НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 519.688

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»
зі спеціальності 121 Інженерія програмного забезпечення на тему:
«Метод та програмне забезпечення для передбачення успішності волонтерських зборів»

Виконав:
студент ІІ курсу, групи КП-21мп
Лавріненко Вадим Володимирович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент
Олещенко Любов Михайлівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент
Онай Микола Володимирович _____

Рецензент:
Директор ННІМУМГ
Таврійського національного університету
імені В.І. Вернадського, д.т.н., професор
Кисельов Володимир Борисович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань. Студент _____

Київ – 2024 року

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Лавріненку Вадиму Володимировичу

1. Тема дисертації «Метод та програмне забезпечення для передбачення успішності волонтерських зборів», науковий керівник дисертації Олещенко Любов Михайлівна, к.т.н., доцент, затверджені наказом по університету № 5217-С від 09.11.2023.
2. Термін подання студентом дисертації «15» січня 2024 р.
3. Об'єкт дослідження: процес передбачення успішності волонтерських зборів.
4. Предмет дослідження: методи, алгоритми розподілених обчислень для класифікації волонтерських зборів.
5. Перелік завдань, які потрібно розробити:
- Дослідити наявні методи передбачення успішності волонтерських зборів.
- Розробити програмний метод, що дозволить передбачати успішність волонтерських зборів.
- Виконати програмну реалізацію методу.
- Порівняти отримані результати з результатами аналогічних методів та зробити висновки.
- Розробити бізнес-модель програмного продукту.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
– інтерфейс розробленого вебзастосунку;
– діаграми компонентів, послідовності;
7. Орієнтовний перелік публікацій:
– Тези доповіді "Метод та програмне забезпечення для передбачення успішності волонтерських зборів" на конференції «Прикладна математика та комп'ютинг. ПМК-2023».

8. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС, к.т.н., доцент		

9. Дата видачі завдання «10» жовтня 2022 р.

Календарний план

№	Назва етапів виконання з/п магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	10.10.2022	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури	04.12.2022	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	13.02.2023	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	10.04.2023	
5.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	10.10.2023	
6.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2023	27.11.2023	
7.	Оформлення текстової і графічної частини магістерської дисертації	25.12.2023	

Студент Вадим ЛАВРІНЕНКО

Науковий керівник Любов ОЛЕЩЕНКО

РЕФЕРАТ

Актуальність теми. Останнім часом в Україні набув поширення волонтерський рух, важливим інструментом якого є програмне забезпечення для зборів на нагальні потреби всіх прошарків населення в умовах війни. Постає проблема у формуванні очікувань до результатів зборів і ризик їх недофінансування. Враховуючи те, що індивідуальна консультація про можливість повного фінансування збору доступна не кожному, закономірно виникає потреба у програмній системі, що зможе автоматизовано надати цю послугу одночасно великій кількості користувачів.

Об'єктом дослідження є процес передбачення успішності волонтерських зборів.

Предметом дослідження є програмні методи та алгоритми розподілених обчислень для класифікації та передбачення успішності волонтерських зборів.

Мета роботи: зменшити час обробки даних зборів за допомогою впровадження архітектури із асинхронною обробкою повідомлень.

Методи дослідження: в роботі використовуються емпіричні методи вимірювання швидкодії прогнозування успішності волонтерських зборів розглянутими методами.

Наукова новизна роботи полягає в тому, що розроблений на основі розглянутих рішень метод із застосуванням розподілених обчислень суттєво зменшує час, необхідний програмі для навчання моделі передбачень та подальшої обробки зборів.

Практична цінність роботи полягає у запропонованій архітектурі розподіленої системи для розміщення оголошень про волонтерські збори та асинхронного передбачення їх успішності, що є конкурентоспроможною та готовою до розгортання у вигляді комерційного продукту.

Апробація роботи. Основні положення і результати даної роботи були представлені на науковій конференції магістрантів та аспірантів “Прикладна математика та комп’ютинг ПМК-2023”.

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, п’яти розділів, висновків та додатків.

У вступі викладено загальні засади та передумови вибору теми та проблематики роботи, обґрунтовано актуальність досліджень, сформульовано мету та задачі досліджень.

У першому розділі проведено аналіз наявних методів для вирішення проблеми; наведено приклади існуючих рішень та їх недоліки; визначено методи, що надалі буде використано в роботі з метою вирішення проблеми.

У другому розділі обґрунтовано вибір методу, що покращується, а також наведено опис алгоритмів, що використовуються у модифікованому методі для передбачень успішності волонтерських зборів.

У третьому розділі наведено опис програмної реалізації вдосконаленого методу, а саме: застосовані технології та архітектура розроблюваної системи.

У четвертому розділі виконано порівняння результатів реалізованого методу з альтернативами з точки зору точності навчених моделей та швидкодії передбачень.

У п’ятому розділі наведено бізнес-модель комерційного продукту, що базується на розглянутому програмному методі.

У висновках проаналізовано результати виконаної роботи та розглянуто напрямки подальших розробок.

Робота виконана на 88 аркушів, містить 4 додатки та посилання на список використаних літературних джерел з 32 найменувань. У роботі наведено 13 рисунків та 19 таблиць.

Ключові слова: класифікація, розподілена система, брокер повідомлень, швидкодія, подійно-орієнтована архітектура, передбачення успішності волонтерських зборів.

ABSTRACT

Actuality. Recently, a volunteer movement has become widespread in Ukraine, a powerful tool of which is the programmatic creation of fundraisers for the urgent needs of all strata of the population in war conditions. Consequently, there is a problem in forming unrealistic expectations for the results of said fundraisers and the risk of their underfunding. Given that not everyone can get an individual consultation on the possibility of full funding of the fundraiser, it is natural that there is a demand for a software system that can provide this service to a large number of users at once in an automated manner.

Object of research is the process of predicting the success of volunteer fundraisers.

Subject of research is methods, algorithms of distributed computing for classification of volunteer meetings.

Goal of the work is to decrease the time spent processing the fundraiser data by introducing an architecture utilizing parallel computing.

Methods of research: the thesis uses empirical methods to measure the performance of volunteer fundraiser success predictions as per the prediction methods described.

Scientific novelty of the thesis is that the method developed based on the analyzed solutions, utilizing distributed computing, significantly reduces the time required for the program to train the prediction model and further process the fundraisers.

Practical value of the work lies in the proposed architecture of a distributed system for posting announcements of volunteer fundraisers and asynchronously predicting their success, which is competitive and ready to be deployed as a commercial product.

Approbation. The results of this thesis were presented at the scientific conference of undergraduate and graduate students, "Applied Mathematics and Computer Science PMK-2023".

Structure and content. The master's thesis consists of an introduction, five chapters, conclusions, and appendices.

The introduction introduces the general principles and prerequisites for choosing the topic and issues of the work, substantiates the relevance of the research, and formulates the purpose and objectives of the research.

The first chapter analyzes the existing methods for solving the problem; provides examples of existing solutions with their problems and shortcomings; identifies the methods that will be used to solve the problem.

The second chapter explains the choice of the method to be improved and provides a description of the algorithms that the modified method utilizes.

The third chapter describes the software implementation of the improved method, namely, the technologies used and the target architecture of the system being developed.

The fourth chapter compares the results of the implemented method with the alternatives in terms of the accuracy of trained models and the speed of predictions.

The fifth chapter presents a business model of a commercial product based on the considered software method.

The key results of the work are analyzed in the conclusions, with possible directions of the work’s future improvements being discussed.

The thesis is presented in 88 pages, it contains 4 appendixes and 32 references to the used information sources, 13 figures and 19 tables are given in the thesis.

Keywords: classification, distributed system, message broker, performance, event-driven architecture, predicting the success of volunteer fundraisers

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

Задача класифікації – задача навчання під наглядом, що має на меті передбачення класу, до якого належить заданий об’єкт, спираючись на попередні відомості. Сам процес передбачає тренування моделі на наборі даних, об’єкти в якому заздалегідь віднесені до одного з відомих класів об’єктів [1].

Розподілена система – це архітектурна концепція, де обчислювальні завдання виконуються на різних вузлах мережі комп'ютерів, що співпрацюють для досягнення спільної мети. Це передбачає розсіяність обчислювальних ресурсів та даних, а також використання механізмів комунікації для обміну інформацією між вузлами. Розподілені системи дозволяють покращити масштабованість, надійність та продуктивність програмних додатків, а також забезпечують їхню адаптованість до змінних умов та вимог користувачів [2].

Паралельна обробка даних – це методологія обчислень, де обчислювальні завдання розбиваються на окремі незалежні підзадачі, які виконуються одночасно на різних обчислювальних ресурсах. Головна мета даної методології полягає в одержанні високої продуктивності та швидкодії за рахунок розділення роботи між багатьма обчислювальними одиницями [3].

Асинхронна обробка даних – це підхід до обчислень, де операції виконуються незалежно одна від одної, без чіткої синхронізації в часі. У цьому контексті програмні операції можуть запускатися без очікування завершення попередніх операцій, що дозволяє ефективно використовувати ресурси та прискорювати обробку завдань [4].

Архітектура, заснована на обміні повідомлень – це структурний підхід до розроблення програмного забезпечення, де взаємодія між компонентами відбувається через обмін повідомленнями. Кожен компонент у такій системі

є незалежним від інших і обмінюється повідомленнями для досягнення взаємодії та виконання завдань [5].

Брокер повідомлень – це проміжний компонент програмної системи, який відповідає за посередництво у взаємодії між різними компонентами, які обмінюються повідомленнями. Брокер служить поштовховим або тяговим механізмом для передачі повідомлень між відправниками та одержувачами, забезпечуючи асинхронний обмін і спрощуючи взаємодію у розподіленому середовищі [6].

Подія у подійно-орієнтованій архітектурі – ключовий концепт, що представляє важливий інцидент або зміну стану, яка сталася в системі або її навколишньому середовищі. Події можуть бути спричинені внутрішніми або зовнішніми стимулами і служать як засіб сповіщення компонентів системи про відбуті зміни. Компоненти можуть взаємодіяти, відправляючи та обробляючи події, що дозволяє системі реагувати на зміни оточення в реальному часі [7].

Подійно-орієнтована архітектура – це підхід до розроблення програмного забезпечення, в якому система визначається та організована навколо виникнення та обробки подій, які відбуваються в її середовищі. Кожен компонент системи може виступати як відправник чи одержувач подій, забезпечуючи розподілену та асинхронну взаємодію між її елементами [8].

Лямбда-функція – коротша форма визначення функцій у багатьох мовах програмування. Вона дозволяє створювати функції без необхідності визначення їх імені, часто використовуючи вирази, які можна викликати безпосередньо [9]. Також в межах даної роботи цей термін використовуватиметься для позначення одиниці програмного забезпечення, реалізованої як лямбда-функція, у середовищі сервісу “AWS Labmda”.

ВСТУП

Сьогодні в Україні значного розвитку набув волонтерський рух як засіб забезпечення матеріальних потреб як військових, так і цивільних. Втім, всі зацікавлені сторони кожного окремого збору – волонтери, що його оголошують; меценати, що вкладають кошти; та бенефіціари – не завжди мають реалістичне уявлення того, чи реальним є повноцінне забезпечення потреби коштами. Так, волонтерські фонди зазначають тенденцію обсягу пожертв до спадання з часом, через що збори та великі проекти повсякчас перебувають під ризиком недофінансування, або ж повного фінансування за часовий період, що перевищує очікуваний. Чи не головною причиною при цьому прийнято вважати доволі очевидну обмеженість ресурсів та погіршення добробуту населення. Це призводить до того, що потенційні меценати прагнуть застосувати власні кошти з максимальною користю – на збори, що є критичними, та, на їх думку, мають гарантований успіх. Водночас, менший пріоритет при цьому надається зборам, що видаються менш нагальними, або ж такими, повне фінансування яких не видається можливим з низки причин. Вочевидь, будь-яка людина не може орієнтуватись у кожній сфері, що може потребувати фінансування, тож часто видається неможливим визначити оцінки вищезгаданих рис збору самостійно, через що існує потреба на більш об’єктивний спосіб їх визначення. Наразі існують програмні методи передбачення успішності збору, втім для них суттєвою залишається проблема швидкодії [10], що зумовлює потребу в їх модифікації задля ефективного застосування в контексті українського волонтерського руху.

Отже, метою даної роботи є розробка методу та заснованого на ньому програмного забезпечення для передбачення успішності заданого збору за умови, що він дозволить обробляти велику кількість даних про збори із задовільними показниками швидкодії.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1. Аналіз методів прогнозування успіху волонтерських зборів

Прогнозування успіху волонтерських зборів належить до ширшої групи завдань, відомих як передбачувальне моделювання, що потребує використання історичних даних і методів статистичного моделювання для створення прогнозів або передбачень щодо майбутніх результатів або подій у заданій предметній області [11].

Зокрема, прогнозування успіху волонтерських зборів передбачає використання історичних даних про аналогічні проекти, таких як деталі проекту, інформація про донорів та результати кампанії для побудови прогностичної моделі. Потім ця модель аналізує різні особливості та фактори, пов'язані з волонтерськими проектами, щоб оцінити ймовірність успіху або невдачі для нових або поточних проектів.

1.1.1. Методи, що спираються на відомості про збори

Логістична регресія – це поширений алгоритм, який використовується для задач бінарної класифікації і може бути застосований для прогнозування успіху або невдачі волонтерського збору, або ж, в загальному випадку, фандрейзингового проекту. Враховуючи різні характеристики, такі як мета збору коштів, тривалість кампанії, залучення донорів, присутність у соціальних мережах та попередня історія збору коштів, логістична регресія може оцінити ймовірність успіху, якою ми вважаємо досягнення збором бажаного фінансування [12].

Випадковий ліс – це метод ансамблевого навчання, який поєднує кілька дерев рішень для прогнозування. Він може працювати як з числовими, так і з категоріальними ознаками та відображати складні взаємодії. Використовуючи такі характеристики, як опис проекту, цільова сума, тривалість кампанії, демографічні дані донорів, модель випадкового лісу може передбачити успіх проекту зі збору коштів [13].

Кожне дерево рішень у випадковому лісі будується шляхом рекурсивного розбиття даних на основі різних ознак. Розбиття виконується шляхом оцінки різних критеріїв, щоб знайти найкраще розбиття, яке максимізує розділення між цільовими класами. Після того, як всі дерева рішень побудовані, прогнози робляться по кожному дереву окремо. Для задач класифікації клас, що отримав більшість голосів серед дерев, обирається як остаточний прогноз. Для задач регресії береться середнє або медіана прогнозів дерев. Навчена модель випадкового лісу може бути використана для прогнозування нових, ще не бачених даних. Модель приймає відповідні характеристики нових даних як вхідні дані і генерує прогнози на основі об'єднання окремих прогнозів дерева. Ефективність моделі Random Forest оцінюється за допомогою відповідних оціночних метрик, таких як точність, достовірність, пригадування, середня квадратична помилка або R-квадрат. Також моделі випадкового лісу забезпечують аналіз важливості ознак, який оцінює важливість кожної ознаки для прогнозування. Цей аналіз базується на тому, наскільки кожна ознака сприяє зменшенню домішок або збільшенню інформативності в процесі побудови дерева. Важливість ознак може допомогти визначити найбільш впливові ознаки в задачі прогнозування.

Машини опорних векторів – це алгоритм машинного навчання, який знаходить оптимальну гіперплощину для поділу точок даних на різні класи. У контексті прогнозування волонтерських зборів SVM може використовувати такі характеристики, як мета збору коштів, тривалість проекту, характеристики донора, попередні результати збору коштів та інші відповідні атрибути для класифікації проектів як успішних чи неуспішних [14].

В цілому, існує декілька моделей даного алгоритму, що застосовуються для різних задач. Так, для задач бінарної класифікації зазвичай використовується стандартна модель SVM (також відома як C-SVM). Для задач багатокласової класифікації можуть використовуватися

такі варіації, як SVM "один проти одного" або "один проти всіх". Для задач регресії використовується модель регресії на основі опорних векторів (SVR). Зазначимо, що при прогнозуванні успішності збору маємо задачу саме бінарної класифікації, адже врешті-решт збір може бути проспонсований лише повністю, або ж не повністю.

SVM відомі своєю здатністю обробляти складні набори даних, обробляти високорозмірні простори ознак та ефективно обробляти нелінійні зв'язки, використовуючи різні функції ядра (наприклад, лінійну, поліноміальну, радіальну базисну функцію). Вони успішно застосовуються в різних галузях, включаючи класифікацію зображень, класифікацію текстів та найважливіше в межах даної роботи, – у фінансах.

