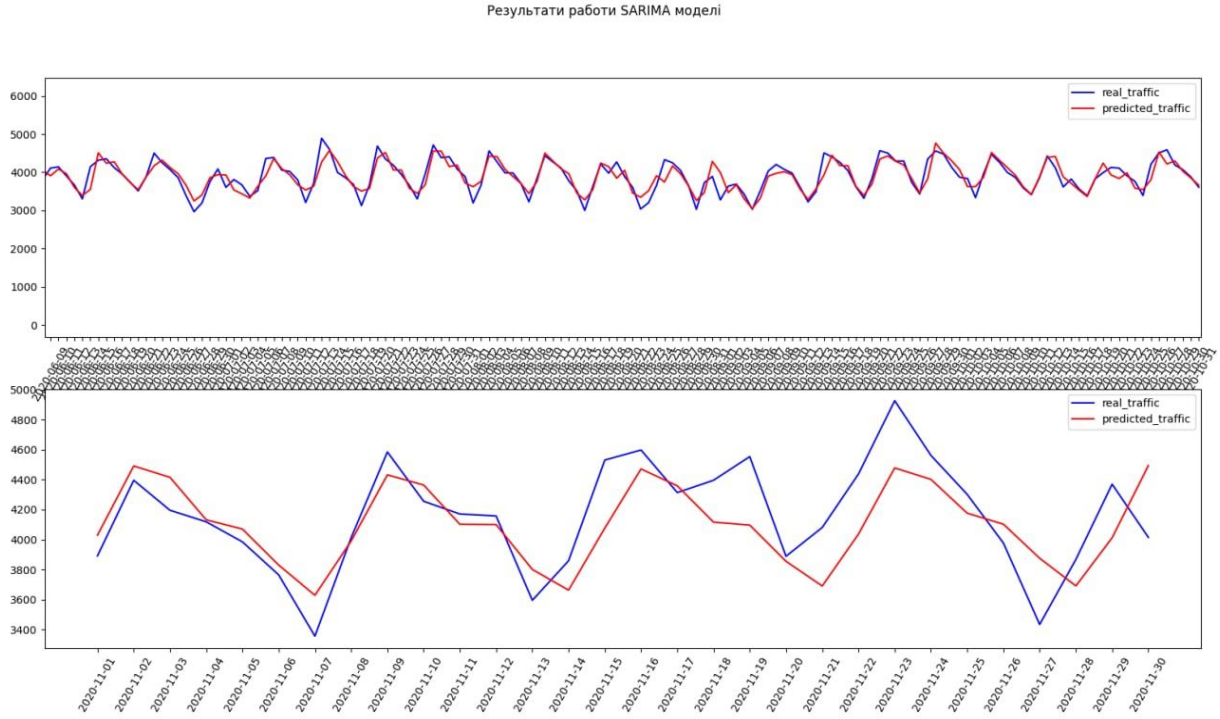


Рис. 2.8. Графік роботи моделі SARIMA

ARIMA то оптимізація параметрів проходила так само з використанням автокореляції та часткової автокореляції. Результатом перевірки моделі на тих же даних, що і попередні моделі, став графік на рис. 2.8, та дані у табл. 2.6.

Таблиця 2.6

Оцінка SARIMA моделі

Назва оцінки	На тренувальній вибірці	На тестовій вибірці
R^2	0.74	0.48
MAE	127.9	206.7
Критерій Люнга-Бокса	Мінімальне значення p-value для всіх параметрів автокореляції – 0.82, тож помилки моделі є «білим шумом».	

Хоча модель показує себе приблизно так само як і попередня ARIMA, важливо врахувати, що для перший прогнозованих елементів точність, судячи з графіку, э досить значною, адже модель добре відтворює шаблони реального трафіку.

2.1.6 Висновки з існуючих моделей

Виходячи з аналізу моделей, який було описано у попередньому підрозділі, видно що для прогнозування найкраще підходять моделі ARIMA та SARIMA, адже вони:

1. Мають одну з найкращих оцінок за заявленими критеріями оцінки

2. Мають підтримку нестаціонарних рядів. Це важливо, бо як було визначено під час аналізу в розділі 1 деякі наявні ряди можна віднести до нестаціонарних.

До мінусів цих моделей можна віднести

1. Набагато більший час навчання ніж у більш простих моделей. Через більшу кількість параметрів модель для одного ряду може навчатись декілька хвилини, на відміну від MA, яка навчається за декілька секунд. Це не є критичним мінусом, адже моделі не мають часто навчатись наново.

2. Можливість перенавчання. Це може бути значимим, тож під час розробки підходу про це варто пам’ятати.

Також варто відзначити, що ці моделі також мають набагато більше параметрів, та також, хоча це не продемонстровано під час аналізу, можуть працювати з трендом у рядах.

2.2 Проектування моделі прогнозування часових рядів

Знаючи всі фактори, зазначені у попередньому було прийнято рішення, для кожного часового ряду проводити моделювання ARIMA та SARIMA, проте, окрім стандартної оптимізації, яка використовує

автокореляцію та часткову автокореляцію, було прийнято рішення у разі якщо відносна середня помилка навченою моделі перевищує певну границю (за замовченням прийняту як 10%) проводиться оптимізація гіперпараметрів за допомогою пошуку за решіткою [25].

Пошук за решіткою – один із найпростіших методів оптимізації для моделі, суть якого полягає в переборі всіх можливих комбінацій для гіперпараметрів моделі. У даній роботі він можливий, адже всі гіперпараметри є дискретними величинами.

Спроектований алгоритм пошуку за решіткою працює за наступними кроками, приведеними нижче.

1. Створення всіх можливих комбінацій гіперпараметрів моделі. Для створення конфігурацій моделі використовуються дані про часовий ряд, завдяки яким можна знайти значення автокореляції та часткової автокореляції, значення яких використовуються для створення можливих значень p та q моделей ARIMA та SARIMA.

2. Так як процес навчання та оцінки кожної моделі є ізольованим, то для всіх конфігурацій створюються задачі, та також створюються паралельні процеси для підрахунку значень цих задач (кількість процесів залежить від кількості ядер центрального процесору).

3. Для оцінки якості моделі з певною конфігурацією виконується покрокова оптимізація, тобто модель навчається на тренувальних даних, отримує значення першого тестового кроку, записує його, та потім знову навчається на тренувальних даних плюс один крок з тестових, і так поки тестові дані не закінчяться. У якості мір оптимізації використовуються середня абсолютна помилка та середня абсолютна помилка в відсотках.

4. На останньому кроці всі оцінки порівнюються, виводиться результат найкращої моделі та її гіперпараметри.

Даний алгоритм пошуку за решіткою, можна використовувати для того щоб знайти кращу модель для часового ряду, проте його істотний мінус в тому що він витрачає великий час на оптимізацію, адже в ньому треба навчити купу моделей, саме тому він буде використовуватись, тільки якщо модель має неприйнятні показники помилки (які встановлюються при конфігурації). Схематичне представлення методу для створення моделі окремого часового ряду моделей зображено на рис. 2.9. Цей метод застосовується до кожного часового ряду, та потім отримана модель надає можливість отримати прогнозовані дані, які можна використати для вирішення задачі.