Нейронні мережі, в тому числі моделі глибокого навчання, можуть бути ефективними у прогнозуванні успіху волонтерських проектів. Ці моделі можуть обробляти великі масиви даних і виявляти складні взаємозв'язки. Наприклад, багатошаровий персептрон або рекурентна нейронна мережа може бути навчена на даних зборів, включаючи такі характеристики, як мета збору коштів, тривалість кампанії, залучення донорів, активність у соціальних мережах та попередні результати проекту [15].

Набори даних зі збору коштів часто включають різноманітні типи інформації, такі як текстові описи, числові характеристики, метрики соціальних мереж і поведінку донорів. Нейронні мережі можуть ефективно обробляти ці багатовимірні дані, використовуючи такі методи, як згорткові нейронні мережі (CNN) для текстових даних, рекурентні нейронні мережі (RNN) для послідовних даних або гібридні архітектури для різноманітних типів даних. Така гнучкість дозволяє нейронним мережам використовувати багату інформацію, доступну в наборах даних зі збору коштів. Також вони можуть навчатися на даних з різними характеристиками і відповідно адаптувати свої внутрішні уявлення. Ця адаптивність робить їх добре пристосованими для роботи з різними контекстами фандрейзингу, такими

як кампанії різної тривалості, різноманітні категорії проектів або мінливі у часі вподобання донорів.

Нейронні мережі мають на меті узагальнювати навчальні дані, щоб робити точні прогнози на основі невидимих даних. Останні дослідження зосереджені на розробці методів для покращення узагальнення, таких як методи регуляризації (наприклад, відсіювання, зменшення ваги), ранньої зупинки та доповнення даних. Ці підходи допомагають нейронним мережам уникати перенастроювання і покращують їхню здатність прогнозувати успіх зборів для нових, ще не бачених проектів.

Random Forest – це ансамблевий метод машинного навчання, який використовується для класифікації та регресії. Він базується на ідейній комбінації багатьох дерев рішень для зниження перенавчання та покращення узагальнюючої здатності моделі.

Основні риси методу “Random Forest” включають в себе вибір випадкової підмножини ознак та випадкову вибірку навчальних даних для кожного дерева, що допомагає зменшити кореляцію між деревами та зробити модель більш стійкою до перенавчання. Кінцеве рішення випадкового лісу визначається шляхом голосування (у випадку класифікації) чи середнього значення (у випадку регресії) дерев.

Метод “Random Forest” відомий своєю високою точністю та вибором оптимальних ознак для задач класифікації та регресії, широко використовується в аналізі даних та машинному навчанні для різних завдань, включаючи прогнозування, класифікацію та важливість ознак [16].

Метод *градієнтного бустингу* – це ансамблевий метод машинного навчання, який використовується для покращення якості прогнозування в задачах класифікації та регресії. Метод градієнтного бустингу працює шляхом послідовного навчання слабких моделей (зазвичай дерев рішень) та поступового покращення результатів.

Основна ідея градієнтного бустингу полягає в тому, щоб кожна нова модель, який додається до ансамблю, вирішувала проблеми або помилки,

зроблені попередніми моделями. Це досягається за допомогою градієнтного спуску, де обчислюється градієнт функції втрат, і нова модель навчається передбачати цей градієнт, що призводить до покращення загальної точності моделі. Градієнтний бустинг дозволяє створювати потужні та гнучкі моделі, які можуть виконувати складні завдання машинного навчання. Градієнтний бустинг використовується в різних галузях, включаючи аналітику даних, рекомендаційні системи та багато інших задач [17].

1.1.2. Методи, що спираються на відомості про користувачів

Алгоритми рекомендацій можна використовувати для оцінки успішності волонтерського проекту, надаючи персоналізовані рекомендації потенційним донорам на основі їхніх уподобань та інтересів. Пропонуючи релевантні проекти, ці алгоритми можуть підвищити видимість і ймовірність підтримки проекту, що в кінцевому підсумку впливає на його успіх.

Колаборативна фільтрація – це алгоритм рекомендацій, який аналізує поведінку та вподобання користувачів для надання персоналізованих рекомендацій. У контексті волонтерських зборів спільна фільтрація може використовувати історичні дані про донорів, щоб визначити схожих донорів, які підтримували подібні проекти в минулому. Потім вона може рекомендувати проекти зі збору коштів потенційним донорам на основі проектів, які вони вже підтримали, або подібних проектів, які отримали значну підтримку. Зрештою, сума потенційних вкладів користувачів порівнюється з цільовою сумою збору, після чого з певною мірою впевненості можна передбачити успішність збору [18].

Першим кроком алгоритму є створення матриці "користувач-збір", яка відображає взаємодію між донорами та зборами, які вони можуть підтримати. Кожен рядок матриці відповідає користувачеві, кожен стовпчик – збору, а клітинки містять, як варіант, обсяг фінансової підтримки донором поточного проекта. Вочевидь, матриця може бути розрідженою, оскільки не

всі користувачі взаємодіяли з усіма елементами. Наступним кроком є розрахунок схожості між користувачами або ж проектами на основі їхніх шаблонів взаємодії. Можна використовувати різні метрики подібності, зокрема, коефіцієнт кореляції Пірсона. Подібність обчислюється шляхом порівняння обсягу вкладів у збори у вищезгаданій матриці. Користувачі або збори зі схожими моделями взаємодії матимуть вищі показники схожості.

Після обчислення оцінки схожості наступним кроком є визначення найближчих сусідів певного користувача або збору. Найближчими сусідами є такі, що мають найвищі показники схожості. Кількість сусідів, що розглядаються, визначається параметром *k*, який може бути визначений заздалегідь або обраний на основі оцінки ефективності. Наступним кроком є прогнозування рейтингів або продажів для конкретного користувача або товару. Для цього береться середньозважене значення рейтингів або продажів найближчих сусідів. Ваги визначаються за показниками схожості, де найближчі сусіди мають більшу вагу в прогнозі.

Фільтрація на основі контенту рекомендує проекти для збору коштів потенційним донорам на основі атрибутів і характеристик самих проектів. Вона враховує такі характеристики, як категорія проекту, мета збору коштів, опис проекту, відеоконтент та оновлення кампанії. Аналізуючи ці атрибути проекту, алгоритм може підібрати потенційним донорам проекти, які відповідають їхнім інтересам та вподобанням. Аналогічно попередньому методу, передбачення з успішності збору спирається на загальний обсяг коштів, що користувачі, яким збір буде рекомендовано, спроможні надати на нього, спираючись на їхню попередню історію вкладів [19].

Першим кроком методу є представлення кожного збору в системі на основі його характеристик або змісту. Це можуть бути такі атрибути, як опис, назва, напрямок, ключові слова. Метою є створення числового представлення або вектора ознак для кожного об'єкта, де кожна ознака представляє характеристику об'єкта. Водночас, кожен користувач також представлений профілем, який фіксує його вподобання або минулі

взаємодії. Профіль користувача може бути створений на основі зборів, з якими користувач раніше взаємодіяв – переглядав, поширював, робив внесок, тощо. Цей профіль допомагає визначити вподобання та інтереси користувача.

Після створення профілів зборів і користувачів, наступним кроком є розрахунок схожості між зборами або між зборами і користувачами. Зазвичай це робиться за допомогою метрик подібності, таких як косинусна подібність або евклідова відстань. Подібність обчислюється шляхом порівняння векторів характеристик зборів або профілів зборів і користувачів. Після обчислення показників схожості наступним кроком є прогнозування фінансування для конкретного збору або ж розміру вкладу користувача. Для цього враховуються оцінки схожості відповідних зборів або користувачів. Збори, які схожі між собою або схожі на вподобання користувача, з більшою ймовірністю будуть підтримані користувачем. Прогноз можна зробити, взявши середньозважене значення продажів або рейтингів подібних проектів або використовуючи інші методи регресії чи класифікації.

Гібридні алгоритми рекомендацій поєднують методи колаборативної фільтрації та фільтрації на основі контенту для надання більш точних і різноманітних рекомендацій. Ці алгоритми використовують дані про поведінку користувачів (наприклад, попередні пожертви) та атрибути проекту (наприклад, категорія проекту, мета збору коштів), щоб генерувати персоналізовані пропозиції. Враховуючи як вподобання користувачів, так і характеристики проекту, гібридні підходи можуть запропонувати збалансовану стратегію рекомендацій.

Алгоритми рекомендацій з урахуванням контексту враховують додаткові контекстні фактори при формуванні рекомендацій. Для проектів волонтерських зборів ці фактори можуть включати поточні тенденції у сфері благодійності, соціальні або екологічні проблеми, пору року (наприклад, сезон відпусток), географічне розташування, або геополітичні

чинники. Враховуючи контекстну інформацію, алгоритм може запропонувати актуальні та своєчасні проекти, що підвищує ймовірність залучення донорів та успіху проекту.

Першим кроком реалізації такого алгоритму є збір відповідної контекстної інформації, яка може вплинути на вподобання користувачів і їх пожертви. Це може бути часова інформація (час доби, день тижня, сезон), географічна інформація (місцезнаходження, погодні умови), інформація про користувача (демографічні дані, минула поведінка) та будь-які інші контекстні фактори, які вважаються важливими для прогнозування. Зібрану контекстну інформацію потрібно інтегрувати з наявними даними про користувачів і збори. Це може включати доповнення профілів користувачів і зборів контекстними характеристиками або створення окремих контекстних профілів.

Подібність між користувачами, зборами, або одного з іншим можна обчислити, враховуючи контекстні фактори разом з профілями користувачів і зборів. Показники схожості, що використовуються в підходах колаборативної фільтрації або фільтрації на основі вмісту, можуть бути розширені для включення контекстної інформації. Розрахунок схожості допомагає визначити користувачів зі схожими вподобаннями та проекти, які відповідають поточному контексту. Після розрахунку контекстної схожості наступним кроком є прогнозування обсягу вкладів на основі контекстних факторів. Цей прогноз можна зробити, беручи до уваги вподобання та поведінку схожих користувачів у схожих контекстах, а також атрибути та характеристики релевантних товарів. Для прогнозування продажів можуть застосовуватися різні методи, такі як колаборативна фільтрація, фільтрація на основі контенту або гібридні моделі.

1.2. Аналіз архітектурних підходів реалізації системи

Монолітна архітектура програмного забезпечення – це традиційний підхід до розробки, де весь програмний продукт розглядається як єдиний,

нероздільний блок. У такій архітектурі всі компоненти та функції програми зазвичай розміщені в одному програмному модулі чи кодовій базі, і вони безпосередньо взаємодіють між собою. Монолітні додатки зазвичай побудовані на одній технологічній платформі та можуть бути підтримувані, тестовані та розгорнуті як одне ціле.

Основні риси монолітної архітектури включають в себе відсутність чіткого розділення функціональності, що може призводити до великих та складних кодових баз, а також ускладнювати масштабування та розвиток додатків. Однак вона може бути більш простою для розроблення та розгортання в малих або простих проектах, де немає потреби в великій роздільності функцій і компонентів.

Значним недоліком застосування такого підходу в межах даної роботи є низька швидкодія та масштабованість передбачень успіху [20]. Так, за умови створення додатку типу “Web API”, можемо припустити, що прогнозування успішності збору виконуватиметься в межах запиту HTTP, що спричиняє затримки у відповіді на запити клієнта. У випадку ж великого навантаження на систему ці затримки можуть спричинити скасування запиту через вичерпання відведеного на нього часу ще до закінчення прогнозування, результатом чого є низька відмовостійкість системи.

Також слід згадати, що в такому випадку пропускна можливість компоненту передбачення успішності прямо залежить від пропускної можливості самого інтерфейсу “Web API”, а отже, система може бути не спроможна виконати прогнозування для збору за рахунок вичерпання наявних потоків, відведених на обробку отриманих клієнтських запитів.

Розподілена архітектура програмного забезпечення – це підхід до розробки, де функціональність програми поділяється на окремі компоненти, які виконуються на різних обчислювальних пристроях і взаємодіють через мережу. Кожен компонент може бути незалежним застосунком чи службою, і вони спільно працюють для забезпечення функціональності програми.

Основні риси розподіленої архітектури включають в себе розділення функціональності на окремі модулі, які можуть бути масштабовані та розгорнуті незалежно, покращену можливість обробки великих обсягів даних та підвищену надійність завдяки резервному копіюванню та балансуванню навантаження [21]. Отже, в межах даної роботи застосування даного підходу може суттєво збільшити пропускну здатність для процесу передбачення успішності зборів за рахунок масштабування, не залежного від компоненти взаємодії з користувацьким інтерфейсом.

Архітектура програмного забезпечення, заснована на обміні повідомлень, – це підхід, де компоненти програми взаємодіють між собою шляхом відправки та отримання повідомлень через спеціалізовані канали чи інші засоби комунікації. Кожен компонент може бути незалежним і виконувати конкретні функції, а взаємодія відбувається через обмін повідомленнями, що дозволяє досягти більш розподіленої та масштабованої системи [22].

Основні риси архітектури, заснованої на обміні повідомлень, включають асинхронну комунікацію між компонентами, високий ступінь розділення функціональності та здатність до побудови розподілених систем. Цей підхід часто використовується для створення складних та розподілених систем, таких як мікросервісні архітектури та системи обробки повідомлень.

В межах даної роботи даний архітектурний підхід може застосовуватись задля обміну інформацією про створення чи оновлення зборів між компонентами транзакційного оновлення даних про згадані збори у сховищі даних та безпосереднього передбачення успішності згаданого збору без потреби іншому робити додатковий запит у сховище, завдяки наявності всіх необхідних даних у тілі повідомлення. Водночас, брокер повідомлень може отримувати повідомлення, створені згідно з розкладом, що забезпечує можливість обробки змін у системі, що не залежать напряму від користувацьких дій.

1.3. Огляд існуючих рішень

“Krowdster” – це краудфандингова маркетингова та аналітична платформа, яка пропонує функції предиктивної аналітики. Вона надає власникам кампаній інформацію та прогнози щодо ефективності краудфандингових проектів на основі історичних даних, ринкових тенденцій та активності в соціальних мережах. “Krowdster” має на меті допомогти оптимізувати стратегії кампаній та збільшити шанси на успіх.

“Krowdster” пропонує детальну аналітику кампаній, що дозволяє користувачам відстежувати та аналізувати ключові показники, такі як прогрес фінансування, демографічні дані бенефіціарів, джерела рефералів та активність у соціальних мережах. Ці дані допомагають власникам кампаній розуміти свою аудиторію, вимірювати ефективність кампанії та приймати рішення на основі даних для покращення результатів. Також платформа надає доступ до повної бази даних потенційних спонсорів та впливових осіб, що дозволяє користувачам шукати та ідентифікувати осіб, які можуть бути зацікавлені у підтримці їхньої кампанії. Платформа використовує предиктивну аналітику, щоб надати власникам кампаній інформацію та прогнози щодо потенційного успіху їхніх краудфандингових кампаній. Аналізуючи історичні дані та ринкові тенденції платформа має на меті запропонувати прогнози та рекомендації для оптимізації стратегій кампаній.

В той час, як конкретний алгоритм рекомендацій зборів меценатам у “Krowdster” не розголошується, дослідження показують, що на рекомендації нових користувачів суттєво впливає історія вкладень “впливових” меценатів – таких, що вклали кошти у понад 10 зборів [23]. Водночас, меценати не отримують порад щодо вкладень з точки зору ймовірності повного фінансування проекту, натомість надаючи перевагу досвіду вищезгаданих “впливових” меценатів як рушію фінансування зборів.

“Kicktraq” – це інструмент предиктивної аналітики, розроблений спеціально для кампаній на “Kickstarter”. Платформа використовує алгоритми машинного навчання для аналізу даних проекту, тенденцій у категоріях, моделях фінансування та метрик соціальних мереж, щоб передбачити ймовірність успіху кампанії. “Kicktraq” надає власникам кампаній прогнози та рекомендації щодо покращення їхніх проектів на “Kickstarter”.

Функціональність “Kicktraq” полягає в аналізі історичних даних попередніх фандрейзингових кампаній, щоб виявити закономірності, тенденції та фактори, які сприяють успіху чи невдачі. Задля цього платформа використовує алгоритми машинного навчання, що роблять прогнози для нових кампаній на основі вхідних змінних, таких як деталі кампанії, цільова аудиторія, цілі фінансування та маркетингові стратегії. Також платформа забезпечує для користувача інформаційну панель, яка відображає прогнозовані ймовірності успіху поточних і майбутніх кампаній. З цією метою в панелі наявні графіки та звіти, які допоможуть власникам кампаній зрозуміти фактори, що впливають на прогнози.

Основною ж конкурентною перевагою “Kicktraq” є моніторинг показників ефективності кампанії в режимі реального часу, включаючи прогрес у фінансуванні, рівень залучення, згадування в соціальних мережах та демографічні дані донорів [24]. Зрештою, платформа надає можливості інтеграції власникам зборів програмними засобами через API, що надає змогу отримати аналітику у вихідному форматі та обробити її власними силами.

“*CrowdRise*” – платформа для збору коштів, що пропонує аналітичні функції, які дають уявлення про ефективність кампанії. Хоча вона не зосереджена безпосередньо на прогнозному моделюванні, вона пропонує організаторам кампаній інформацію на основі даних, включаючи тенденції пожертвувань, показники залучення та статистику соціальних мереж, яка може допомогти у прийнятті рішень і потенційно сприяти успіху кампанії.

“CrowdRise” надає власникам кампаній дані про отримані пожертви в режимі реального часу, що дозволяє їм відстежувати хід збору коштів. Сюди входить така інформація, як сума пожертвувань, дані про донорів та часові позначки внесків. Також користувачі можуть отримати доступ до показників, які дають уявлення про ефективність їхніх фандрейзингових кампаній. Це можуть бути дані про кількість прихильників, охоплення кампанії, рівень залучення та інші відповідні показники [25].

“CrowdRise” надає власникам кампаній інструменти звітності та аналітики для відстеження прогресу їхніх зусиль зі збору коштів. Користувачі мають доступ до даних про пожертви, охоплення кампанії та показники залучення в режимі реального часу, що дозволяє їм відстежувати свій успіх і приймати рішення на основі даних.

1.4. Висновки до розділу

У даному розділі було проведено аналіз наявних методів прогнозування успішності волонтерських зборів, а також наявних комерційних продуктів, що їх застосовують. Отриману інформацію було систематизовано з метою подальшого застосування за розробки власного рішення, спираючись на наведені особливості реалізації, переваги та недоліки методів та продуктів.

2. МОДИФІКАЦІЯ МЕТОДУ ПРОГНОЗУВАННЯ УСПІШНОСТІ ВОЛОНТЕРСЬКИХ ЗБОРІВ

2.1. Аргументація вибору базового методу прогнозування успіху зборів

При виборі базового методу було розглянуто наявні відомості про існуючі комерційні системи, а також низку переваг та недоліків стандартного набору методів, що мають на меті прогнозування тих чи інших величин, спираючись на історичні дані системи у заданій предметній області. Так, задача передбачення успішності збору належить до задач бінарної класифікації, де збір може належати до 2 класів: успішний та неуспішний. Зрештою, за базовий метод було обрано застосування градієнтного бустингу та, як наслідок, навчання моделі, що прогнозуватиме успіх волонтерських зборів. Це рішення спиралось на низку чинників:

- Точність прогнозування. Градієнтний бустинг є потужним інструментом для аналізу великих обсягів даних і виявлення складних залежностей. Вони можуть виявляти приховані кореляції та патерни, які можуть бути непомітними для людського аналізу. Це дозволяє досягати більш точних прогнозів продажів товарів, зменшуючи помилки і варіацію результатів.
- Автоматизація процесу. Розроблення програмного забезпечення з прогнозуванням успіху зборів за допомогою градієнтного бустингу дозволяє автоматизувати цей процес. Так, чи не найважливішим чинником у цьому є відсутність потреби у наявності теоретичних знань про предметну область, зокрема, у царині економіки. Також, ця риса підходу забезпечує те, що після налагодження програми вона може самостійно аналізувати вхідні дані, генерувати прогнози і оновлювати їх на основі нових даних, без потреби регулярного коригування залежно від макроекономічних чинників.
- Гнучкість і адаптивність. Градієнтний бустинг може враховувати різноманітні фактори та приховані залежності, що не вдалось би

виявити за самостійного опанування контексту предметної області та відлагодження реалізованого алгоритму, спираючись на неї. Це дозволяє програмному забезпеченню адаптуватись до змінних умов та враховувати нові фактори, що впливають на успішність волонтерських зборів у заданому проміжку часу. Таким чином, виконуються більш точні та актуальні прогнози.

- Масштабованість програмного забезпечення. За попереднього розгляду предметної області та обраного методу, було виявлено значну залежність між ступенем задоволення потреб користувача та швидкодією системи, актуальністю прогнозів. Так, було визначено значний недолік не наявного методу, що полягає у низькій швидкодії навчання та операцій прогнозування за виконання програми в одному потоці, без розпаралелювання обчислень та розподілу операцій у чергу на обробку [26]. Отже, спираючись на очевидний недолік, даний метод було обрано як такий, що має потенціал до вдосконалення завдяки деталях імплементації та архітектури системи.

2.2. Модифікація обраного методу прогнозування успіху зборів

Розглянемо модифікований метод прогнозування успішності волонтерських зборів, модифікація якого спричинена застосуванням у межах системи з розподіленням обчислень з метою своєчасної обробки оновлених даних та надання оновлених прогнозів у контексті багатокористувацької масштабованої системи з вебдодатком.

Навчання моделі за методом градієнтного бустингу передбачає такі етапи:

1. Початок виконання алгоритму передбачає створення слабкої моделі – дерева рішень, що передбачає цільову змінну.
2. Надалі визначачається помилка між передбаченими та фактичними значеннями цільової змінної. Мета навчання – мінімізація помилки

поточної моделі у подальших ітераціях. Так, на даному етапі розраховується градієнт функції втрат відносно прогнозів попередньої моделі.

3. Надалі створюється нова слабка модель, цільова змінна якої - значення градієнту з попереднього кроку. Модель додається до ансамблю. Попередня модель при цьому лишається незмінною.
4. Передбачене значення цільової змінної для кожної ітерації обчислюється як сума попереднього передбаченого значення цієї змінної та передбаченого моделлю з кроку 3 значення похибки.
5. Кроки з 2 до 4 повторюються, додаючи нові моделі до ансамблю доти, доки не буде досягнута задовільна точність, або поки не вичерпано вказану кількість ітерацій.

Розглянемо базовий алгоритм застосування градієнтного бустингу для прогнозування у контексті вебзастосунку в таблиці 2.1.

Таблиця 2.1

Базовий алгоритм навчання нейронних мереж для прогнозування у контексті вебзастосунку

Вхідні дані:	Історичні дані ведення волонтерських зборів, оформлені у файлі “csv”.
Вихідні дані:	Навчена модель, здатна передбачити успішність збору, заданого у базі даних.
Крок 1.	Завантаження даних в оперативну пам’ять.
Крок 2.	Нормалізація даних.
Крок 3.	Розподіл набору даних на навчальний та тестовий.
Крок 4.	Виконання навчання моделі до виконання умови зупинки.
Крок 5.	Верифікація моделі за допомогою надання довільної вибірки тестових даних як вхідних даних для навченої моделі.

Таблиця 2.2

Базовий алгоритм безпосередньо прогнозування у контексті вебзастосунку

Вхідні дані:	Новостворений збір.
Вихідні дані:	Прогнозована оцінка успішності збору.
Крок 1.	Завантаження даних в оперативну пам’ять.
Крок 2.	Нормалізація даних нового збору.
Крок 3.	Надання даних збору на вхід.
Крок 4.	Запис отриманого прогнозу в базу даних.

Зауважимо, що, в цілому, запропонований підхід також передбачає кроки зазначеного вище першого алгоритму. Водночас, розуміємо, що даний підхід не є масштабованим, а в межах запиту HTTP є й ненадійним з точки зору продуктивності та часу виконання через обмеження, поставлені клієнтом та сервером на максимальний час виконання запиту.

З огляду на ці чинники, було розроблено алгоритм паралельної обробки запитів на створення нових зборів, наведений у таблиці 2.3.

Таблиця 2.3

Алгоритм обробки запитів на прогнозування зборів

Вхідні дані:	Новостворений збір
Вихідні дані:	Прогнозована оцінка успішності збору
Крок 1.	Отримання запиту від клієнта.
Крок 2.	Завантаження поточних даних про збір.
Крок 3.	Формування і відправлення повідомлення про прогнозування для даного збору в чергу повідомлень.

Продовження табл. 2.3

Крок 4.	Прогнозування успішності збору в окремому потоці, спираючись на кроки наведені у таблиці 2.

Водночас, слід також розуміти, що для динамічної роботи системи недостатньо лише прогнозувати успішність окремих зборів, оскільки ситуація на ринку повсякчас змінюється, а отже, навчена модель поступово втрачає актуальність та потребує повторного навчання.

Алгоритм регулярного перенавчання мережі складається з кроків, наведених у таблиці 2.4.

Таблиця 2.4

Алгоритм повторного навчання моделі

Вхідні дані:	Автоматизований розклад запуску програми.
Вихідні дані:	Навчена модель.
Крок 1.	Завантаження поточних навчальних даних в оперативну пам’ять.
Крок 2.	Дозавантаження даних про збори, що було закрито – успішно чи ні – в період між поточним виконанням навчання та попереднім.
Крок 3.	Нормалізація даних.
Крок 4.	Розподіл набору даних на навчальний та тестовий.
Крок 5.	Виконання навчання моделі до виконання умови зупинки.
Крок 6.	Автоматизована верифікація моделі за допомогою надання довільної вибірки тестових даних для навченої моделі.
Крок 7.	Завантаження поточних активних зборів у оперативну пам’ять.
Крок 8.	Відправлення повідомлення для кожного збору в чергу на оброблення без очікування на завершення будь-якого з них.

Наведений вище алгоритм можемо узагальнити блок-схемою, наведеною на рис. 2.1.

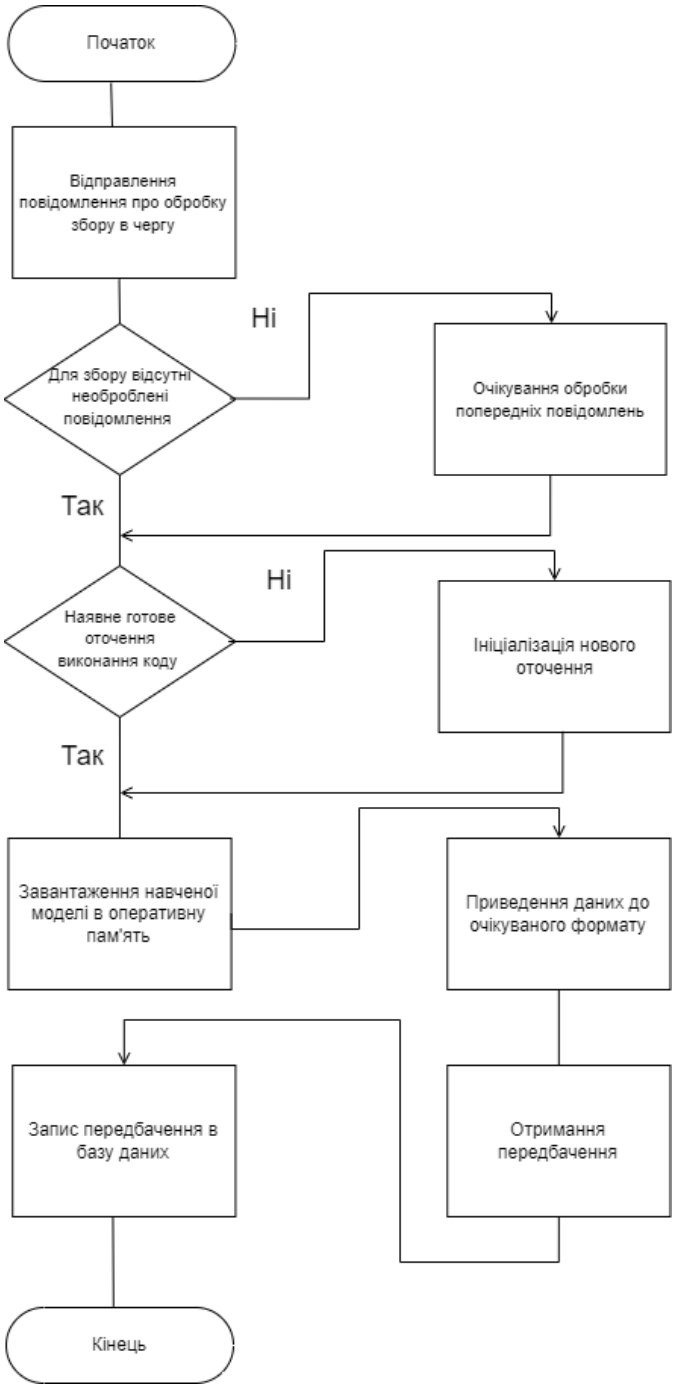


Рис. 2.1. Блок-схема модифікованого методу передбачення успішності зборів

Діаграму послідовності для випадку обробки двох повідомлень, згідно з описаним алгоритмом, наведено на рис. 2.2.

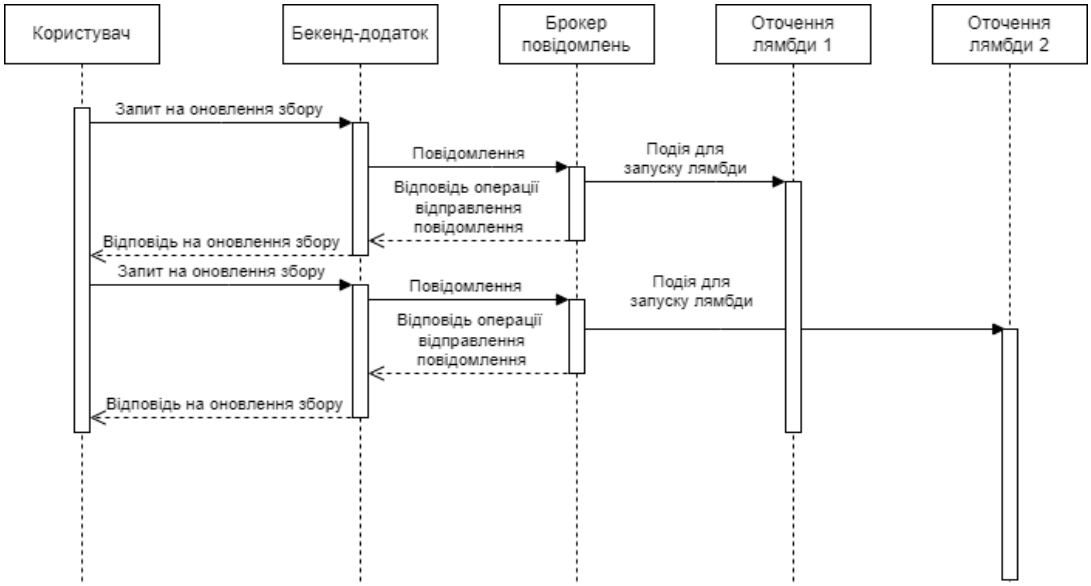


Рис. 2.2. Діаграма послідовності паралельної обробки двох повідомлень

2.3. Особливості реалізації модифікованого методу передбачення успішності зборів

Можемо стверджувати, що запропонована модифікація методу прогнозування успішності зборів має значні переваги над базовим за такими характеристиками, як продуктивність, масштабованість, відмовостійкість, портативність, та ін. Зокрема, це помітно за використання великих обсягів історичних даних – як вхідних за початкового розгортання системи, так і користувацьких даних, накопичених за термін експлуатації системи. Масштабування системи для підтримки одночасного доступу великої кількості користувачів та забезпечення своєчасного оброблення запитів є вкрай важливими вимогами при реалізації системи, тож на них буде звернено додаткову увагу при реалізації системи, що наведено в наступних розділах.

Важливим чинником подальшого вибору технічних засобів для програмної імплементації методу є їх відповідність архітектурним вимогам, що постають за розпаралелювання обчислень. Так, передусім зазначимо, що

обрана програмна черга повідомлень має виконувати такі функціональні вимоги:

- Масштабованість. Кількість обробників повідомлень має не бути обмеженою занизьким – з точки зору продуктивності системи, що буде надалі визначено емпірично – значенням. Натомість, очікуємо, що кількість обробників має бути конфігурована програматично.
- Одночасний багатокористувацький доступ до черги. Незалежно від кількості одночасних обробників, провайдер черги має забезпечити ідемпотентність повідомлень та можливість лише одноразового оброблення кожного з них, з метою уникнути стану гонитви, за якого повідомлення одночасно обробляється кількома потоками, призводячи до негативних побічних ефектів.
- Дедуплікація. Будь-яке повідомлення з запитом на обробку звіту, що вже наявне у черзі і ще не було обробленим, має бути проігноровано і не надіслано на повторну обробку, оскільки очікується, що потворне прогнозування для звіту не призведе до принципово змінених результатів, натомість зайнявши надлишкові обчислювальні ресурси.

2.4. Висновки до розділу

В межах даного розділу було виконано аналіз предметної області та наявних методів прогнозування успіху волонтерських зборів. Так, за результатами роботи було виконано такі кроки:

- Було обрано базовий метод.
- Було виявлено недоліки базового методу, що потребують модифікації в контексті проектованої системи.
- Було розроблено модифікацію методу, що виправляє вищезгадані недоліки.

- Було виконано попередній аналіз особливостей технічних засобів, необхідних для програмної реалізації методу.