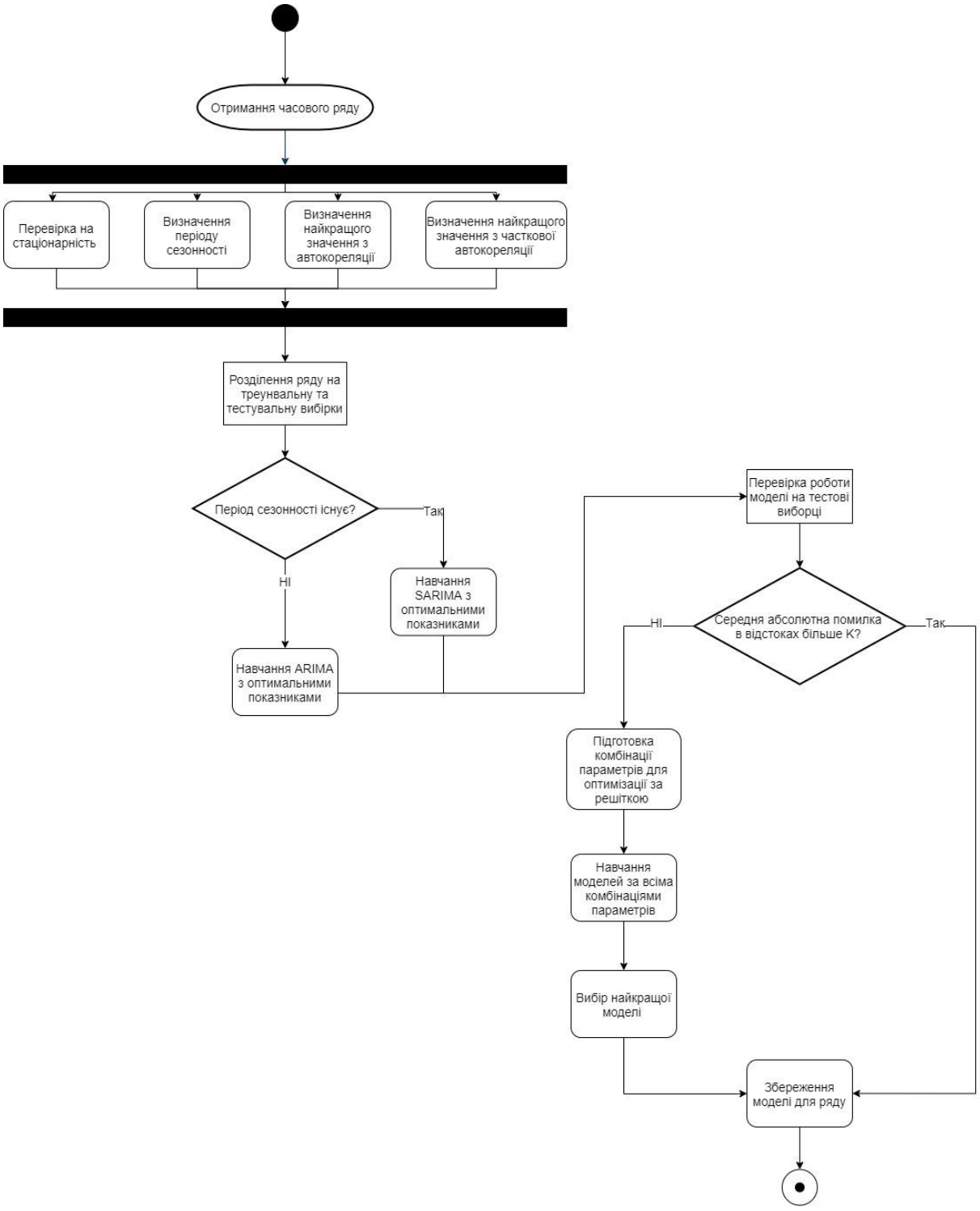


Рис. 2.9. Схематичне зображення методу для навчання моделей прогнозування часових рядів.

2.3 Проектування методу прийняття рішень на основі генетичного алгоритму

Як було зазначено у розділі 1 для прийняття рішень було вирішено використати генетичний алгоритм, тож було дещо модифіковано стандартну схему роботи даного алгоритму, додано параметр включення еліт [26] та також параметр відношення включених кампаній для початкової популяції. Ці параметри надають більшої гнучкості алгоритму. Загальна схема роботи генетичного алгоритму представлена нижче на рис. 2.10.

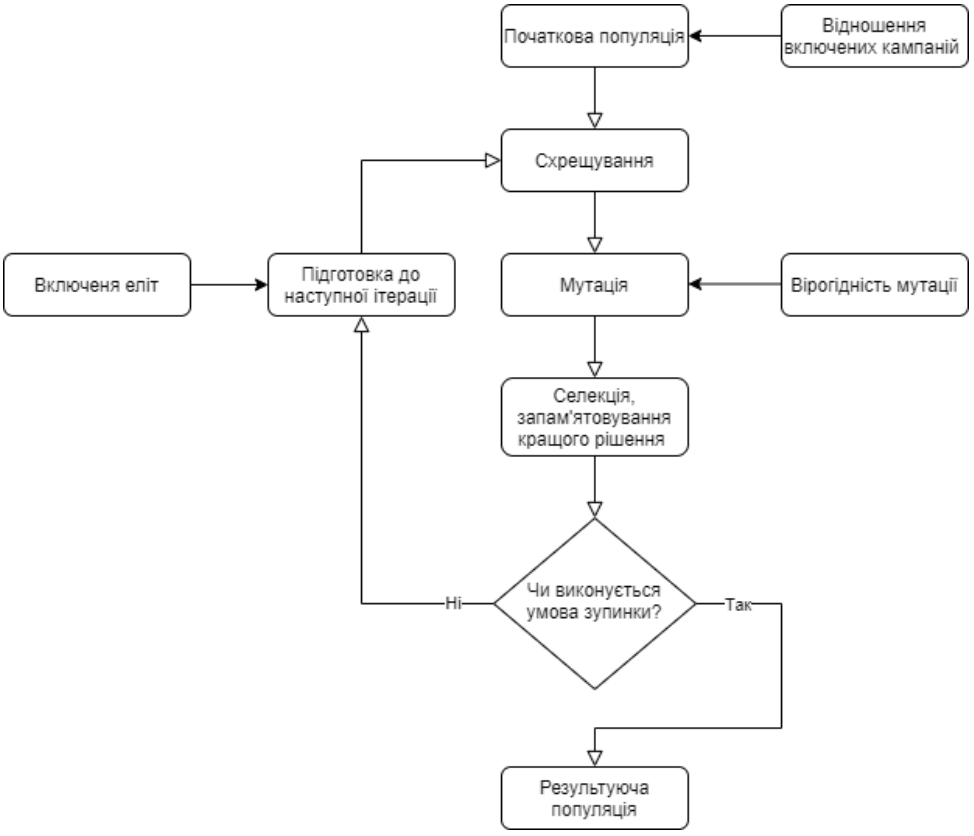


Рис. 2.10. Схема роботи генетичного алгоритму

Варто зазначити, що у якості рішення, яке повинен знайти алгоритм, виступає структура даних, яка містить план на певний період для всіх продуктів з позначкою за кожен день по яких саме рекламних каналах він рекламується. Тобто у результаті роботи алгоритму, наприклад для двох продуктів, по трьох рекламних каналах (a, b, c) та на рекламний період у один місяць буде отримано структуру даних плану.

Зважаючи на таку структуру функція пристосування рішення має наступну формулу:

$$\sum_{p=0}^{P} \min\left(\sum_{c=0}^{C} \sum_{d=0}^{D} ptraffic_{p,c,d} * price_{p,d} * ccoef_c, budget_p\right) \quad (2.7)$$

де $ptraffic_{p,c}$ – трафік продукту в певний день з певного рекламним каналом, $price_{p,d}$ – ціна одного переходу по продукту за певний день, $ccoef_c$ – коефіцієнт для масштабування за прогнозованим трафіком рекламного каналу, $budget_p$ – бюджет продукту на рекламний період та P, C, D – кількість продуктів, каналів, та днів у рекламному періоді відповідно.

А для реалізації операцій схрещування та мутації структуру даних, яка представляє рішення, треба серіалізувати в бітовий масив і далі всі операції вже відбуваються зрозумілим чином: для схрещування варто замінити частину бітів з одного масиву на частину з іншого, а для мутація відбувається проходження по масиву та 0 замінюється на 1 та навпаки з певним шансом. Після закінчення всіх операцій відбувається зворотна де серіалізація рішення у звичну структуру.

2.4 Висновки до розділу

1. Було визначено критерії оцінки моделей часових рядів, проаналізовано наявні рішення статистичних моделей часових рядів, оцінено їх та описано підхід до їх побудови.

2. Серед всіх моделей обрано найбільш підходящі зважаючи на результати аналізу та інформації, отриману під час аналізу наявних даних, отриману з попереднього розділу.

3. Визначено, що підбір параметрів моделей може не завжди видавати оптимальний результат (за середньою абсолютною помилкою у відсотках),

тож описано алгоритм оптимізації гіперпараметрів заснований на пошуку за решіткою.

4. Приведено схему методу для побудови моделі прогнозування окремого часового ряду, завдяки чому можна буде реалізувати метод побудови у наступних розділах.

5. Спроектовано метод прийняття рішень на основі генетичного алгоритму, визначено структуру результату планування рекламного періоду, та визначено функцію оцінки пристосування окремих рішень генетичного алгоритму.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗЦІЇ УПРАВЛІННЯ РЕКЛАМНОЮ КАМПАНІЄЮ

3.1 Робота з БД

3.1.1 Підготовка СУБД для роботи

Як було згадано раніше TimescaleDB – розширення (extension) для СУБД PostgreSQL і не є самостійним рішенням, тож для того щоб почати використовувати TimescaleDB потрібно спочатку для встановити основну СУБД. Процес був виконаний для ОС Windows 10, та саме встановлення є тривіальним, тож воно не буде висвітлити в даній роботі. Після встановлення СУБД було конфігуровано root користувача.

Наступним кроком було встановлення розширення TimescaleDB на локальну машину під Windows 10. Процес схожий на встановлення основної СУБД, за винятком того що треба було відстежити чи вимкнений сервер PostgreSQL та чи наявні всі потрібні бінарні файли у системній змінній PATH. Після виконання інсталяції розширення було виведено повідомлення: TimescaleDB installation completed successfully. Press ENTER/Return key to close...