Сама ж модифікація методу, як наслідок, полягає в застосуванні інструментів розпаралелювання обчислень, що дозволять ізолювати прогнозування успішності кожного окремого збору та перенавчання моделі задля актуалізації інформації про поточний стан системи, в окремих потоках, із застосуванням асинхронної черги повідомлень.

3. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ МОДИФІКОВАНОГО МЕТОДУ ПРОГНОЗУВАННЯ

3.1. Обґрунтування обраного набору технологій

З метою надання системою функціонального користувацького інтерфейсу, що надає можливості створення й перегляду волонтерських зборів, а також відстежування прогнозів з їх успішності, було вирішено виконати систему у вигляді вебзастосунку, внутрішня мережа клієнтської частини містить консольні додатки, що виконують асинхронну обробку даних в другорядних потоках.

Вибір набору технологій для програмної реалізації було виконано, спираючись на зазначені вище вимоги. Так, перелік застосованих при реалізації програмних засобів наведено далі.

3.1.1. .NET 6

Серверну частину системи, відповідальну за комунікацію з клієнтською частиною, було вирішено розробляти за допомогою фреймворку ".NET 6", зокрема наданого ним шаблону для реалізації вебзастосунків "Minimal API". У такого вибору є низка причин, зокрема, маломасштабний характер функціональності, за який відповідальна ця частина системи. Так, мінімалістичний характер фреймворку часто призводить до підвищення продуктивності. Видаляючи непотрібні абстракції та зменшуючи накладні витрати, додатки ".NET Minimal API" можуть забезпечувати кращі характеристики продуктивності порівняно з традиційними фреймворками "ASP.NET". Також ".NET Minimal API" має вбудовану підтримку ін'єкції залежностей, що дозволяє легко керувати та впроваджувати залежності в компоненти своїх додатків. Це сприяє кращій модульності, тестуванню та підтримці кодової бази.

".NET Minimal API" є кросплатформним, що означає, що можна створювати додатки в середовищах "Windows", "macOS", "Linux" та, що найважливіше в межах даної роботи, – для віртуальних машин, на яких

розгортаються додатки “AWS Lambda”. Ця гнучкість дозволяє орієнтуватися на ширший спектр платформ і охоплювати більшу аудиторію. Втім, контейнеризація додатку, що є необхідною умовою такого розгортання сервісу, й сама має низку переваг. Так, мінімалістичний характер “.NET Minimal API” робить його добре придатним для контейнерного розгортання. Додаток займає менше місця і швидше запускається, що може бути корисним у сценаріях, коли додатки потрібно швидко розгортати і масштабувати в контейнерних середовищах, зокрема, при сценаріях різкого масштабування за збільшення користувацького трафіку в системі. Зрештою, головною вимогою за вибору “.NET” як фреймворку бекенд-додатку була наявність підтримки ним “AWS SDK” [27] – програмного інтерфейсу з технологіями, на які спирається інфраструктура обміну повідомленнями, описана нижче.

3.1.2. Angular

“Angular” – це широко розповсюджений фреймворк “JavaScript”, що підтримується Google. Він забезпечує міцну основу для створення складних додатків, в тому числі для візуалізації даних про прогнози успіху волонтерських зборів у проектованій системі. Модульна архітектура та потужні функції Angular роблять його придатним для роботи з великомасштабними додатками, гарантуючи, що розроблюваний додаток може масштабуватися в міру зростання користувацького навантаження.

“Angular” використовує компонентну архітектуру, де додаток поділяється на багаторазові та незалежні компоненти. Такий підхід добре узгоджується з модульною природою розроблюваної системи, що передбачає окремі модулі для відображення переліку зборів, інформації про конкретний збір, прогнозів про нього, тощо; дозволяючи інкапсулювати різні частини функціональності прогнозу в компоненти багаторазового використання. Така модульність покращує підтримуваність, організацію

коду та можливість повторного використання, що призводить до більш ефективного процесу розробки.

Також обраний фреймворк надає широкі можливості для подальшого розширення системи. Так, функціональність регулярного асинхронного прогнозування успішності зборів може надалі бути розширена функціональністю зі сповіщення клієнтської частини застосунку про оновлену оцінку в реальному часі. Так, задля цього застосунки “Angular” мають широкий асортимент клієнтських бібліотек, для роботи з вебсокетами як потенційним варіантом виконання комунікації в реальному часі, а також “Redux Store” як засіб асинхронного збереження даних для інтерфейсу в оперативній пам’яті та їх оновлення, спираючись на систему реагування на події.

3.1.3. Python

“Python” має велику екосистему бібліотек, таких як “NumPy” та “Pandas”, які надають комплексні можливості для маніпулювання, аналізу та візуалізації даних. Ці бібліотеки дозволяють інтуїтивно підготувати історичні дані проектованої системи до обробки та, власне оброблювати їх з метою виявлення закономірностей та взаємозв'язків про предметну область.

Втім, головним чинником за вибору даної мови програмування була її широка підтримка функціональностей, пов’язаних з машинним навчанням. Такі бібліотеки, як “Scikit-learn”, “TensorFlow” та “PyTorch”, надають потужні алгоритми та інструменти для побудови предиктивних моделей, що є неодмінною умовою виконання проекту.

Зрештою, зазначимо, що цінною можливістю “Python” є здатність до інтеграції з багатьма сервісами завдяки наданим наборам розробки ПЗ. Так, в межах даної роботи вкрай необхідною є можливість інтеграції з низкою сервісів “AWS”, зокрема, “Simple Queue Service” та “Lambda”, про які йтиме мова далі – задля втілення функціональних вимог проектованої системи з

паралельної обробки даних. Так, наявність даного аспекту мови також забезпечує й високу масштабованість системи задля надання можливості роботи багатьом користувачам водночас, за рахунок розгортання додатку через сторонні сервіси.

3.1.4. AWS Lambda

“Лямбди” розроблені для роботи в рамках архітектури, керованої подіями, де кожна лямбда-функція запускається певною подією або запитом. Такий підхід забезпечує високу масштабованість і відокремленість системи, що дозволяє створювати розподілені системи, які швидко реагують на події та ефективно обробляють змінні робочі навантаження. Так, цей підхід є бажаним з огляду на використаний спосіб розподілення обчислень – відправлення повідомлень для кожного окремого збору в чергу, та, в інших випадках, – періодичний запуск програми за розкладом. Це дозволяє паралельно виконувати обчислення для даних різних користувачів, без виникнення стану гонитви.

“AWS Lambda” підпадає під концепцію безсерверних обчислень, яка усуває потребу в резервуванні та управлінні серверами. Масштабування автоматично виконується “AWS”, гарантуючи, що розподілена система зможе впоратися з різними рівнями навантаження й кількістю користувачів, що водночас користуються системою.

“AWS Lambda“ дозволяє легко масштабувати розподілену систему. Коли робоче навантаження зростає, AWS автоматично виділяє необхідні обчислювальні ресурси, щоб впоратися з ним. Таке автоматичне масштабування гарантує, що система зможе впоратися з високим трафіком або раптовими сплесками попиту без необхідності планування потужностей або операцій ручного масштабування.

Зрештою, філософія застосування “AWS Lambda” заохочує модульний підхід до проектування систем, що значною мірою підтримує проектована архітектура, що передбачає розподіл відповідальності з

відображення даних та кількох стадій їх обробки в окремі програмні компоненти.

3.1.5. AWS Simple Queue Service

“AWS SQS” – це сервіс, відповідальний за створення асинхронної черги повідомлень, що почергово отримують сервісами – обробниками задля паралельної обробки різних частин даних у системі. “AWS SQS” дозволяє роз'єднати компоненти в розподіленому додатку, надаючи надійний і масштабований сервіс обміну повідомленнями. Завдяки “SQS” компоненти можуть взаємодіяти асинхронно через черги повідомлень, що дозволяє їм працювати незалежно та у власному темпі. Таке розділення підвищує стійкість системи, відмовостійкість і масштабованість. Він може обробляти великі обсяги повідомлень і пристосовуватися до різних робочих навантажень без необхідності ручного втручання

“SQS” надає можливість розвантаження важких або трудомістких завдань з критичного шляху основного додатку серверної частини, зокрема, безпосереднього створення прогнозів для волонтерських зборів та донавчання моделі. Використовуючи “SQS” як буфер між різними компонентами, ми можемо асинхронно обробляти повідомлення у фоновому режимі [28]. Це допомагає рівномірно розподілити робоче навантаження і запобігає виникненню вузьких місць, гарантуючи, що додаток залишається швидким і масштабованим, залишаючи за собою змогу обробки запитів великої кількості користувачів вебзастосунку. Власне, “AWS SQS” розроблений для підтримки розподілених архітектур, подібних до проектованої в межах даної роботи. Цей сервіс може бути легко інтегрований в розподілені системи та архітектури мікросервісів, де компоненти взаємодіють через черги. Розподілена природа “SQS” дозволяє створювати високомасштабовані, відмовостійкі та слабо зв'язані системи, які можуть охоплювати різні регіони та зони доступності.

3.1.6. Docker

Docker – це платформа для контейнеризації програмного забезпечення, що дозволяє упаковувати програми та їх залежності в контейнери, які можна легко переносити та виконувати у різних системах. Так, в межах даної роботи це необхідно для розгортання лямбда-функцій, написаних мовами “C#”, “Python”, оскільки вони мають залежності на певні бібліотеки, що недоцільно розгортати інакше в межах екосистеми “AWS Lambda”.

Головною перевагою контейнерів, створених даною платформою, є те, що вони можуть бути запущені на будь-якій підтримуваній платформі, що забезпечує можливість побудови дистрибутивів залежностей цільовою архітектурою, використаною в “AWS Lambda”, на противагу тій, що розгорнута на машині, на якій відбувається розробка.

3.1.7. AWS CloudFront

“AWS CloudFront” є невід’ємною частиною клієнтських додатків, розгорнутих в середовищі “AWS”, виконуючи роль служби глобальної доставки контенту і кешування вебсайтів та інших ресурсів у мережі. Вона дозволяє розповсюджувати контент до користувачів з більшою швидкістю та ефективністю, забезпечуючи низьку затримку завантаження, зменшення навантаження на сервери та захист від DDoS-атак. CloudFront пропонує глобальну мережу кешування та має можливість інтеграції з іншими послугами AWS для покращення продуктивності та безпеки вебзастосунків.

В межах даної роботи, маємо змогу зробити припущення, що дані у системі не є часто змінюваними за рахунок низької кількості користувачів, що водночас додають нові збори, а також завдяки тому, що перенавчання моделі з подальшим застосуванням її для прогнозування успішності зборів відбувається за заздалегідь відомим розкладом. Спираючись на це, маємо змогу вже забезпечити майбутню швидкодію систему, застосувавши ресурси сервісу “AWS CloudFront” з метою створення кешу даних про

окремі збори в географічних локаціях, що найближчі до користувача, дані якого будуть актуальні до часу вищезгаданого процесу перенавчання.

Водночас, суттєвою перевагою застосування “CloudFront” у цьому контексті є змога кешування всього вебдодатку, завдяки чому суттєво зменшується час його завантаження за першого відкриття сторінки у браузері користувача.

3.1.8. Amazon S3

“Amazon S3” – це об'єктове сховище в хмарному середовищі від “AWS”, яке надає можливість зберігати та керувати об'єктами, такими як файли, зображення, відео та інші дані, у віртуальних сховищах. “Amazon S3” є масштабованим, надійним і досить доступним сервісом, який можна використовувати для забезпечення збереження та доступу до даних з будь-якого місця в Інтернеті.

В межах роботи “Amazon S3” доволі широко застосовується задля збереження спільних даних між програмними компонентами системи, зокрема, серіалізованих моделей передбачення успішності зборів. Водночас, функціональність з розгортання статичних вебсайтів є необхідною для розгортання клієнтської частини у формі односторінкового додатку.

3.2. Обґрунтування обраної архітектури ПЗ

Проектована архітектура системи передбачає наявність окремих програмних модулів для клієнтського вебзастосунку, що надає графічний інтерфейс користувачам; серверного застосунку, що надає “Web API” для комунікації серверної та клієнтської частини; низки програм – асинхронних обробників подій з оновлення користувацьких даних – створених чи оновлених волонтерських зборів.

Спираючись на наведений вище перелік програмних технологій для програмної реалізації системи, було спроектовано архітектуру системи, що

є доцільною, з точки зору використання технологій та процесу розподілення обчислень, на який спирається система. Так, зазначимо, що внаслідок необхідності застосування таких хмарних сервісів як “AWS Lambda” та “AWS Simple Queue Service”, вочевидь, було прийнято рішення розгортати кінцеву систему в хмарному провайдері “Amazon Web Services” задля інтеграції з уже наявною там архітектурою та уникнення складнощів, пов’язаних з мережевим налаштуванням та виділенням власних апаратних ресурсів за локального розгортання всієї системи. З урахуванням цього, було виконано діаграму окремих компонентів системи саме у нотації “AWS” (рис. 3.1) задля відображення використаних сервісів провайдера.

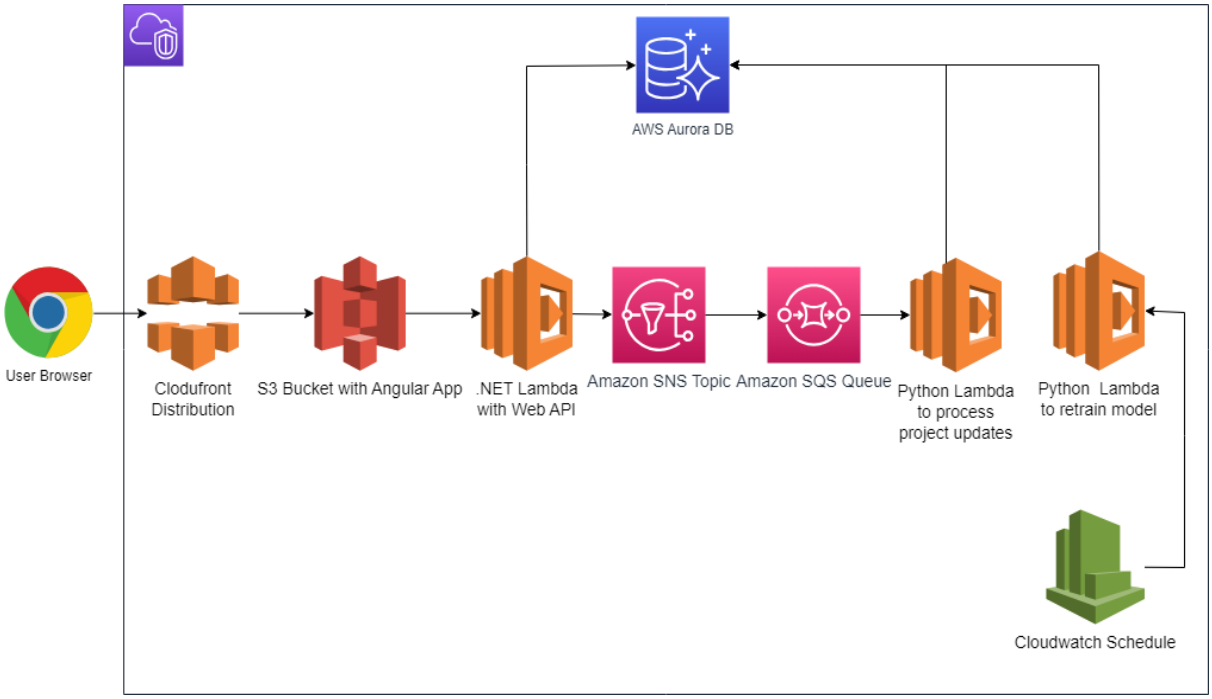


Рис. 3.1. Діаграма компонентів системи в анотації “AWS”

Наведена діаграма архітектури містить такі компоненти:

- Веббраузер користувача, через який відбувається взаємодія з клієнтською частиною системи.
- “Cloudfront” – сервіс “AWS”, елемент інфраструктури, відповідальний за налаштування мережевої комунікації між клієнтською машиною та додатком.

- “S3” – сервіс “AWS”, елемент інфраструктури, відповідальний за збереження необхідних файлів для розгортання вебзастосунку з користувацьким інтерфейсом.
- “AWS Lambda”, написана на фреймворку “.NET” – серверна частина системи, відповідальна за комунікацію з клієнтською. Містить логіку для фільтрації зборів для їх подальшого відображення клієнту, створення нових зборів, відправлення повідомлень про оновлення зборів на асинхронну обробку.
- Елемент “SNS” – сервіс “AWS”, елемент інфраструктури, відповідальний за перенаправлення створеного серверною частиною повідомлення на всі черги впорядкованої обробки повідомлень, що мають підписку на нього.
- Елемент “SQS” – сервіс “AWS”, елемент інфраструктури, відповідальний за функціональність черги, що міститиме повідомлення для обробки. За отримання нового повідомлення автоматично запускає виконання лямбд-функції за умови, що наразі виконується не максимальна їх кількість.
- Лямбда-функція, відповідальна безпосередньо за прогнозування успішності збору, що було оновлено користувачем.
- Лямбда-функція, відповідальна за тестування швидкодії системи, що передбачає створення навантаження на систему певною кількістю повідомлень. Вона передбачає конфігурацію, що розробник може змінювати задля збільшення навантаження на систему, а також зміни цільової архітектури та базового методу, для яких виконується тестування.
- Елемент “CloudWatch Event Bridge” – сервіс “AWS”, елемент інфраструктури, відповідальний за періодичний запуск лямбда-функції перенавчання моделі, згідно з певним розкладом.
- Лямбда-функція, відповідальна за періодичне перенавчання моделі.