Яке повідомляє про успішну установку. Під час встановлення також була проведена початкова конфігурація, за допомогою консольного інструменту timescaledb-tune, яка виявляє рекомендовані для даної конфігурації системи показники PostgreSQL та TimscaleDB і пропонує їх встановити.

Для завершення підготовки у СУБД була створена БД ads_data, та до неї підключено розширення TimescaldeDB, за допомогою SQL команди нижче, після виконання якої отримано повідомлення про успішне виконання скрипту:

```
CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE
```

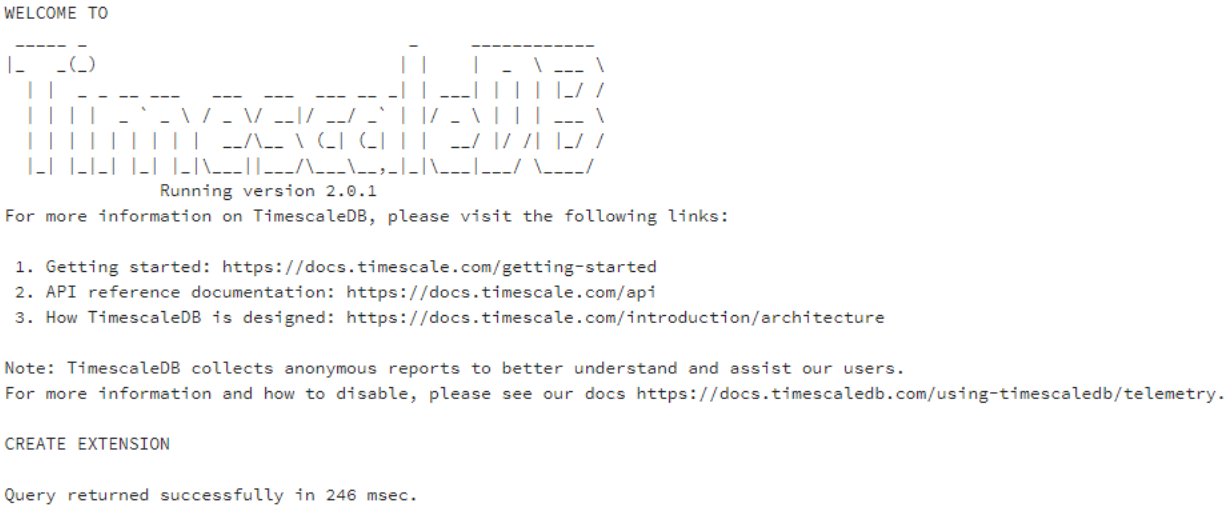
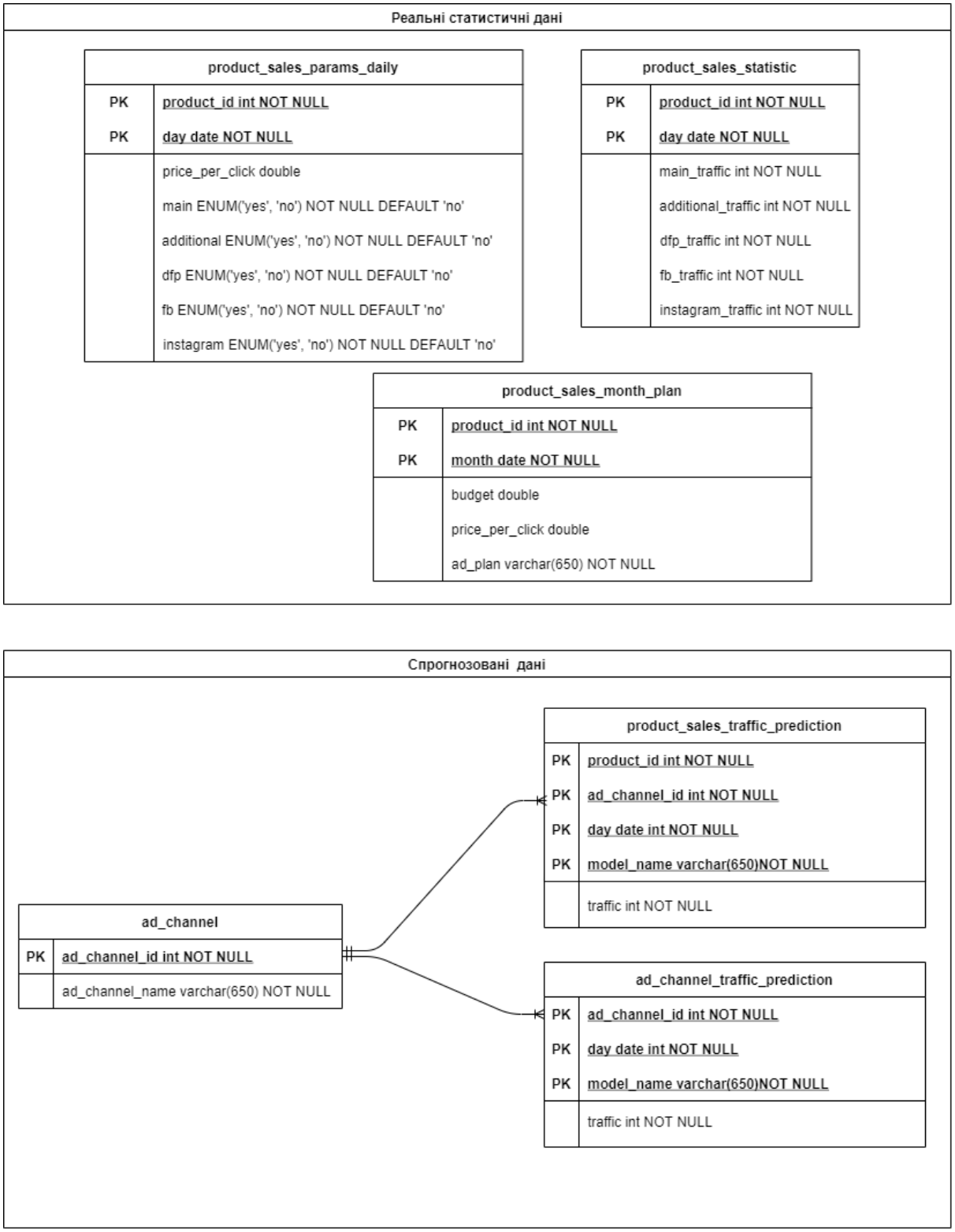


Рис. 3.1. Повідомлення про успішне додавання розширення до БД

3.1.2 Створення схеми даних у БД

Для того, щоб почати наповнювати БД даними, та працювати з ними спочатку було спроектовано схему, яка складається з таблиць, опису їх колонок та обмежень, що на них покладаються. При побудові схеми БД окрім функціональних потреб, які використані під час побудови та роботи предикативних моделей, було також ураховано, що дані можуть бути використані в умовах певних бізнес процесів, які можуть бути втілені на основі даної моделі ПЗ.



На рис. 3.2 зображено домени даних, їх таблиці, та зв’язок між ними,

Рис. 3.2. ER діаграма схеми даних БД

що використовуються у ПЗ. Опис наявних сутностей, та причину їх використання наведено у нижче:

product_sales_statistic – зберігає в собі часовий ряд з даними по трафіку за кожний день по кожному, наявному, продукту також розділені по окремими рекламним каналам. Так як таблиця зберігає дані за часом її, методами TimescaleDB перетворено у hypertable [27], з індексом над колонкою day та інтервал створення чанку виставлено в 7 днів.

product_sales_params_daily – зберігає в собі дані з налаштування по наявності реклами продукту в певний день в певному рекламному каналі, та також ціну за клік на конкретний день. Так як таблиця зберігає дані за часом її, також перетворено у hypertable, з індексом над колонкою day.

product_sales_month_plan – зберігає рекламні параметри певних продуктів на рекламний період (в даній системі буде розглядатись як календарний місяць), такі як бюджет, та останню ціну кліка на цей період. Також перетворено у hypertable, але проіндексовано за колонкою month.

ad_channel –зберігає список рекламних каналів, які приймають участь у процесі прогнозування (яка є частиною автоматизації управління) та їх id, і використовується у інших таблицях з домену прогнозування даних, для забезпечення цілісності.

product_sales_traffic_prediction – зберігає прогнозовані дані по трафіку продукту з певного рекламного каналу за день. Також, кожен запис має колонку model_name, в якій зазначається id моделі, що зробила прогноз. Перетворена у hypertable.

ad_channel_traffic_prediction – зберігає прогноз по сумарній кількості кліків за певним рекламним каналом, за певний день. В іншому таблиця має таку саму структуру як і product_sales_traffic_prediction.

3.2 Створення API ПЗ

Для того, щоб з даними з БД можливо було взаємодіяти за певними правилами, було обрано спосіб взаємодії через WEB API, адже це дозволяє

чітко визначити протокол взаємодії у даному ПЗ з даними, та забезпечити зрозумілість у роботі з додатком.