- База даних “Amazon RDS Aurora”, відповідальна за збереження історичних даних про збори у системі, а також обчислених прогнозів успішності поточних зборів.

3.3. Аналіз розробленої системи

Розроблена система передбачає не лише обробку наявного для навчання та тестування набору даних, а й надходження даних про збори, створених користувачем за допомогою графічного інтерфейсу. Графічний інтерфейс користувача передбачає такий набір функціональності:

1. Створення та редагування зборів (рис. 3.2). Так, користувач має змогу змінювати назву збору, його опис, категорію, цільову суму та валюту, в якій вона обчислюється, дати початку та кінця, а також дані про надходження. В межах даної роботи інтеграцію з банківською системою, що вела б облік надходжень, було замінено можливістю самостійного введення отриманого обсягу пожертувань, зафіксованого у задану дату, що входить у межі тривалості збору.
2. Перегляд наявного переліку зборів (рис. 3.3). Користувачеві надається можливість перегляду списку зборів, що містить інформацію про назву, категорію, кількість днів до кінцевого терміну, частку отриманих коштів від бажаних, а також показник передбаченої успішності збору. На сторінці наявна функціональність пошуку зборів серед переліку за назвою, переходу на сторінку редагування виділеного збору, а також його видалення.

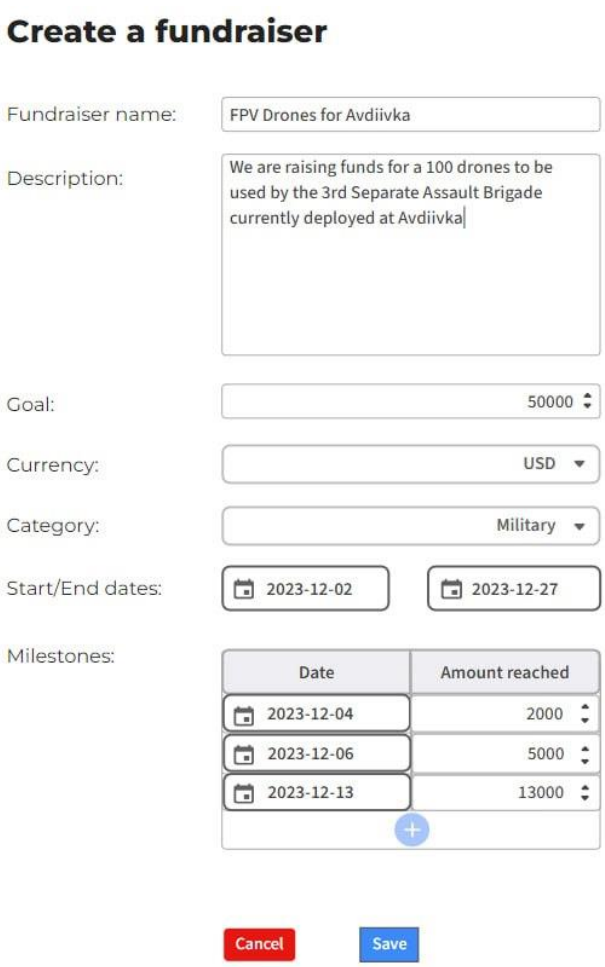


Рис. 3.2. Екран створення збору

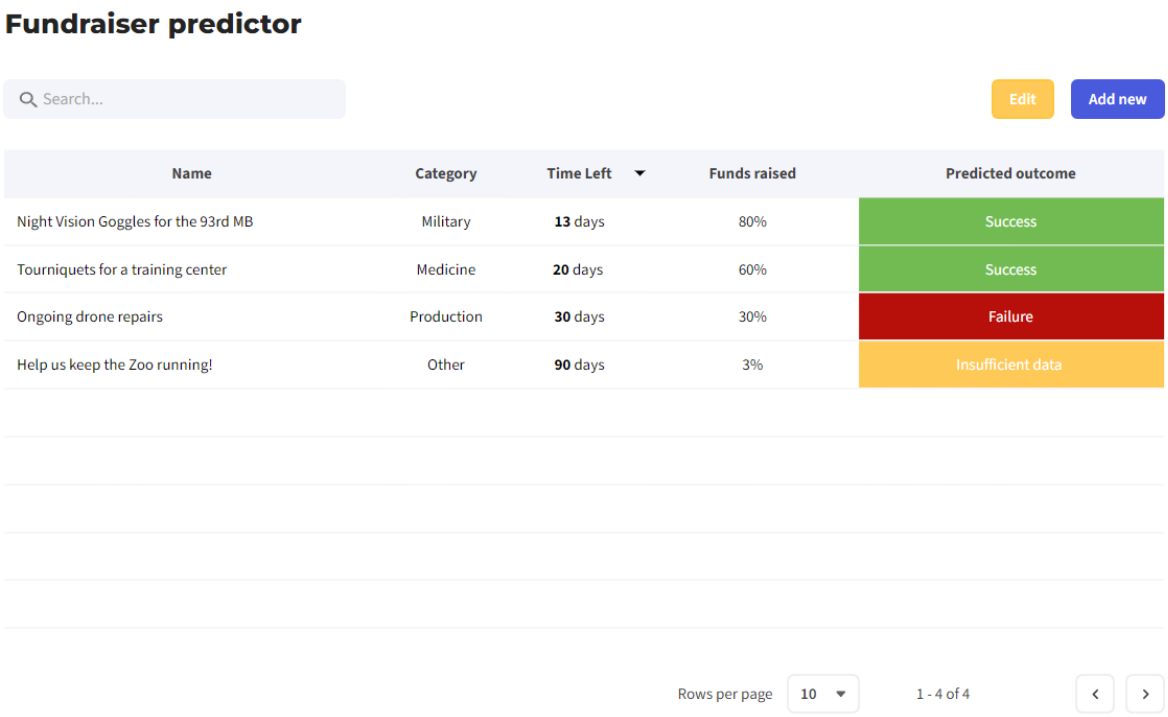


Рис. 3.3. Екран списку створених зборів

Як було згадано вище, в межах даної роботи надходження коштів на збори було імітовано за допомогою користувацького інтерфейсу, що надає можливість введення історичних даних про надходження коштів, замість взаємодії з існуючою платіжною системою. Водночас, майбутні розробки системи можуть бути націлені на інтеграцію з банковою системою. Так, запропонована реалізація системи містить низку переваг, що забезпечать швидкість та надійність обліку надходжень у системі:

1. Використання брокера повідомлень “Amazon SQS” передбачає збільшення швидкодії передбачень за рахунок паралельних обчислень.
2. Обраний брокер повідомлень забезпечує надійність надходження та вдалої обробки повідомлень. Так, надсилання клієнтським застосунком повідомлень у чергу як в основному регіоні “AWS”, так і другорядному, забезпечить вдале надходження повідомлення до обробника та його обробку навіть у випадку технічних збоїв інфраструктури в одному з регіонів.
3. Обрана конфігурація черги забезпечує одноразове надходження повідомлень – без дублікатів. Таким чином неможливим є подвійне зарахування користувацьких коштів через технічний збій інфраструктури.
4. Обрана конфігурація черги також передбачає виключно в хронологічному порядку, завдяки чому кінцеве передбачення успішності збору грунтується виключно на актуальні дані.

3.4. Висновки до розділу

В рамках дослідження шляхів програмного виконання модифікованого методу прогнозування успішності волонтерських зборів було виконано вибір та обгрунтування набору технологій для реалізації системи, а також спроектовано відповідну їй архітектуру.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Порівняння точності та швидкодії навчання обраного базового методу з аналогами

Було виконано аналіз точності класифікації моделей, отриманих обраним методом, а також логістичною регресією та випадковим лісом, як можливі базові методи. Навчання моделей та пов'язане тестування було виконано з набором даних, наданих компанією "Kickstarter", що містить 375765 рядків про фандрейзингові проекти, розміщені у вищезгаданому ресурсі [29]. Так, кожен рядок відповідає певному збору та містить 15 стовпців, що відповідають таким його рисам:

- внутрішній ідентифікатор збору в базі даних сервісу, ціле число;
- назва збору, рядкове значення;
- назва категорії, до якої належить збір, рядкове значення;
- назва підкатегорії, до якої належить збір, рядкове значення;
- назва цільової валюти, в якій визначено бажаний обсяг коштів для реалізації проєкт, рядкове значення;
- назва країни, де реалізується проєкт, рядкове значення;
- кінцева дата збору, значення типу дати у форматі "YYYY-MM-DD HH:mm:ss";
- бажаний обсяг коштів для реалізації проєкта, у цільовій валюті, ціле число;
- початкова дата збору, значення типу дати у форматі "YYYY-MM-DD HH:mm:ss";
- поточна сума, яку було зібрано проєктом у цільовій валюті, число з плаваючою комою;
- поточна сума, зібрана в доларах США, згідно з обчисленнями системи "Kickstarter" , число з плаваючою комою;
- поточна сума, зібрана в доларах США, згідно з обчисленнями системи "fixer.io" , число з плаваючою комою;
- кількість меценатів, що вклали кошти у проєкт, ціле число;

- поточний стан проєкта, рядкове значення.

Зауважимо, що в межах даної роботи згаданий вище набір даних, наданий у файлі формату “csv” не було інтегровано безпосередньо з базою даних, що містить інформацію про збори, створені користувачем розробленої програмної системи. Натомість, цей файл зберігається у сховищі, наданому сервісом “Amazon S3”. У лістингу 4.1 наведено приклад коду, що завантажує набір даних у пам’ять задля навчання моделі та завантажує .

Лістинг 4.1. Завантаження набору даних у пам’ять та збереження навченої моделі

```
import tempfile
import boto3
import joblib
df = pd.read_csv("s3://dissertation-resources/ks-projects-201801.csv")
model = XGBClassifier()
# WRITE
s3_client = boto3.client('s3')
bucket_name = "dissertation-resources"
key = "xgboost.pkl"
with tempfile.TemporaryFile() as fp:
    joblib.dump(model, fp)
    fp.seek(0)
    s3_client.put_object(Body=fp.read(), Bucket=bucket_name, Key=key)
```

Перед виконанням коду навчання, необхідне додаткове перетворення набору даних. Так, частина колонок, що використовуються для навчання моделі та передбачення успішності збору, надалі вважаються такими, що містять дані перелічуваного типу даних, отже будуть представлені як

цілочисельне значення із визначеного діапазону. Отже, для передбачень визначеня такі перелічувальні типи даних:

- категорія збору;
- цільова валюта збору;
- країна, де виконується збір;
- кінцевий стан збору.

Набір даних містить виключно рядкові значення в зазначених колонках. Код, що створює відповідність між рядковими значеннями вищезгаданих колонок та їх цілочисельними відповідниками, наведено у лістингу 4.2.

Лістинг 4.2. Створення словників відповідності рядкових і цілочисельних значень колонок набору даних

```
categorical_features = ["main_category" , "currency" , "state", "country"]
le = LabelEncoder()
for x in categorical_features:
    df[x] = le.fit_transform(df[x].values)
    le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    print(le_name_mapping)
```

Результати виконання коду наведено на рис. 4.1.

```
{'Art': 0, 'Comics': 1, 'Crafts': 2, 'Dance': 3, 'Design': 4, 'Fashion': 5, 'Film & Video': 6, 'Food': 7, 'Games': 8, 'Journali
sm': 9, 'Music': 10, 'Photography': 11, 'Publishing': 12, 'Technology': 13, 'Theater': 14}
{'AUD': 0, 'CAD': 1, 'CHF': 2, 'DKK': 3, 'EUR': 4, 'GBP': 5, 'HKD': 6, 'JPY': 7, 'MXN': 8, 'NOK': 9, 'NZD': 10, 'SEK': 11, 'SG
D': 12, 'USD': 13}
{'canceled': 0, 'failed': 1, 'live': 2, 'successful': 3, 'suspended': 4, 'undefined': 5}
{'AT': 0, 'AU': 1, 'BE': 2, 'CA': 3, 'CH': 4, 'DE': 5, 'DK': 6, 'ES': 7, 'FR': 8, 'GB': 9, 'HK': 10, 'IE': 11, 'IT': 12, 'JP':
13, 'LU': 14, 'MX': 15, 'N,0"': 16, 'NL': 17, 'NO': 18, 'NZ': 19, 'SE': 20, 'SG': 21, 'US': 22}
```

Рис. 4.1. Результати виконання коду створення словників відповідності рядкових і цілочисельних значень колонок набору даних

Зазначимо, що підготовка даних також передбачає додаткові обчислення на основі вхідних значень дат початку та кінця зборів. Так, у навчанні моделі та передбаченнях використовується тривалість збору у днях, обчислена як кількість днів від початку збору до його кінця для

тренувальних даних, або ж від початку збору до поточної дати у випадку передбачення успішності активного збору. У лістингу 4.3 наведено код, що виконує обисані обчислення.

Лістинг 4.3. Обчислення тривалості збору

```
def calculate_duration_days():
    days_apart = []
    launched = df["launched"].values
    i = 0
    deadline = df ["deadline"].values
    for x in launched:
        days_apart.append((parser.parse(deadline[i]) - parser.parse(x)).days)
        i = i +1
    return days_apart
df["days_apart"] = calculate_duration_days()
df[["launched", "deadline", "days_apart"]]
```

Результати виконання коду наведено на рис. 4.2.

	launched	deadline	days_apart
0	2015-08-11 12:12:28	2015-10-09	58
1	2017-09-02 04:43:57	2017-11-01	59
2	2013-01-12 00:20:50	2013-02-26	44
3	2012-03-17 03:24:11	2012-04-16	29
4	2015-07-04 08:35:03	2015-08-29	55
...	...	...	...
378656	2014-09-17 02:35:30	2014-10-17	29
378657	2011-06-22 03:35:14	2011-07-19	26
378658	2010-07-01 19:40:30	2010-08-16	45
378659	2016-01-13 18:13:53	2016-02-13	30
378660	2011-07-19 09:07:47	2011-08-16	27

378661 rows × 3 columns

Рис. 4.2. Результати виконання обчислення тривалості збору

Розгляньмо код, що виконує навчання моделі на прикладі градієнтного бустингу. Спершу набір даних ділиться на навчальну та тестову вибірки у співвідношенні 70% до 30%. Надалі створюється екземпляр класу класифікатора, що реалізує метод градієнтного бустингу. Він навчає модель на вхідних даних з навчальної вибірки, намагаючись передбачити значення кінцевого стану збору, закодованого як вищезгаданий тип перелічення. Зокрема, найбільшої точності було досягнуто за конфігурації класифікатора, наведеної в лістингу 4.4:

- швидкість навчання – 0.15;
- кількість дерев, що будує класифікатор, – 150;
- максимальна глибина дерева – 5;
- цільова функція – логістична.

Лістинг 4.4. Навчання моделі, що використовує градієнтний бустинг

```
X = df[features]
y = df[label]
train_x,test_x,train_y,test_y = train_test_split(X,y,test_size=0.3)
model = XGBClassifier(learning_rate=0.15, n_estimators=150, max_depth=5,
objective='binary:logistic', nthread=2)
model.fit(train_x,train_y)
y_pred = model.predict(test_x)
score = sklearn.metrics.accuracy_score(y_pred,test_y)
print(score)
```

Отримані результати точності моделей, що використовують логістичну регресію, випадковий ліс та градієнтний бустинг, наведено у табл. 4.1.

Так, результати підтверджують доцільність використання градієнтного бустингу для поставленої задачі, оскільки цей метод забезпечує найкращі результати.

Таблиця 4.1

Точність отриманих моделей

Метод	Точність, %
Логістична регресія	76
Випадковий ліс	79
Градієнтний бустинг	83

Водночас, архітектура розробленої системи передбачає виконання навчання моделі за розкладом кілька разів на добу, внаслідок чого швидкодія саме цієї частини системи наразі не є критичною для функціонування системи загалом, але становить можливий ризик для її функціонування по мірі того, як змінюються нефункціональні вимоги з одночасним збільшенням частоти навчання моделі. З огляду на це, порівняльну характеристику швидкодії процесу навчання моделей вищезгаданих методів було пропущено, адже він не вплинув на вибір методу.

З метою порівняння швидкодії навчання моделей, що використовують методи, що розглядаються, було обрано конфігурацію “AWS Lambda”, що встановлює максимальний обсяг оперативної пам’яті для оточення виконання програми у 512 МБ. Дана конфігурація передбачає використання кожним оточенням одного віртуального процесора, швидкодія якого є аналогічною фізичному процесору сімейства “Intel Xeon”, що має 6 фізичних ядер з максимальною тактовою частотою 3.3 ГГц та 16 МБ кеш-пам’яті L3 [30]. Ця конфігурація була обрана емпірично, як така, що є мінімально необхідною для завантаження навчального набору даних в оперативну пам’ять та виконання навчання моделі.

З отриманих результатів бачимо, що середній час виконання лямбди навчання за методом градієнтного бустингу є найбільшим (табл. 4.2).

Тим не менш, отримана тривалість у 217 секунд задовольняє умови швидкодії, для яких застосування “AWS Lambda” є доцільним, а отже, на даному етапі втілення системи, вважаємо його задовільним, оскільки в межах даної роботи точність моделі має більший пріоритет, аніж швидкодія процесу навчання.

Таблиця 4.2

Середній час навчання отриманих моделей

Метод	Середній час виконання функції навчання моделі, с
Логістична регресія	217
Випадковий ліс	260
Градієнтний бустинг	279

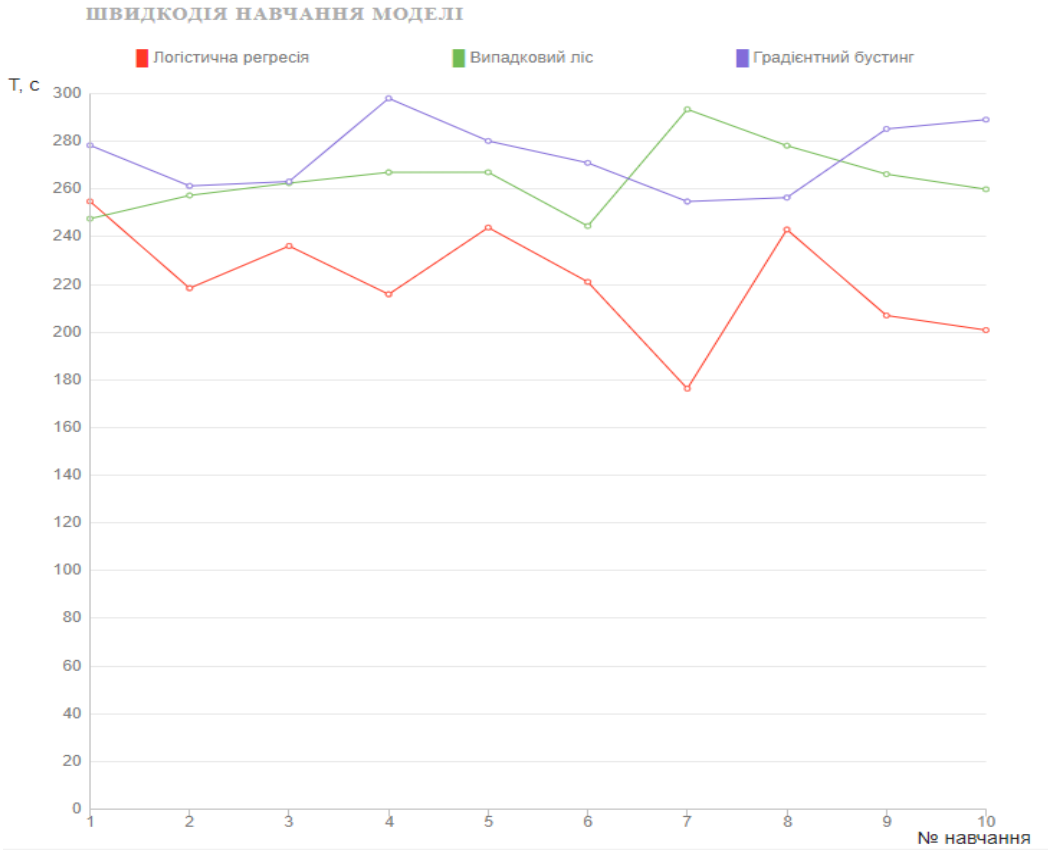


Рис. 4.3. Швидкодія навчання моделей

4.2. Порівняння швидкодії передбачення запропонованого методу з аналогами

Однією з поставлених задач роботи є покращення швидкодії прогнозування великої кількості зборів водночас. З огляду на це, було виконано низку тестів швидкодії прогнозування успіху з використанням архітектур послідовної обробки подій та паралельної обробки на основі обміну повідомленнями між сервісами. Зокрема, для другого варіанту було виконано тестування для двох різних конфігурацій лямбда-функції: з використанням заздалегідь запущених 10 екземплярів функцій, завдяки чому можемо уникнути довгого часу початкового розгортання оточення функції; та з конфігурацією, що передбачає холодний старт функцій та, як наслідок, довший час виконання коду для першого набору повідомлень. Результати тестів наведено у табл. 4.3.

Таблиця 4.3

Загальний час прогнозування заданого набору зборів

Використана архітектура	Послідовна обробка	Паралельна обробка за допомогою повідомлень з використанням 10 готових лямбд	Паралельна обробка за допомогою повідомлень з використанням лямбд з холодним стартом
Час обробки 10 зборів, с	29	2	30
Час обробки 50 зборів, с	81	8	36
Час обробки 100 зборів, с	156	15	49
Час обробки 500 зборів, с	813	126	149

Так, згідно з результатами, послідовна обробка показує суттєво гірші результати зі збільшенням кількості повідомлень, попри кращі результати за кількості повідомлень, що становить 10. Це пояснюється тим, що в даному випадку ділянка коду, відповідальна за ініціалізацію лямбди, виконується лише раз, на противагу початковій ініціалізації всіх 10 лямбд, завдяки чому вдається нівелювати переваги паралельної обробки повідомлень, обробивши їх послідовно в одному потоці з задовільними показниками швидкодії. Тим не менш, показники за більшого навантаження дають можливість стверджувати, що застосування паралельної обробки за допомогою брокера повідомлень є більш ефективним.

Водночас діаграма розсіювання часу виконання коду для кожного повідомлення вказує на значну перевагу наявності готових оточень для лямбди, оскільки основна частина часу, витраченого на обробку перших повідомлень за конфігурації, що їх не має, витрачена саме на перший запуск лямбди та завантаження залежностей коду.

Зазначимо ж, що “AWS” лишає запущеними вже наявні оточення лямбди, за умови що вони активно використовуються. Отже, можемо припустити, що готовий комерційний продукт стабільно матиме швидкодію, що відповідає конфігурації з повсякчас запущеними лямбдами, хоч в окремих випадках слід очікувати меншої швидкості обробки повідомлень за рахунок потреби у запуску нових оточень.

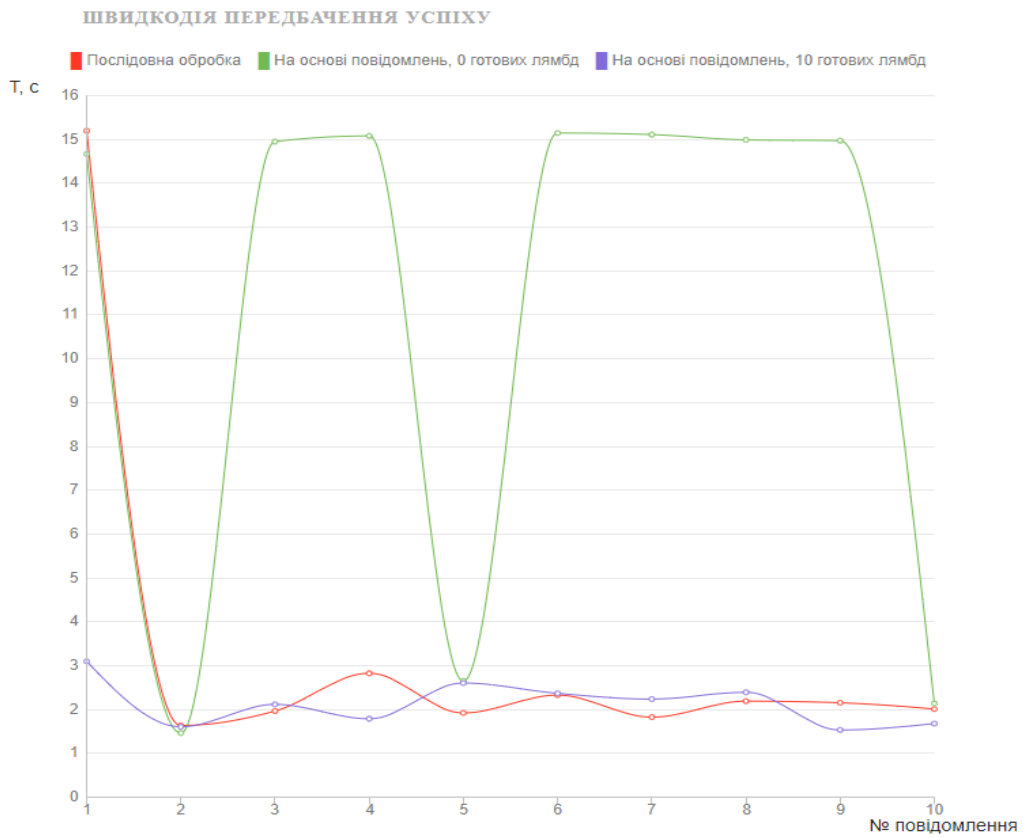


Рис. 4.4. Швидкодія конфігурацій для 10 повідомлень

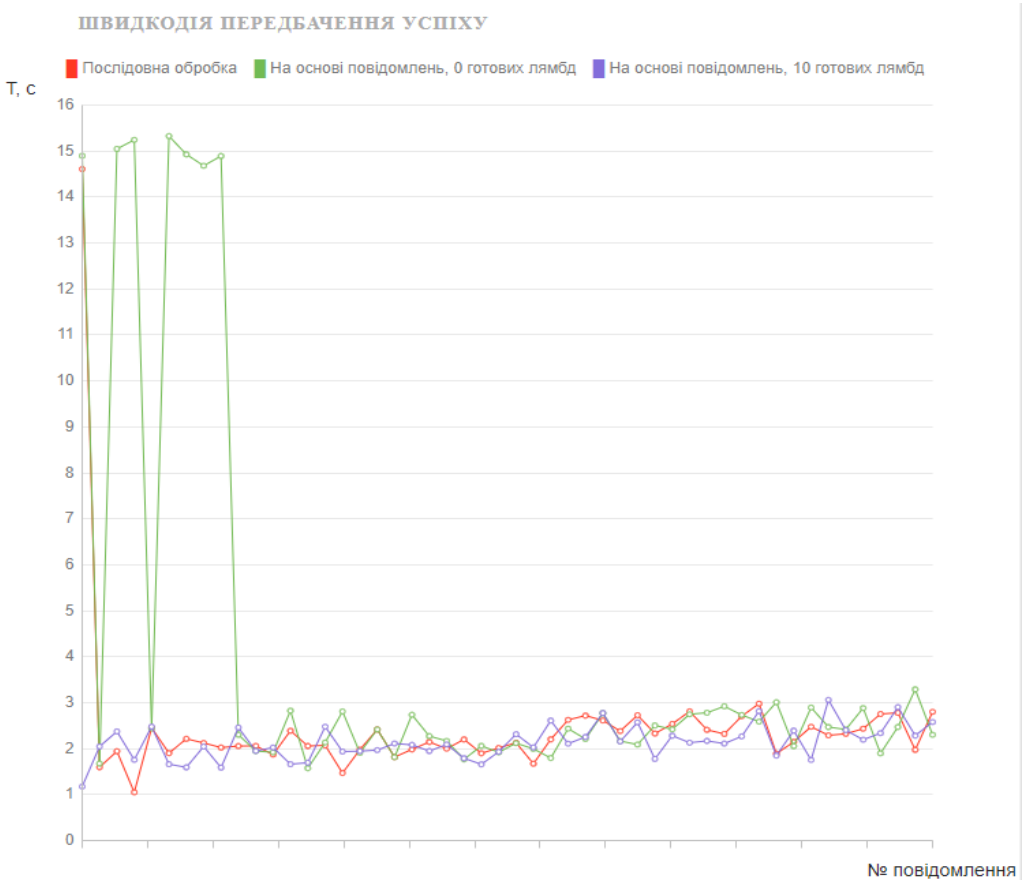


Рис. 4.5. Швидкодія конфігурацій для 50 повідомлень

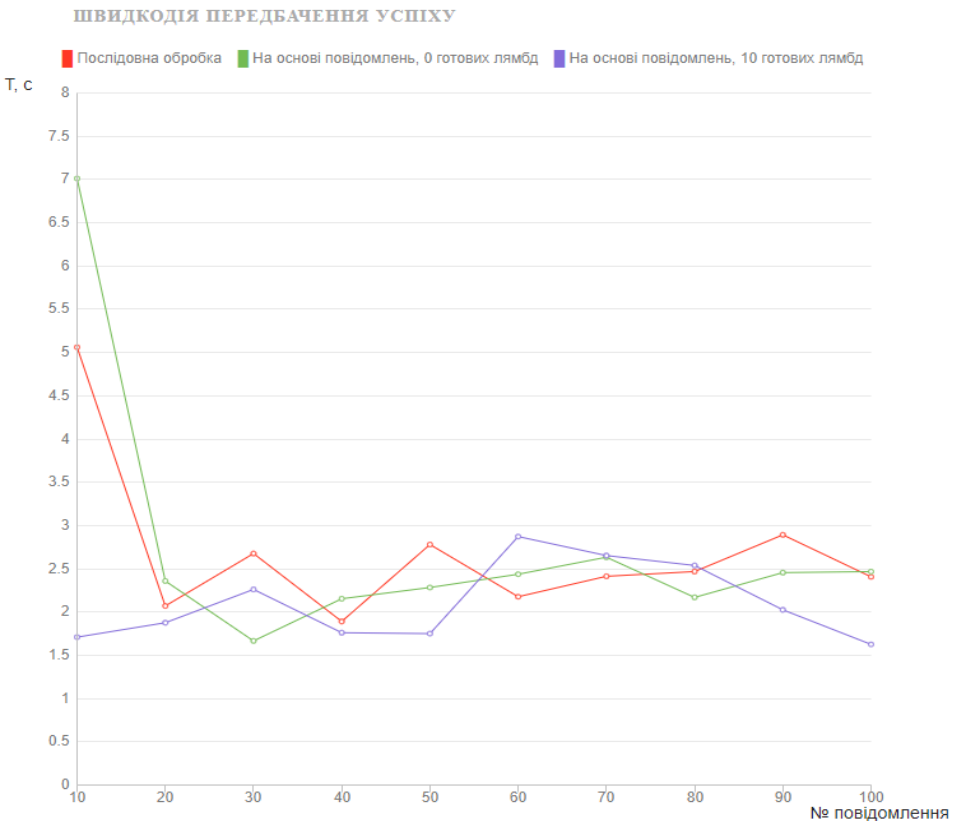


Рис. 4.6. Середні показники швидкодії конфігурацій для 100 повідомлень

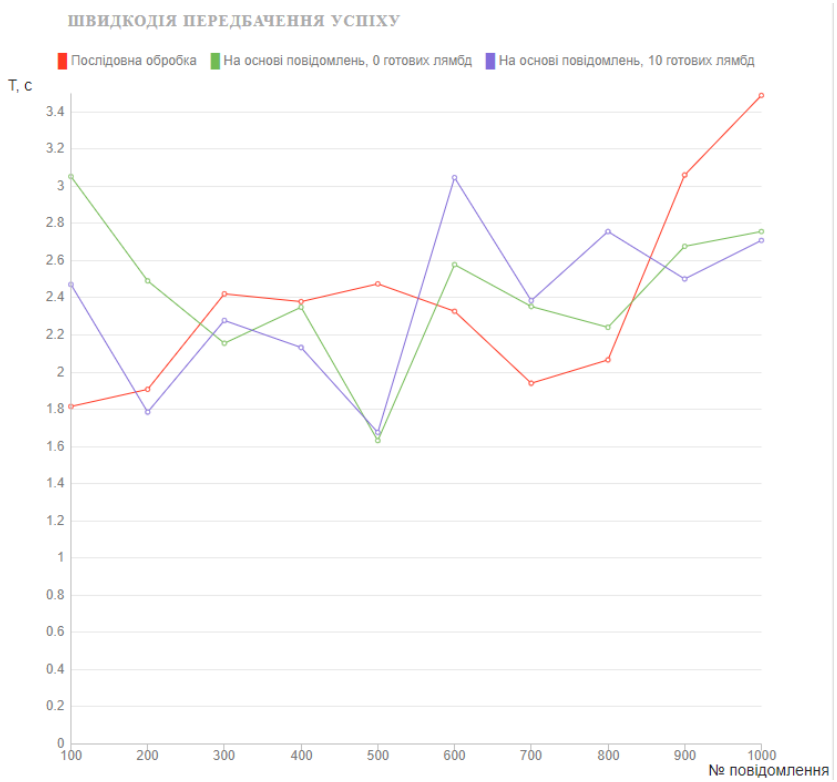


Рис. 4.7. Середні показники швидкодії конфігурацій для 500 повідомлень

Окремо слід також зазначити, що за виконання тестів інфраструктуру було налаштовано таким чином, щоб максимальна кількість екземплярів лямбди, запущених водночас, не перевищувала 10. Це пов’язано з тим, що під час розробки використовувався кластер “AWS Aurora”, конфігурація якого не дозволяє ефективно надавати велику кількість з’єднань, що було помічено за підготовчого виконання тестів, де значним фактором у часі виконання функцій стали операції запису в базу даних. В комерційних же умовах можемо розраховувати на наявність більших обчислювальних ресурсів, внаслідок чого швидкодія системи додатково збільшиться, порівняно з показниками, наведеними на рис. 4.7.