У якості підходу до проектування API було обрано REST [28], через те що це дозволить значно спростити розробку API, не зважати на сурові вимоги, наявні у інших підходах, при цьому зберегти зрозумілість та легкість під час клієнт – серверної взаємодії та забезпечити масштабованість при додаванні нових ресурсів у систему (зараз, як видно з схеми даних БД, їх невелика кількість).

Основною мовою програмування для розробки API було обрано Python, так як ця мова, по-перше, має розвинуті інструменти для створення back end додатків, та є зручною для даного типу задач, а, по-друге, також використовується для розробки machine learning алгоритмів, що потрібні для прогнозування часових рядів, тож є можливість розробити весь проект за допомогою однієї мови програмування, яку варто використати.

У якості основного фреймворку для розробки API було обрано FastAPI [29], через такі його особливості:

1. Має гарний, інтуїтивний дизайн, що прискорює розробку.

2. Націлений саме на REST API.

3. Легкий у вивченні.

4. Наявність вбудованих інструментів для автоматичної документації коду.

Першим кроком на створені API була розробка структури проекту з розділенням його за файлами/пакетами кожен з яких буде відповідати за свою частину роботи API. У результаті було створено наступні модулі:

database – відповідає за підключення до БД, та надає його іншим частинам проекту.

models – зберігає у собі опис всіх моделей БД, які будуть використовуватись в проекті.

schemas – опис схем даних у коді, які використовуються саме для прийому або відправки даних через кінцеві точки API (на відміну від моделей, які використовуються виключно для зручної роботи з БД).

crud – у цьому модулі зібрані методи для стандартних операції з моделями даних (створення, зчитування, зміна, видалення) у яких також використовуються схеми даних.

main – модуль у якому створюються всі кінцеві точки API, та який використовує для цього розроблені моделі, схеми та методи інших модулів.

Як було зазначено вище, FastAPI має можливість автоматичного документування коду тож після розробки основних кінцевих точок API було переглянуто роботу цієї особливості фреймворку за шляхом /docs.

3.3 Реалізація ПЗ для створення рекламного плану на основі генетичного алгоритму

3.3.1 Опис наявних структур даних

Для реалізації генетичного алгоритму було розроблено кілька класів у яких знаходяться потрібні структури даних разом з корисними методами для роботи з ними. Діаграма класів представлена на рис. 3.3.

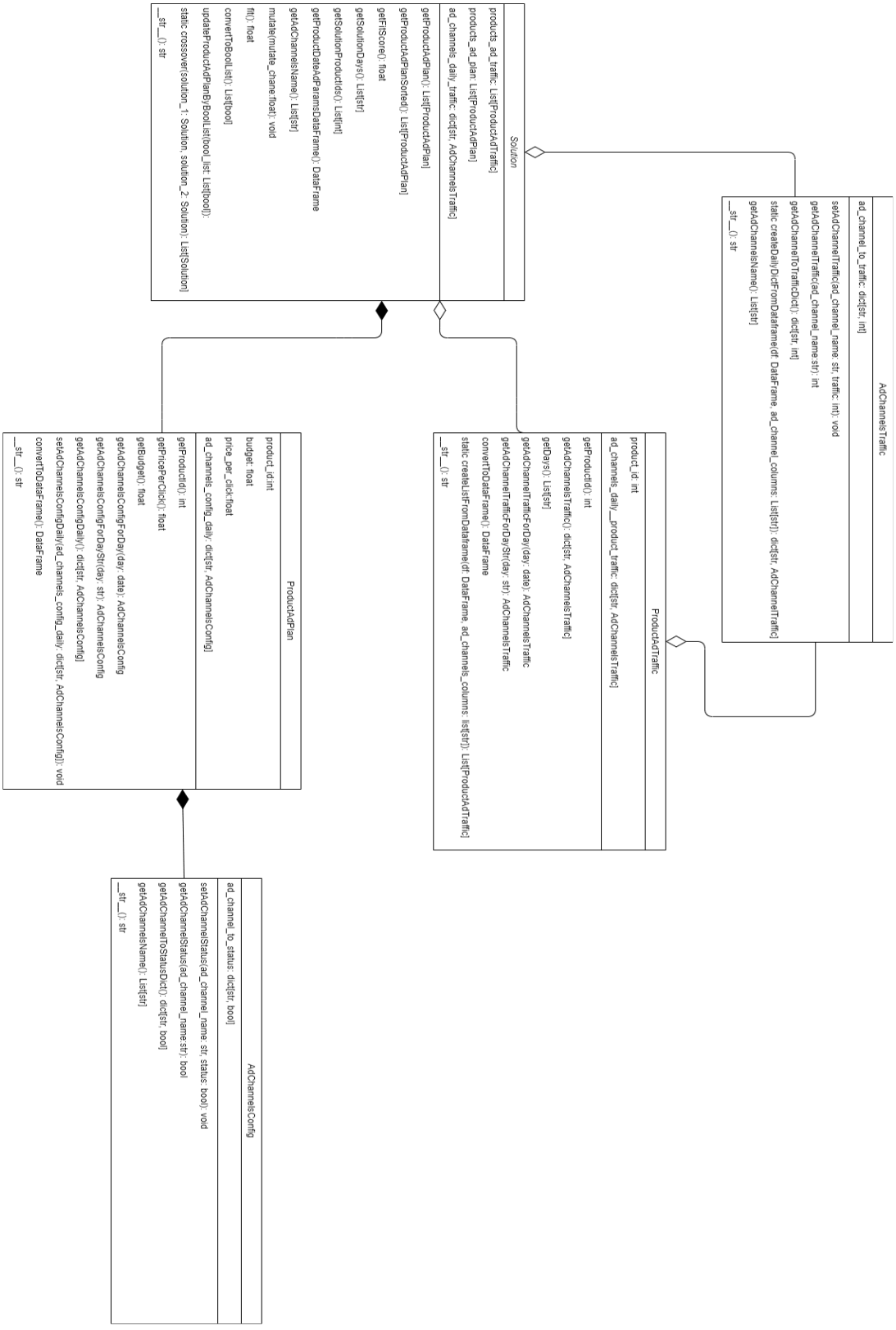


Рис. 3.3. Діаграма класів для генетичного алгоритму

Як видно з діаграми більшість класів агрегуються певним чином з основним класом – Solution, адже він насправді є єдиним типом з яким

працює генетичний алгоритм, а інші класи є допоміжними і створені задля полегшення роботи з певними типами даних. Детальну інформацію по класам приведено нижче.

1. Клас AdChannelsTraffic. Цей клас являє собою обгортку над словником у якому ключами виступають ідентифікатори рекламних каналів, а значеннями – трафік по певним каналам. Опис методів приведено у табл. 3.1.

Таблиця 3.1

Опис методів класу AdChannelsTraffic

Назва методу	Опис
setAdChannelTraffic	Присвоєння певному рекламному каналу значення трафіку
getAdChannelTraffic	Отримання значення трафіку по певному рекламному каналу
getAdChannelToTrafficDict	Отримання словника відповідності рекламних каналів до значень трафіку
createDailyDictFromDataframe	Створення екземпляру класу з наявного DataFrame
getAdChannelsName	Отримання ідентифікаторів всіх рекламних каналів
__str__	Приведення екземпляру до рядка

Цей клас використовується у класах Solution та ProductAdTraffic, адже дана структура зручна водночас і для презентації сумарного трафіку за певним каналом за рекламний період, так і для презентації трафіку по продукту по рекламному каналу.

2. Клас ProductAdTrafic. Ця структура даних зберігає в собі дані про трафік для певного продукту за рекламний період, та має методи для роботи з цими даними. Використання цього класу на діаграмі зображено агрегацією, адже насправді його екземпляри є спільними для всіх

екземплярів Solution під час роботи алгоритму (так як дані для всіх однакові), тож вони не знищуються після знищення окремого Solution, тож це саме не композиція а агрегація. Також екземпляри можуть використовуватись окремо від Solution. У табл. 3.2 описано методи, якими оперує даний клас.

Таблиця 3.2.

Опис методів класу ProductAdTrafic

Назва методу	Опис
getProductId	Отримання ідентифікатору продукту
getAdChannelsTraffic	Отримання словника, у якому ключи – це дати, а значення – екземпляри AdChannelsTraffic
getDays	Отримання дат, по яким екземпляр має інформацію по трафіку
getAdChannelTrafficForDay	Отримання екземпляру AdChannelsTraffic для певної дати
getAdChannelConfigForDayStr	Отримання екземпляру AdChannelsTraffic для певної дати, заданої рядком
convertToDataFrame	Перетворення екземпляру класу на DataFrame
createListFromDataframe	Створення списку ProductAdTrafic з DataFrame
__str__	Перетворення екземпляру у рядок

3. Клас AdChannelsConfig. Цей клас схожий за структурою на AdChannelsTraffic, розглянутому вище, але замість трафіку зберігає статус рекламного каналу: чи включено по ньому рекламу чи ні. Використовується дана структура тільки у класі ProductAdPlan, і поза ним немає сенсу, тож зв’язок між ними – композиція. У табл. 3.3 наведено опис методів даного класу.