4.3. Висновки до розділу

В даному розділі було проведено аналіз доцільності використання обраного методу, спираючись на порівняння з аналогічними методами. Зокрема, було виконано тестування точності та швидкодії для базових моделей класифікації зборів, а також тестування швидкодії прогнозування для запропонованої модифікації методу, порівняно з конфігурацією, що передбачає послідовну обробку.

Передусім, отримані результати підтвердили доречність використання градієнтного бустингу як базового методу завдяки кращій точності передбачень, порівняно з іншими методами. Водночас, швидкодія процесу навчання моделі на основі вищезгаданого методу вважаємо задовільною в межах даної роботи, попри те, що вона показує гірші показники, ніж аналоги. Тим не менш, маємо взяти до уваги, що її можна, за необхідності, покращити за допомогою оптимізації залежностей контейнера додатку, так і за використання конфігурації функції, що передбачає більші обчислювальні спроможності. Отже, за подальших розробок та комерційного розгортання системи слід врахувати потенційні дії з оптимізації лямбди навчання моделі.

Додаткове тестування швидкодії процесу ж довело доцільність використання розподілених обчислень задля прискорення процесу прогнозування успіху зборів за великого (10 і більше запитів у межах секунди) обсягу оновлень зборів у системі. Так, залежно від обраної конфігурації та кількості зборів, що очікують на обробку, спостерігаємо від двох- до десятикратного збільшення швидкодії за рахунок паралельної обробки повідомлень, на противагу послідовній. Також, менш очевидне покращення полягає в тому, що розподілення системи таким чином значно збільшує можливості її масштабування, оскільки кількість обробників є гнучкою та може регулюватись залежно від очікуваного навантаження; в той час, як час послідовної обробки зростає лінійно, залежно від кількості повідомлень, та може масштабуватись лише за рахунок виділення додаткових обчислювальних ресурсів.

Водночас, результати тестування показали наявність недоліку в конфігурації, що завжди передбачає створення нових екземплярів лямбди, в тому, що час холодного старту кожної з них є доволі високим, порівняно зі стандартним часом безпосередньої обробки повідомлення [31]. Достеменно знаємо, що існують методи оптимізації даної фази виконання коду за допомогою оптимізації розміру додатку, отже подальші розробки за темою роботи мають їх передбачити задля забезпечення достатніх показників нефункціональних вимог для комерційного розгортання системи.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

5.1. Опис проблеми

З початком повномасштабної війни на території України, гостро, як ніколи раніше, постає проблема матеріального забезпечення, необхідного для функціонування багатьох ланок нашого суспільства – починаючи з матеріально незахищених прошарків, та закінчуючи Збройними Силами України. Так, користуючись досвідом, набутим ще з 2014 року, розквіту набув волонтерський рух: небайдужі співвітчизники залучають кошти для підтримки захисників та постраждалих, зокрема надаючи значну допомогу у матеріальне забезпечення наших воїнів.

Водночас, волонтерська діяльність в цілому бурхливо розвивається, запроваджуючи шляхи для фінансування проєктів як всередині держави, так і з закордону. Як наслідок значних потреб на фронті та залучення великої кількості людського ресурсу до пошуку та закупівлі необхідних матеріалів та засобів, маємо значну кількість волонтерських зборів, що оголошуються щодня. Цілком природньо, що як у власників зборів, так і бенефіціарів закуплених під час них товарів, так і вкладників у закупівлі виникають у процесі певні складнощі, що не було виправлено за поточної системи.

Передусім, волонтерам складно залучити достатню кількість коштів для закриття оголошених зборів. Так, повідомлення про них ширяться, переважно, через соціальні мережі, через що аудиторія вкладників обмежена, переважно, колом знайомих або ж власними підписниками. Отже, волонтеру, що не веде належну роботу з громадськістю, значно складніше збирати кошти, аніж блогеру з великою аудиторією.

Також слід зазначити, що процес є доволі незручним і для потенційного вкладника. Так, більшість населення не має змоги активно стежити за соціальними мережами та докладати зусиль до пошуку зборів. Вочевидь, в цьому питанні вирішальну роль відіграють обмеженість самостійного пошуку інфлюенсерів або ж окремих організацій та вибору

серед оголошених ними зборів на потреби, що є найактуальнішими саме для користувача.

Слід також зазначити певну хаотичність у процесі оплати зборів. Так, в той час, як не оголошеним стандартом є ведення “Банок” від “Monobank” задля прозорості та зручності, через певні обмеження волонтери можуть бути змушені збирати кошти за реквізитами банківського рахунку, тощо. Це проблематично для деяких користувачів через низку причин, найвагомішими серед яких є неможливість використання наданих платіжних систем (переважно, платниками з закордону) та незручність оформлення переказу.

Важливою для громадян також є прозорість ведення зборів, що не має жодного централізованого аудиту. Так, важко довірити власні кошти незнайомій людині, що не має досвіду волонтерства та не може надати фізичні докази їх використання за призначенням. Задля покращення ситуації, волонтери мають за звичку документувати витрати: вести паперові звіти з матеріальних витрат, а також публікувати фотозвіти з підтвердженням передачі придбаних товарів бенефіціарам. Тим не менш, їх доволі важко відстежити самотужки, а закупівлі, що ще не було завершено, – й зовсім неможливо перевірити через відсутність матеріального результату. Ці чинники спонукають потенційних меценатів утриматись від вкладання коштів, або ж зменшити розмір внеску до такого, що не є істотним для них.

Зрештою, проблеми виникають і через оцінки ризику закриття збору з недостатньою сумою – як у його власника, так і меценатам, що хотіли б зробити внесок істотного розміру, сподіваючись на успішну закупівлю. Так, через спосіб поширення інформації про збір серед аудиторії, важко достовірно оцінити, яку аудиторію він дійсно охопив, і яка її частка готова внести кошти. Водночас, власники бізнесу, готові до співпраці у закупівлях, прагнуть мати відомості про успішність конкретного волонтера, перед початком співпраці з ним – вочевидь, минулі не закриті збори не нададуть

впевненості у подальшій співпраці. Зрештою, це значить, що без детального пошуку історії зборів волонтера та ґрунтовного її опрацювання аналітиком, важко з будь-якою впевненістю передбачити, чи охоплення аудиторії достатнє задля ведення масштабних зборів та, як наслідок, інвестицій бізнесу в них.

Отже, в наших реаліях виникла реальна необхідність у створенні програмного забезпечення, що оптимізує процес створення, ведення і керування зборами, звітування про них з аналізом ризиків поточних, спираючись на попередні проєкти волонтера.

Усі вищезгадані проблеми було узагальнено у схемі “Дерево проблем”, наведеній на рис. 5.1.

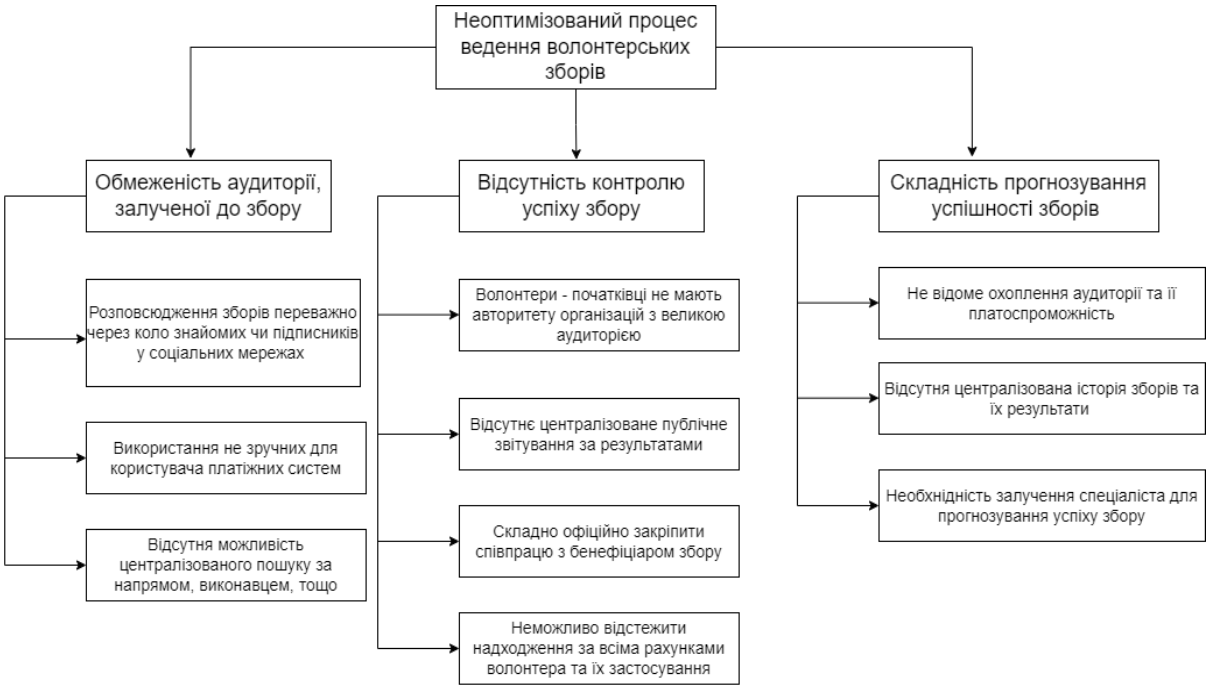


Рис. 5.1. Дерево проблем

5.2. Зацікавлені сторони

В межах даної роботи, спершу необхідно виділити сторони, зацікавлені у створенні системи, а також сфери їх зацікавленості у проєкті. Так, результати проведеної з виявлення зацікавлених сторін роботи наведено далі (табл. 5.1).

Таблиця 5.1

Зацікавлені сторони та їхні інтереси

№	Зацікавлена сторона	Сфера зацікавленості
1	Розробники	З огляду на ситуацію, що сталась навколо війни та сфери волонтерських зборів, кожен розробник особисто зацікавлений в успішності системи та, зокрема, якнайшвидшому задоволенню потреб військових.
2	Волонтери	Створена система полегшить просування зборів до аудиторії, збільшуючи обсяги зібраних коштів та, відповідно, обсяг матеріальної допомоги, що буде надано волонтером. Також система допоможе централізувати зібрані кошти, уникаючи потребу власноручного використання низки платіжних систем і банківських рахунків. Водночас, волонтери отримають оцінку ризикованості оголошених зборів, завдяки чому зможуть зосередитись на власному профілі закупівель, що найдоречніше організувати.
3	Благодійні фонди	Система надасть фондам додаткову платформу для збору коштів для окремих потреб, забезпечуючи спосіб для просування профільних зборів.
4	Меценати – фізичні особи	Користувачі зможуть інтуїтивно шукати збори та отримувати рекомендації з закупівлями за напрямками, що цікавлять їх особисто.
5	Меценати – юридичні особи	Організації зацікавлені в аудиті волонтерів, з якими взаємодіятимуть: мати формальне підтвердження їх співпраці з бенефіціарами; певні метрики успішності та ризикованості зборів, за якими можна судити про доцільність інвестицій у них.

Продовження табл. 5.1

№	Зацікавлена сторона	Сфера зацікавленості
6	Військові	Реєстрація військових угруповань як бенефіціарів та підтвердження ними співпраці з волонтерами спростить бюрократичні труднощі, пов’язані з формальним закріпленням відносин через окремі документи.
7	Цивільні бенефіціари	Індивідуальні фізичні особи або організації, що отримують допомогу, матимуть змогу закріпити матеріальні відносини з волонтером через розроблювану систему в той час, як зробити це офіційними документами проблематично або ж неможливо.
8	Постачальники продукції	Постачальникам зручно отримувати конкретні чисельні дані про наявні на ринку потреби через збори на певні товари та укладати угоди про постачання з волонтерами, що мають історію успішних закупівель, а отже, є більш надійними партнерами для тривалої співпраці.

Далі необхідно розрахувати силу впливу кожної окремої зацікавленої сторони. Аналогічно, розрахунки наведено далі (табл. 5.2).

Таблиця 5.2

Якісний аналіз зацікавлених сторін

№	Зацікавлена сторона	В	З	Р=В*З
1	Розробники	7	10	70
2	Волонтери	10	10	90
3	Благодійні фонди	5	5	25
4	Меценати – фізичні особи	9	9	81

Продовження табл. 5.2

№	Зацікавлена сторона	В	З	Р=В*З
5	Меценати – юридичні особи	8	7	56
6	Військові	10	8	80
7	Цивільні бенефіціари	5	6	30
8	Постачальники продукції	3	5	15

Надалі, спираючись на отримані чисельні результати, необхідно розробити стратегію взаємодії з кожною зі сторін. Так, було обрано такі підходи:

- Розробники мають доволі великий вплив на розроблюваний продукт, який було оцінено чисельно як 70. Так, це значить, що керівництво проекту має доволі тісно вести з ними комунікацію, заохочувати до якісного виконання посадових функцій та творчого підходу задля задоволення потреб користувацької бази. Водночас, цей ентузіазм не має диктувати бізнес-вимоги до проекту так, як це робитимуть інші сторони, тож, ймовірно, слід меншою мірою орієнтуватись на їх погляди, обираючи стратегічний напрямок розвитку проєкту.
- Волонтери мають найбільший можливий вплив на продукт, оскільки остаточна його мета – саме задоволення їх потреб і, як наслідок, суттєве збільшення ефективності процесу волонтерських зборів. З огляду на це, необхідно вести з ними дуже тісну співпрацю, залучати їх до обговорення наступних кроків та спиратись на їх побажання при розширенні системи новою функціональністю.

- Вочевидь, благодійні фонди не є пріоритетною стороною при виборі подальшого напрямку розвитку проєкту, оскільки вони, ймовірно, вже мають власні засоби поширення інформації, інтеграції з платіжними системами та угоди про співпрацю з комерційними організаціями. Попри те, що їх залучення до проекту не є пріоритетним, формальні комунікації все ж налагодити необхідно з перспективами на подальшу співпрацю та розповсюдження певних зборів саме через створену систему.
- Меценати – фізичні особи є чи не найвпливовішою стороною для системи. Так, вона напрямлена на спрощення процесу пошуку зборів та сплати вкладів у них, через що необхідно забезпечити максимальний комфорт для якомога більшої аудиторії користувачів. З огляду на це, керівництву необхідно вести тісний діалог з громадськістю, зважати на її побажання при реалізації нової функціональності системи.
- Дещо меншу силу впливу мають меценати – організації. Так, безумовно, вони зацікавлені в проекті, тож керівництво має слідкувати за задоволенням їх потреб. Втім, співпраця з ними має дещо менший пріоритет, тож необхідно продовжувати перемовини з метою більшого їх залучення в майбутнє проєкту.
- Військові мають великий вплив на майбутнє проєкту як бенефіціари від більшості волонтерських зборів, що буде розміщено на даній платформі. Так, необхідно вести тісний діалог з підрозділами різних рівнів – від взводів і до вищого керівництва – з метою полегшення їх роботи з системою, зокрема, з метою закріплення взаємодії з волонтерами та наявних потреб, що необхідно закрити.
- Вочевидь, існує й прошарок бенефіціарів – цивільних, що також прагнуть закрити певні потреби через розроблену систему. Втім, через поточне зосередження саме на військовій тематиці,

вважаємо, що вони мають доволі низький вплив на платформу. Так, з ними необхідно вести комунікації задля подальшого залучення їх до роботи системи у більших обсягах та поступового розвитку з метою задоволення їхніх потреб, проте це не є пріоритетом наразі.