Таблиця 3.3

Опис методів класу AdChannelsConfig

Назва методу	Опис
setAdChannelStatus	Присвоєння певному рекламному каналу значення трафіку
getAdChannelStatus	Отримання значення трафіку по певному рекламному каналу
getAdChannelToStatusDict	Отримання словника відповідності рекламних каналів до значень трафіку
getAdChannelsName	Отримання ідентифікаторів всіх рекламних каналів
__str__	Перетворення екземпляру у рядок

4. Клас ProductAdPlan. Дана структура даних зберігає у собі дані про різні рекламні параметри продукту, такі як бюджет, ціна за перехід з реклами, та рекламний план на певний період. Саме цей план і буде встановлено за допомогою генетичного алгоритму для кожного продукту, що є у рішенні. У якості такого рекламного плану виступає словник з відповідністю дати до екземпляру класу AdChannelsConfig. Опис методів класу надано у табл. 3.4.

Таблиця 3.4.

Опис методів класу ProductAdPlan

Назва методу	Опис
initFromProductAdParams	Фабричний метод, що використовується для створення ProductAdPlan з класу ProductAdPlan, який являє собою DTO для бюджету та ціни продукту
getProductId	Отримання ідентифікатору продукту
getBudget	Отримання бюджету продукту
getPricePerClick	Отримання ціни за рекламний перехід

getAdChannelsConfigForDay	Отримання рекламного плану на певний день

Продовження табл. 3.4

getAdChannelsConfigForDayStr	Отримання рекламного плану на певний день, заданий рядком
getAdChannelsConfigDaily	Отримання рекламного плану по днях, у вигляді словника
.setAdChannelsConfigDaily	Присвоєння рекламного плану
convertToDataFrame	Перетворення екземпляру на DataFrame
__str__	Перетворення екземпляру у рядок

5. Клас Solution. Екземпляри цього класу є окремими рішення генетичного алгоритму, серед яких якраз і шукається найкраще можливе. Кожне рішення має в собі посилання на список ProductAdTraffic, для того щоб знати трафік по продуктам, словник відповідності днів до екземплярів AdChannelsTraffic, щоб мати інформацію про сумарний трафік по певному рекламному каналу за певний день, та також список ProductAdPlan, які мають в собі певний рекламний план, який якраз і буде різним для двох різних рішень. Тобто цей клас агрегує у собі всю інформацію та є основним для роботи генетичного алгоритму, адже всі операції відбуваються саме над його екземплярами. Основні дії над рішеннями, які потрібні генетичному алгоритму також знаходяться у цьому класі, для подальшого полегшення реалізації самого скрипту алгоритму. Опис методів класу надано в табл. 3.5.

Таблиця 3.5

Опис методів класу Solution

Назва методу	Опис
getProductsAdPlan	Отримання списку рекламних планів для продуктів
getProductsAdPlanSorted	Отримання списку рекламних планів для продуктів, відсортованих за ідентифікатором продукту
getFitScore	Отримання оцінки пристосованості рішення
getSolutionDays	Отримання списку днів рекламного плану
getSolutionProducts	Отримання списку ідентифікаторів рекламних продуктів
getProductDateAdParamsDataFrame	Отримання даних про продукти в одному DataFrame
getAdChannelsName	Отримання списку імен рекламних каналів
mutate	Мутація рішення, з заданою вірогідністю
fit	Підрахунок оцінки пристосування
__convertToBoolList	Перетворення рішення у список bool значень
updateProductAdPlanByBoolList	Метод для того, щоб задати рекламний план за списком bool значень
crossover	Схрещування двох рішень
__str__	Перетворення рішення у рядок

3.3.2 Реалізація генетичного алгоритму

1. Першим кроком алгоритму, як продемонстровано на рис. 2.10, є створення початкової популяції, що відбувається за допомогою функції initRandomSolutions, яка приймає на вхід дані по трафіку, кількість осіб в популяції та спеціальний параметр enable_rate, який відповідає відношенню кількості включених каналів за всі дні, до загальної кількості каналів, що в певних випадках може допомогти швидше знайти більш підходяще рішення.

2. Після створення початкової популяції створюється змінна best_solution, у якій буде зберігатись краще рішення та також запускається цикл на задану кількість ітерацій, у якому відбуваються всі подальші кроки алгоритму.

3. На початку роботи циклу відбуваються селекція та відбір, тобто для всіх рішень знаходиться їх оцінювальне значення, вони за ним сортуються за спаданням, і перше рішення порівнюється з наявний кращим, і якщо воно краще, то стає новим кращим рішенням.

4. Починається формування рішень для нового покоління, і першим кроком для цього є схрещування, для якого, спочатку кожному наявному рішенню ставиться у відповідність його вірогідність стати пращуром (ця вірогідність відповідає значенню пристосування) та за цією вірогідністю обирається пара різних батьків, які потім схрещуються, даючи два нових рішення. Це відбувається поки кількість нової популяції не буде достатньою.

5. Для кожного рішення з новою популяції відбувається мутація, з заданою вірогідністю.

6. Нова популяція заміщує стару (з зберіганням деяких кращих представників старою популяції, якщо заданий параметр elitism).

7. Після закінчення роботи циклу повертається значення кращого рішення серед усіх поколінь та також наявний звіт роботи у спеціальному файлі, в якому зазначені кращі рішення на кожній ітерації, їх оцінка та опис.

У результаті тестування алгоритму для заданих даних було знайдено рішення, яка краще здатне трохи краще розподілити трафік між продуктами, таким чином щоб підняти загальну прибутковість рекламної сітки на кілька відсотків (в рамках тієї ж моделі) ніж рішення у якому реклама завжди ввімкнена для всіх продуктів по всім рекламним каналам.

3.4 Висновки до розділу

1. Було створено БД, яка визначена під час аналізу у розділі 1, підготовлено її до роботи та розроблено схему даних у ній, завдяки який ПЗ здатне зберігати та працювати з даними.

2. Створено API програмного забезпечення, завдяки якому користувач зможе додавати та переглядати наявні дані, які використовуються під час роботи спроектованого методу автоматизації управління рекламними кампаніями.

3. Описано структури даних, які використовує алгоритм створення рекламного плану, та реалізовано метод на основі генетичного алгоритму, спроектований у попередніх розділах . У результаті тестування даного методу виявлено, що план, створений ним, є більш ефективним, ніж той, у якому реклама включена для кожного продукту на кожний день рекламного періоду.

В додатку А надано опис архітектури ПЗ у вигляді Software Architecture Document, в якому приведено фактори, які вплинули на створення архітектури, діаграми до не та її особливості.

ВИСНОВКИ

Результати роботи показали, що застосування методів машинного навчання для прогнозування часових рядів разом з генетичним алгоритмом може бути доцільним для побудови програмного забезпечення автоматизації управління рекламою у умовах ресурсних обмежень. Одне з рішень реалізації такого ПЗ було приведене у роботі та також були розглянуті альтернативи для кожної з задач:

1. Побудова інфраструктури для роботи такого ПЗ. Було проаналізовано наявні рішення для TSDB, обґрунтовано вибір TimescaleDB, застосовано дану СУБД при реалізації ПЗ.
2. Прогнозування рекламного трафіку. Напочатку було проаналізована наявні дані, на основі цього приведені та оцінені наявні статистичні моделі для роботи з часовими рядами та спроектовано модель для прогнозування окремих часових рядів в рамках загального методу.
3. Розробка рекламного плану. Було спроектовано метод для планування реклами на певний період, враховуючи обмеження та наведено приклад реалізацій генетичного алгоритму.

Виконаний аналіз показує, що може існувати багато альтернатив до кожної частини проблеми, та немає універсального способу побудови інфраструктури, прогнозування трафіку та розробки рекламного плану, адже всі ці частини залежать від сторонніх умов: наявних даних, їх природу, бізнес умови та області, у якому буде працювати ПЗ.

Пропонована реалізація ПЗ може застосуватись для автоматизації управління рекламою, в умовах різних рекламних бюджетів кожного продукту та обмеженої кількості трафіку за кожним рекламним каналом, та завдяки ній не потрібно залучати людину до прийняття рішень про рекламу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Luis Miralles-Pechuána, Hiram Ponceb, Lourdes Martínez-Villaseñorb
.
A novel methodology for optimizing display advertising campaigns using genetic algorithms. [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/-science/article/abs/pii/S1567422317300935

2. Maryam Karimzadehgan, Wei Li, Ruofei Zhang, Jianchang Mao. A stochastic learning-to-rank algorithm and its application to contextual advertising. – [Електронний ресурс] – Режим доступу: https://dl.acm.org/doi/abs/10.1145/1963405.1963460.

3. Громенко В.В., Шевченко В.Л. (Hromenko V.V., Shevchenko V.L.) Автоматизація управління рекламною кампанією в умовах ресурсних обмежень MSTIoE 2021-8. 8-ма Східно-Европейська конференція “Математичні та програмні технології Internet of Everything” (14.05.2021, Київ). Зб.матер., КНУ ім.Т.Шевченка с.17-18 pp.17-18.

4. Громенко В.В., Шевченко В.Л. (Hromenko V.V., Shevchenko V.L.) Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень MSTIoE 2021-8. 8-ма Східно-Европейська конференція “Математичні та програмні технології Internet of Everything” (14.05.2021, Київ). Зб.матер., КНУ ім.Т.Шевченка с.18-20 pp.18-20.

5. Time series database - Wikipedia, the free encyclopedia – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Time_series_database

6. Trend of Time Series DBMS Popularity. DB-Engines. – [Електронний ресурс] – Режим доступу: https://db-engines.com/en/ranking_trend/time+series+dbms

7.InfluxDB. - Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/InfluxDB

8. NoSQL. - Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/NoSQL

9. Kdb+. Національна бібліотека ім. Н. Е. Баумана. – [Електронний ресурс] – Режим доступу: https://ru.bmstu.wiki/Kdb%2B

10. Prometheus Docs. Prometheus Authors. – [Електронний ресурс] – Режим доступу: https://prometheus.io/docs/introduction/overview/

11. Graphite FAQ. The Graphite Project Revision 9ee195c3. – [Електронний ресурс] – Режим доступу: https://graphite.readthedocs.io/en/latest/faq.html

12. InfluxDB vs. Graphite for Time Series Data & Metrics Benchmark. Chris Churilo. – [Електронний ресурс] – Режим доступу: https://www.influxdata.com/blog/influxdb-outperforms-graphite-in-time-series-data-metrics-benchmark/

13.TimescaleDb Docs. Timescale, Inc. – [Електронний ресурс] – Режим доступу: https://docs.timescale.com/timescaledb/latest/

14. DB-engines. System Properties Comparison InfluxDB vs. TimescaleDB. DB-Engines. – [Електронний ресурс] – Режим доступу до ресурсу: https://db-engines.com/en/system/InfluxDB%3BTimescaleDB

15. TimescaleDB vs. InfluxDB: Purpose built differently for time-series data. Timescale, Inc. – [Електронний ресурс] – Режим доступу до ресурсу: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/

16. Augmented Dickey–Fuller test. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Augmented_Dickey%E2%80%93Fuller_test

17. Autoregressive integrated moving average. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

18. Ljung–Box test – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Ljung%E2%80%93Box_test

19. Coefficient of determination. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Coefficient_of_determination

20. Mean absolute error. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Mean_absolute_error

21. Moving-average model. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Moving-average_model

22. Auto-regression Analysis (AR). – Paul Bourke. – [Електронний ресурс] – Режим доступу: http://paulbourke.net/miscellaneous/ar/.

23. Autoregressive–moving-average model. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model

24. Seasonal ARIMA models. Applied Time Series Analysis Penn State Eberly college of science. – [Електронний ресурс] – Режим доступу: https://online.stat.psu.edu/stat510/lesson/4/4.1

25. Hyperparameter optimization. – Wikipedia, the free encyclopedia. – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search

26. Improvements in genetic algorithms. J.A. Vasconcelos; J.A. Ramirez; R.H.C. Takahashi; R.R. Saldanha. – [Електронний ресурс] – Режим доступу: https://ieeexplore.ieee.org/abstract/document/952626

27. Benefits of Hypertables and Chunks. Timescale, Inc. – [Електронний ресурс] – Режим доступу: https://docs.timescale.com/timescaledb/latest/overview/core-concepts/hypertables-and-chunks/##benefits-of-hypertables-and-chunks

28. Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. – [Електронний ресурс] – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

29. FastAPI. Sebastián Ramírez. – [Електронний ресурс] – Режим доступу: https://fastapi.tiangolo.com/.

ДОДАТКИ

Додаток А

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень

Software Architecture Document (SAD) Version 1.0

Автор: Громенко В.В.

Версія 1.0

ІСТОРІЯ РЕВИЗІЇ

НОМЕР ДОКУМЕНТУ	ВИПУСК /ПЕРЕГЛЯД	ДАТА ВИПУСКУ / ПЕРЕГЛЯДУ
1	**v1.0**	**2021/05/30**

ЗМІСТ

SOFTWARE ARCHITECTURE DOCUMENT

1. Вступ

Інтернет магазини, агрегатори оголошень соціальні мережі постійно адаптують свої технології для ефективної просування різноманітних продуктів через свої рекламні сітки, проте чим більше продуктів та каналів залучено у одну рекламну кампанію – тим важче спланувати як і коли саме показувати той чи інший продукт та у якому саме каналі, тож виникає задача автоматизації процес планування реклами.

Саме вирішення цієї задачі і є основною ціллю даного ПЗ, яке комбінує в собі статистичні методи для прогнозування часових рядів, метод прийняття рішень, заснований на генетичному алгоритмі, та REST API і БД направлене на роботу з часовими рядами.

Цей документ надає огляд високого рівня та пояснює архітектуру системи для автоматизації рекламних кампаній в умовах ресурсних обмежень та також визначає цілі архітектури, use cases, що підтримуються системою, архітектурні стилі та компоненти, які були обрані для реалізації.

1.1 Ціль

Software Architecture Document (SAD) надає опис архітектури системи для автоматизації рекламних кампаній в умовах ресурсних обмежень. Він представляє ряд різних архітектурних поглядів (до яких входить use case, Process, Deployment, Implementation), щоб зобразити різні аспекти даної системи.

1.1.1 Визначення проблеми

При наявності рекламної сітки на кілька рекламних каналів варто визначати як саме користуватись її ресурсами, якими є трафік з кожного каналу і при зростанні кількості продуктів дана проблема стає досить складної для вирішення її людиною, особливо якщо врахувати, що окрім обмеження трафіку також у кожного продукту існує власний рекламний бюджет, при перевищенні якого реклама більше не приносить користі. Основною задачею у даній роботі є зробити систему, яка буде автоматично створювати план на певний рекламний період.

1.2 Сфера застосування

Цей документ описує різні аспекти проектування системи, які вважаються архітектурно важливими, тож розробникам які мають працювати з даної системою автоматизації, варто ознайомитись з даним документом, для того щоб мати розуміння про наявні компоненти, варіанти використання та рішення, які були впроваджені в ПЗ.

1.3. Визначення, акроніми та абревіатури

• API – прикладний програмний інтерфейс.

• PostgreSQL- система управління реляційними базами даних (СУБД).

• TimescaleDB - це реляційна база даних для часових рядів, що є розширення PostgreSQL.

• SAD - Software Architecture Document

• UML - уніфікована мова моделювання

• TSDB – Time series database.

1.5. Огляд документу

Для повного документування всіх частин архітектури системи SAD містить наступні підрозділи.

Розділ 2 – описує використання різних шарів архітектури

Розділ 3 – описує архітектурні цілі та обмеження системи.

Розділ 4 – описує найважливіші use case системи.

Розділ 5 – описує логічний вигляд системи, включаючи визначення інтерфейсу та операцій.

Розділ 6 – описує шар даних системи.

Розділ 7 – особливості та проблеми архітектури.

2. Опис архітектурних шарів.

1. Use Case шар.

Аудиторія: усі зацікавлені сторони системи, включаючи кінцевих споживачів результатів роботи системи.

Область: описує набір сценаріїв та / або випадків використання, які представляють якусь важливу центральну функціональність системи.

Пов’язані артефакти: Модель use case та опис до неї.

2. Шар логіки.

Аудиторія: розробники, зайняті в дизайні API.

Область: відповідає за дотримання функціональні вимог та бізнес-логікі системи.

Пов’язані артефакти: Діаграма компонентів логіки.

3. Шар даних

Аудиторія: розробники, адміністратори баз даних

Область: описує архітектурні елементи в моделі даних, а також те, як дані переміщуються у системі.

Пов’язані артефакти: модель даних, data flow діаграма.

3. Архітектурні цілі та обмеження

1. Реалізована система призначена як доказ концепції, для того щоб продемонструвати життєздатність методу автоматизації реклами. Продовження розробки системи до більш готової до використання у реальних проектах вимагає подальшої розробки та проектування нових архітектурних рішень. Тому даний документ є корисним майбутнім архітекторам рішень та розробникам.

2. Система реалізована на мові програмування Python, перш за все через те що у ній наявні ML компоненти, які згодом можуть ставати складнішими і будуть вимагати потужної інфраструктури, яка присутня у даній мові. Також вона має значні показники популярності та стабільності використання у бекенд розробці, що дає можливість використати її на всю систему.

3. У системі дані можна поділити на ті, що відносяться до бізнес-логіки та часові ряди, саме тому якості основної БД використовується PostgreSQL плюс розширення TimescaleDB, яке може покрити одразу дві ці частини даних, тож не треба буде використовувати спеціалізовано рішення для часових рядів і задарма вводити ще одну БД, яка ускладнить архітектуру.

4. З системою можна взаємодіяти за допомогою WEB API, яке клієнт викликає з своїх сервісів, які не стосуються даної системи, через те що

клієнт сам має заповнювати даними систему, та сам має вирішувати що робити з рекламним планом, який надає основний метод системи.

5. Немає особливих обмежень, що стосуються продуктивності системи. Передбачається що система може відповідати на запити з певною затримкою, а основна робота методу створення рекламного плану відведена на фоновий процес, про який користувач не знає.

4. Шар Use Case

Список випадків використання, які представляють основні функції системи для автоматизації перебігу рекламної кампанії:

1. Переглянути параметри для рекламного періоду

2. Додати параметри для рекламного періоду

3. Видалити параметри для рекламного періоду

4. Додати дані про трафік продукту за певний день

5. Переглянути дані по певному трафіку продукту.

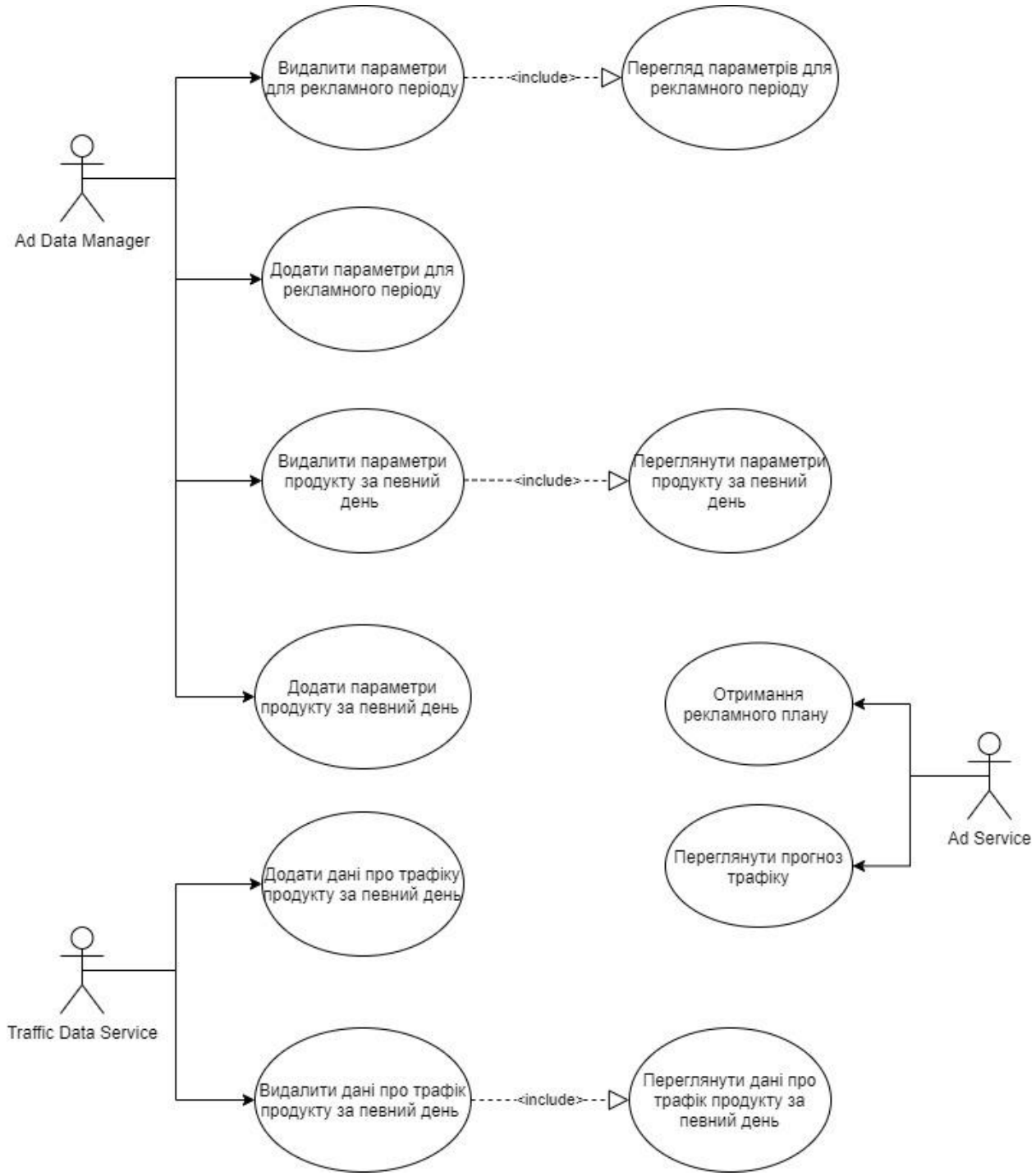
6. Додати параметри продукту на певний день.

7. Видалити параметри продукту на певний день

8. Переглянути параметри продукту на певний день

9. Отримати рекламний план на рекламний період.

10. Переглянути прогноз трафіку



Use case діаграма.

4.1 Користувачі системи

У системи є три типи користувачів:

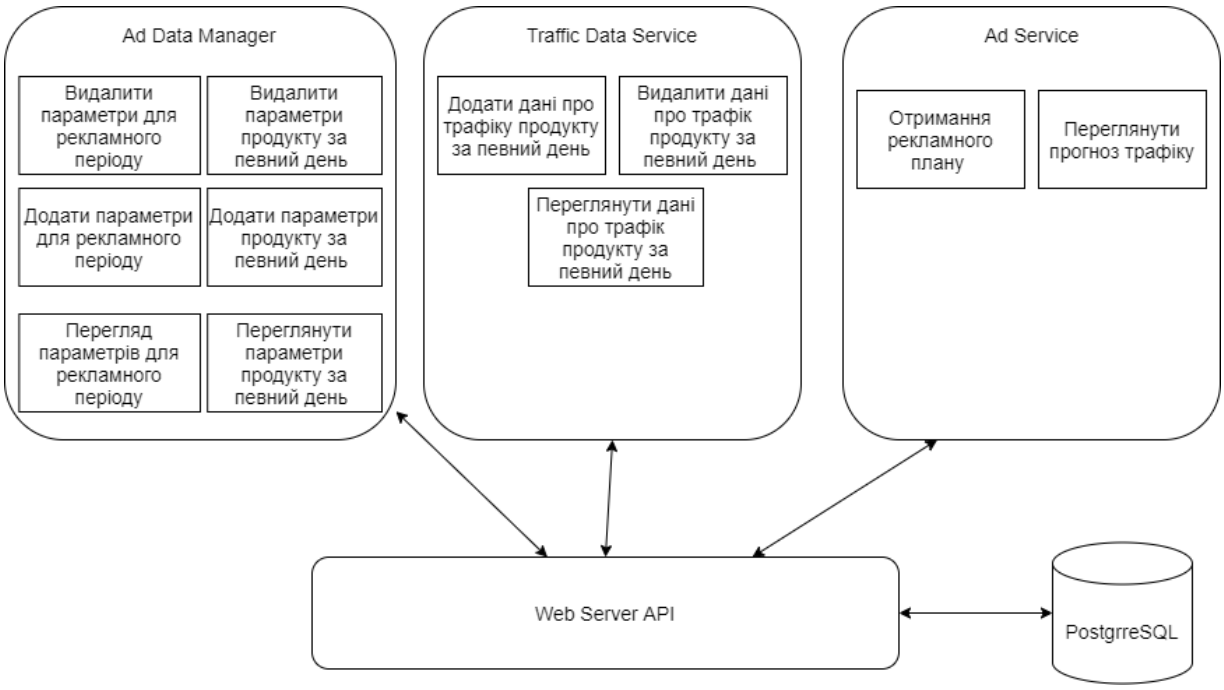
1. Traffic Data Service. Сервіс, який наповнює системи даними по трафіку. Це може бути як real-time программа, яка має інтеграцію з, наприклад, Google Analytics та пересилає дані з нього в API, або ж сервіс який в певний момент часу записує купу даних блоками.

2. Ad Data Manager. Сервіс, який наповнює систему рекламними параметрами. Це може бути адміністративна панель для команди продажів, через яку вони вносять потрібні бюджети та ціну на певний рекламний період

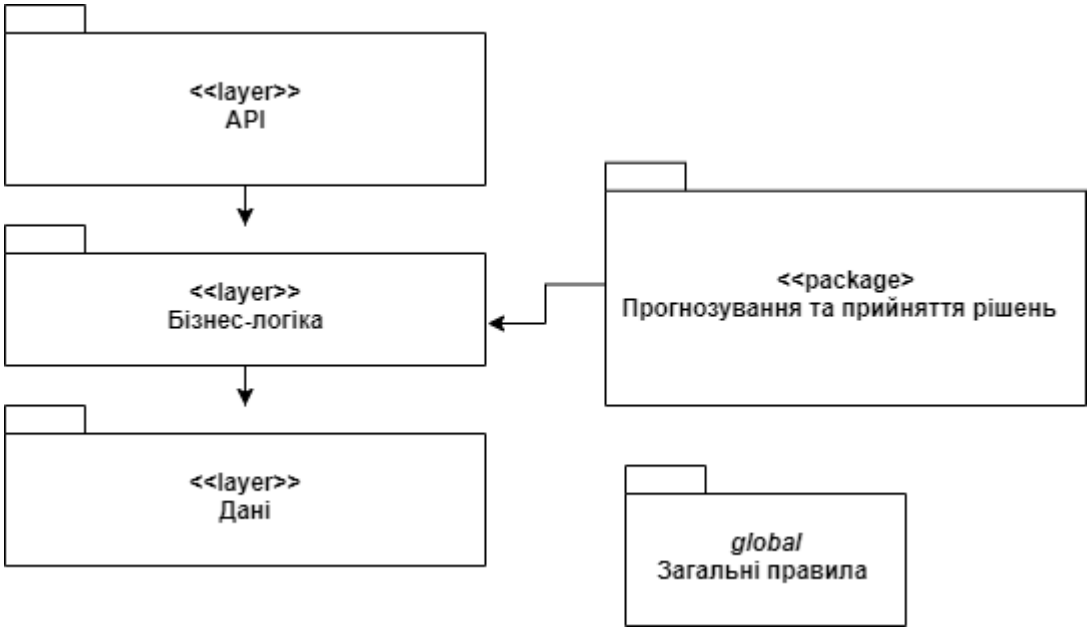
3. Ad Service. Сервіс для отримання рекламного плану. Це має бути ПЗ, яке відповідає за показ реклами і може керувати різними рекламними каналами, тож для нього рекламний план буде конфігурацією.

4.2 Реалізація use case

Діаграма реалізації функціональних можливостей, описує, як високо рівневі елементи архітектури забезпечують бажану функціональність системи.



5. Шар логіки

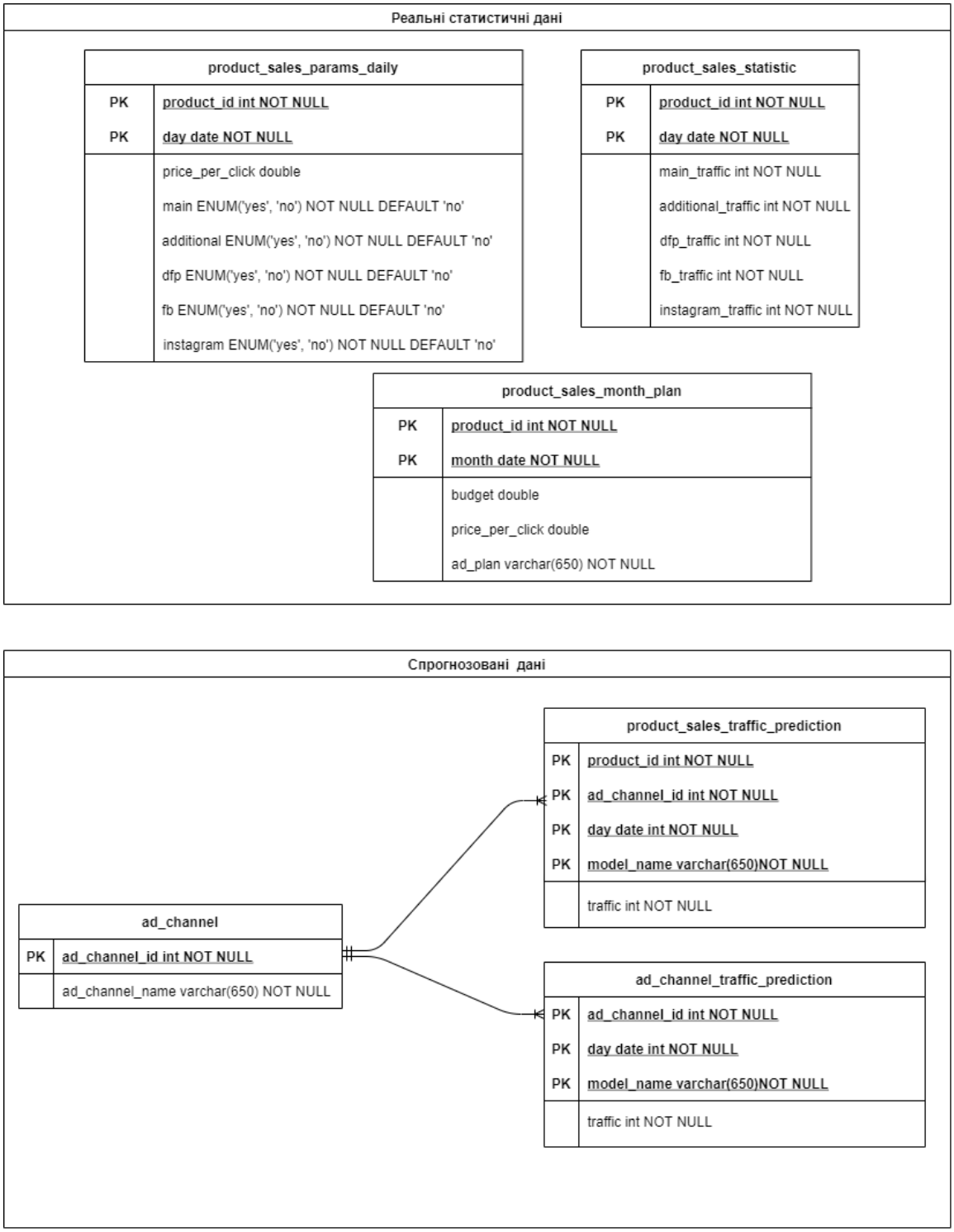


Вище зображено діаграму підшарів та пакетів, яка має стандартний поділ, на 3 шари, проте у якості шару, який відповідає за взаємодію з користувачем використовується на UI, а API, так як система розрахована на роботу у якості окремого сервісу для інших програм. Також у підшар бізнес логіки вхожить пакет для прогнозування та прийняття рішень, який виводиться окремо у системі, так як не має зв’язку з API, але використовує та впливає на підшар з даними. Також існує глобальний пакет з правилами для системи.

6. Шар даних

Представлення даних представляє значну частину сервісу автоматизації рекламних кампаній. Саме через те що істотна частина даних є часовими рядами було обрана відповідна конфігурація PostgreSQL з встановленим розширенням TimescaleDB, яке має спеціальні структури даних, які називаються гіпертаблиці, що можуть оптимізувати запити до часових рядів та їх збереження.

Окрім використання TimscaleDB, у шарі даних також варто звернути увагу на схему даних на рисунку нижче, адже там зображено зв’язок між різними таблицями та колонки, які в них знаходяться.



7. Особливості та проблеми

1. ПЗ розраховано на роботу з невеликим середніми рекламними сітками (не більше кількох тисяч продуктів), через обмеження компоненту прогнозування та прийняття рішень (вони є досить ресурсозатратними).

2. Тяжкі ML задачі, мають виконуватись на фоні, за певним розпорядком, адже моделі прийняття рішень та прогнозування неможливо навчати під час відповідь на запит користувача.

3. Як було зазначено вище ПЗ має бути скоріш доказом концепцію про можливість застосування методу, заснованому на поєднанні статистичних моделей та генетичного алгоритму, для автоматизації перебігу рекламної кампанії, тож не була детально розглянута проблема розгортання ПЗ.

Додаток Б

ЗАТВЕРДЖУЮ
Директор ТОВ «ІТЕЛ УКРАЇНА»
Огородник Л.В.
_____
травня 2021 року

АКТ
впровадження результатів дослідження
студента Київського національного університету імені Тараса Шевченка
Громенко Владислава Віталійовича
на тему:
"Програмне забезпечення автоматизації управління рекламною кампанією в умовах ресурсних обмежень"

Комісія у складі:
голова комісії – директор Огородник Людмила Василівна
члени комісії:
– Інженер з технічного нагляду Мусієнко Олександр Іванович
– Помічник керівника малого підприємства без апарату управління Смачило Соломія Андріївна

Встановила та цим актом засвідчує, що нижчеперелічені результати досліджень:

1. Запропонований метод для автоматизації перебігу рекламних кампаній одночасно кількох продуктів.
2. Розроблена система знижує залученість людини у процес контролю перебігу рекламної кампанії.
3. Удосконалено методи планування рекламних кампаній.

впроваджені при розробці програмної системи для управління рекламою.

Голова комісії Огородник Л. В.

Члени комісії Мусієнко О. І.

Смачило С. А.