- Найменший вплив на систему мають саме постачальники товарів, що закуповують волонтери. Вочевидь, вони мають доволі пасивну роль спостерігача за станом ринку з метою визначення наявних потреб у товарах, а також потенційних клієнтів для закупівель. Наразі плануємо лише вести спостереження за їх діями у відповідь на стан платформи та вимогами, що при цьому можуть виникнути, проте про офіційну співпрацю наразі мова не йде.

5.3. Інноваційне програмне рішення

Відповідно до вищезазначених проблем та інтересів зацікавлених сторін, можемо визначити основні характеристики кінцевого продукту, що буде створено в межах даної роботи.

Так, створена система, передусім, складатиметься з вебдодатку, підтримуваного окремим API-сервісом для бекенду, через який відбуватиметься взаємодія з користувачем. Цей додаток покликаний забезпечити потреби меценатів та волонтерів: так, буде надано таку основну функціональність, як:

- створення зборів;
- пошук та перегляд зборів користувачами;
- перегляд волонтерами власної статистики та передбаченої успішності збору.

Особливу увагу буде приділено саме аналітичній функціональності зборів. Так, окрім можливості переглянути орієнтовну динаміку поточних зборів, волонтери також можуть переглянути додаткову аналітику минулих зборів. До неї належатиме як безпосередньо архів зборів та їх результатів, так і перелік рекомендацій до кожного з них, як-от: виділення зборів, що

мали в цілому нереалістичні очікування – завеликий обсяг коштів чи замалий часовий проміжок; збори, що виконувались за непопулярною тематикою; збори, що відбувались за несприятливих з певних причин умов на ринку; та ін. Дана функціональність покликана заохотити волонтерів навчатись на помилках та зосередитись на профілі зборів, що є більш успішними, а отже, й більш корисними для суспільства.

Дане програмне рішення має бути доступним якомога ширшому колу користувачів, отже використання не має потребувати додаткового спеціалізованого навчання, високих системних вимог чи встановлення додатку на користувацький пристрій. Отже, на даному етапі система взаємодіятиме з користувачами через вебдодаток, доступний через веббраузер як на домашніх, так і на мобільних пристроях.

Належну увагу також буде приділено безпеці особистих даних користувача. Так, вочевидь, робота зі зборами потребує надання користувачеві можливості виконання плати. Отже, разом з інтеграцією з низкою платіжних систем, автоматично необхідно буде виконати їх стандарти до обробки персональних даних та забезпечення зашифрованості платіжних реквізитів користувача.

Наразі на ринку відсутні програмні рішення, спроможні передбачити успішність волонтерського збору – власне, відсутні й системи, що мають централізовану базу даних про поточні збори та, відповідно, достовірну інформацію про динаміку на ринку. Так, існують програмні окремі програмні методи для передбачення успішності фандрейзингових кампаній [32], які, тим не менш, не можуть бути застосовані у комерційному продукті у напрямку волонтерського руху без належних змін.

Водночас, програмні засоби, що будуть застосовані при реалізації, вже мають аналоги в інших галузях, зокрема, на ринку продажу нерухомості. Втім очевидно, що царина волонтерських зборів потребує врахування низки інших факторів, тож не є відповідним наявним рішенням.

Також система міститиме компонент оновлення наявної інформації з прогнозування. Так, він буде виконаний у формі програми, що запускається за розкладом та виконує переобчислення наявних прогнозів з урахуванням інформації (нових зборів, динаміки переказів) за останній період. Вочевидь, буде забезпечено масштабування даної програми з метою утримання належного рівня продуктивності, залежно від активності системи та кількості користувачів. Завдяки цьому користувачі повсякчас матимуть актуальну інформацію в узгоджені терміни, згідно з якою зможуть робити більш виважені рішення про оголошення зборів чи їх фінансування.

5.4. Комерційний програмний продукт. Конкурентні переваги програмного продукту

Більшість волонтерів діють невеликими командами на неоплачуваних засадах. Внаслідок орієнтації на закриття всіх тих потреб, що наявні в бенефіціарів, з якими відбувається співпраця, вони не мають змоги орієнтуватись на стан ринку, через що певні збори просуваються занадто повільно, з ймовірністю закриття у неповному обсязі. Розроблена система покликана ідентифікувати такі збори та попередити волонтерів про це заздалегідь, адже витрачені таким чином час та кошти можуть бути витрачені зі значно більшою користю – на збір в іншому напрямку, що має всі шанси задовольнити потреби вчасно.

Водночас, пряма взаємодія представника бенефіціара спроможна не лише верифікувати діяльність низки волонтерів, а й дати чіткий сигнал про наявні на ринку потреби.

Для звичайного ж користувача, що прагне матеріально підтримати волонтерську ініціативу, розроблена система істотно спрощує вибір збору для підтримки. Так, на противагу поширеній наразі потребі пошуку зборів самотужки у публікаціях в соціальних мережах профільних волонтерів чи фондів, система надає функціональність пошуку за інтересами. Водночас, з метою збереження коштів користувача і використання їх за призначенням,

при пошуку більшу релевантність матимуть збори верифікованих волонтерів, що мають певну репутацію, підкріплену звітами, а також ті, що потребують термінового добору коштів через гостру потребу бенефіціара.

Отже, конкурентними перевагами вебдодатку централізованої платформи для розміщення волонтерських зборів є:

- Оптимізація процесу розміщення волонтерами зборів та їх відстежуваності за рахунок оновлення опису призначення та звітування за результатами.
- Наявність інтелектуального фільтрування зборів, спираючись на вподобання за напрямком та історію попередніх вкладів. Водночас, підтримуватиметься система репутації волонтерів, згідно якої потенційно недобросовісні матимуть менший пріоритет при виведенні результатів пошуку, таким чином забезпечуючи подальше застосування коштів вкладника за призначенням. Водночас, користувачів буде заохочено залучатись до зборів волонтерів, що успішно пройшли аудит діяльності – мають документальне підтвердження взаємодії з бенефіціаром та сумлінно звітують про результати закупівель.
- Надання волонтерам програмної можливості прогнозування успішності зборів, а також отримання порад стосовно попередніх. Так, це допоможе їм надалі вести більш перспективні збори, знайшовши певну нішу серед спеціалізацій. Водночас, це значною мірою підтримає діяльність індивідуальних активістів, що не мають матеріальної можливості залучити аналіз відповідних чинників спеціалістом.

5.5. Ринок аналогічних програмних рішень

Задля сегментації ринку, перш за все, слід поділити частини системи, що просуватимуться окремим сегментам клієнтів. Так, можемо виділити такі призначення використання системи:

1) ведення волонтерських зборів;
2) вкладення коштів у збори.

Так, перший спосіб використання притаманний таким сторонам:

- індивідуальні волонтери, що створюють збори;
- волонтерські організації та благодійні фонди, що створюють збори;
- бенефіціари зборів.

Почнімо аналіз з першої групи споживачів – індивідуальних волонтерів. Так, система розробляється з урахуванням, передусім, їх потреб. Наразі не існує жодної централізованої системи оголошень про відкриття зборів та їх поширення, тому цьому класу користувачів доводиться з цією метою взаємодіяти з аудиторією через соціальні мережі, найпоширеніші серед яких – “Telegram”, “Twitter”, “Meta”. Водночас, вони не надають жодної функціональності донесення інформації потенційним меценатам чи навіть доступного пошуку за спеціалізацією, як це робитиме створена система.

В цілому, волонтеру важко розпочати свою діяльність, адже для цього необхідно мати наявну аудиторію, а також володіти певними знаннями про наявні потреби (зокрема, у війську), що потрібно забезпечувати новими зборами. Вочевидь, він також не спроможний залучитись допомогою аналітика – професіонала, праця якого передбачає належну грошову винагороду, що міг би вказати пріоритетні напрямки закупівель та їх розмір, який волонтер може реалістично розраховувати успішно закрити. Розроблювана система покликана спростити процес входу в галузь волонтерства та нівелювати наведені проблеми за допомогою

функціональності пошуку зборів клієнтом та передбачення успішності збору.

Також розраховуємо на певну співпрацю з уже наявними фондами та волонтерськими організаціями. Вочевидь, наявні організації вже мають певні канали розповсюдження інформації про збори, а їх структура передбачає наявність аналітичного відділу, що має уявлення про поточну ситуацію на ринку та, відповідно, може передбачити успішність збору. Тому наведена вище вигода для індивідуальних волонтерів не є для організацій нагальною. Водночас, вважаємо, що наявність альтернативного джерела поширення інформації та аналітики з успішності зборів є привабливою в довгостроковій перспективі, отже цілком логічно розраховувати на глибшу взаємодію з цими організаціями (спільні проекти, тощо).

Логічно, що в системі окрема роль виділяється бенефіціарам зборів, вони не мають потреби в пошуку чи сплаті внесків у збори, як і в створенні зборів заради своєї вигоди. Натомість, вони виконують наглядову роль в системі: підтвердження взаємодії з волонтерами та дійсності виконання ними матеріальних зобов'язань. Наведені вище платформи не мають такої можливості, тож дана можливість суттєво спрощує бюрократичні процеси, надаючи допоміжний спосіб верифікації волонтерства, паралельно з оформленням договорів, тощо.

Другий же спосіб взаємодії із системою притаманний таким сторонам:

- вкладники – фізичні особи;
- вкладники – організації.

В цілому, потреби фізичних осіб є вкрай важливими для системи. Так, критично необхідно забезпечити їм графічний інтерфейс, що надає змогу легко знайти збори та сплатити кошти. Наразі, в цілому, відсутні системи, що надавали б аналогічні програмні можливості, тож користувачеві лишається тільки власноруч шукати інформацію в мережі Інтернет. Особливу увагу звернено на матеріальну відповідальність перед користувачем – так, система навмисно просуває волонтерів з кращою

репутацією, задля уникнення ситуації, коли волонтер не є добросовісним та внесені меценатами кошти використано не за призначенням.

Передбачається, що основний прибуток системи надходитиме саме з комісії сервісу при внесенні фізичними особами коштів на збір.

Передбачено окрему роботу з вкладниками – організаціями. Так, очікуємо, що певна їх частина вже має попереднє уявлення про збори, для яких готові надати кошти. Тим не менш, відбуватиметься і співпраця з підприємцями, для яких важливою буде консультація представників системи. Так, вона полягатиме у наданні інформації про передбачувану динаміку окремих зборів (за згоди волонтерів-власників) задля вибору пріоритетнішого – того, в який вкласти кошти найдоцільніше.

5.6. Унікальна ціннісна пропозиція

Основна ціннісна пропозиція проекту для волонтерів має декілька проявів, а саме:

1. Надана централізована платформа для розміщення зборів суттєво зменшує потребу в самостійному їх просуванні до цільової аудиторії. Так, вочевидь, волонтери все рівно зацікавлені в зростанні власної аудиторії поза платформою та власному донесенні оголошень про збори до них, але це не є обов’язковою запорукою успіху збору. Натомість, система суттєво спрощує бар’єр для входу ентузіаста до волонтерської діяльності.
2. Верифікованість співпраці волонтера з бенефіціаром у системі суттєво спрощує процес надання відповідних доказів зацікавленим меценатам, внаслідок чого, цілком закономірно, збільшується частина аудиторії, що має довіру до збору та готова вкласти в нього кошти.
3. Система репутації, наявна у системі, покликана заохочувати волонтерів вести діяльність добросовісно, а саме вести збори, на

які реалістично спроможні зібрати кошти, а також ретельно звітувати про закриті закупівлі.

4. Зрештою, вбудована функціональність прогнозування успішності зборів є унікальною для волонтерської галузі. Так, вигода, яку отримують волонтери, полягає у автоматичному спрямуванні їх діяльності в оптимальному напрямку, за якого вони зможуть максимальним чином задовольнити потреби бенефіціарів, з якими вони співпрацюють.

Водночас, основну унікальну ціннісну пропозицію пропозицію для користувача-вкладника становить система репутації для волонтерів, що складається з успішності їх попередніх зборів, наявності звітності та верифікації бенефіціара. Так, висока оцінка даного показника гарантує легкий доступ користувача до зборів даного волонтера, водночас забезпечуючи певну гарантію оптимального використання його коштів саме за призначенням.

5.7. Потоки доходів. Структура витрат

Згідно з поточним планом, дохід з надання системи у використання користувачам складатиметься з комісії з кожного платежа, виконаного для зборів. Отже, планові доходи, структура витрат та прибуток на наступні 3 роки наведено далі.

Таблиця 5.3

План доходу від продажу товарів та послуг на 2024 р.

№	Назва статті доходу	1 кв. 2024 р.	2 кв. 2024 р.	3 кв. 2024 р.	4 кв. 2024 р.	Всього
1	Середній вклад ($)	1	2	4	5	
2	Кількість транзакцій (тис.)	10	100	500	1000	1610

Продовження табл. 5.3

№	Назва статті доходу	1 кв. 2024 р.	2 кв. 2024 р.	3 кв. 2024 р.	4 кв. 2024 р.	Всього
3	Комісія від транзакції (%)	1	1	1	1	
4	Виторг від комісії (тис. $)	0.1	2	20	50	72.1

Таблиця 5.4

Структура та розрахунок витрат (тис. $) на 2024 р.

№	Назва статті витрат	1 кв. 2024 р.	2 кв. 2024 р.	3 кв. 2024 р.	4 кв. 2024 р.	Всього
1	Заробітна плата	10	10	10	10	40
2	Оренда приміщень	2	2	2	2	8
3	Маркетинг та реклама	3	3	3	3	12
4	Операційні витрати	2	2	2	2	8
5	Всього витрат (без оподаткування)	17	17	17	17	68

Таблиця 5.5

Прибуток за 2024 р.

№	Назва статті доходу	1 кв. 2024 р.	2 кв. 2024 р.	3 кв. 2024 р.	4 кв. 2024 р.	Всього
1	Прибуток	-16.9	-15	3	23	4.1

Таблиця 5.6

План доходу від продажу товарів та послуг на 2025 р.

№	Назва статті доходу	1 кв. 2025 р.	2 кв. 2025 р.	3 кв. 2025 р.	4 кв. 2025 р.	Всього
1	Середній вклад ($)	5	6	7	5	

Продовження табл. 5.6

№	Назва статті доходу	1 кв. 2025 р.	2 кв. 2025 р.	3 кв. 2025 р.	4 кв. 2025 р.	Всього
2	Кількість транзакцій (тис)	1200	1500	1700	2000	6400
3	Комісія від транзакції (%)	1	1	1	1	
4	Виторг від комісії (тис. $)	60	90	119	100	369

Таблиця 5.7

Структура та розрахунок витрат на 2025 р.

№	Назва статті витрат	1 кв. 2025 р.	2 кв. 2025 р.	3 кв. 2025 р.	4 кв. 2025 р.	Всього
1	Заробітна плата	12	12	15	15	54
2	Оренда приміщень	3	4	4	5	16
3	Маркетинг та реклама	3	4	5	4	16
4	Операційні витрати	3	4	4	5	16
5	Всього витрат (без оподаткування)	21	24	28	29	102

Таблиця 5.8

Прибуток за 2025 р.

№	Назва статті доходу	1 кв. 2025 р.	2 кв. 2025 р.	3 кв. 2025 р.	4 кв. 2025 р.	Всього
1	Прибуток	39	66	91	71	267

Таблиця 5.9

План доходу від продажу товарів та послуг на 2026 р.

№	Назва статті доходу	1 кв. 2026 р.	2 кв. 2026 р.	3 кв. 2026 р.	4 кв. 2026 р.	Всього
1	Середній вклад ($)	5	4	6	4	
2	Кількість транзакцій (тис)	2100	2000	1800	1850	7750
3	Комісія від транзакції (%)	1	1	1	1	
4	Виторг від комісії (тис. $)	105	80	108	74	367

Таблиця 5.10

Структура та розрахунок витрат на 2026 р.

№	Назва статті витрат	1 кв. 2026 р.	2 кв. 2026 р.	3 кв. 2026 р.	4 кв. 2026 р.	Всього
1	Заробітна плата	18	18	20	20	54
2	Оренда приміщень	5	5	5	6	16
3	Маркетинг та реклама	4	4	4	4	16
4	Операційні витрати	5	6	6	6	16
5	Всього витрат (без оподаткування)	31	27	28	29	115

Таблиця 5.11

Прибуток за 2026 р.

№	Назва статті доходу	1 кв. 2026 р.	2 кв. 2026 р.	3 кв. 2026 р.	4 кв. 2026 р.	Всього
1	Прибуток	74	53	80	45	252

Отже, робимо висновок, що, згідно з прогнозами, проект починає бути прибутковим, починаючи з четвертого кварталу першого року. Водночас,

передбачається стабільний ріст прибутку впродовж другого року з можливим частковим падінням інтересу аудиторії до системи впродовж третього кварталу.

5.8. Канва бізнес-моделі стартапу

Задля узагальнення вищезгаданих фактів про проект, було створено канву його бізнес-моделі (табл. 5.12).

Таблиця 5.12

Канва бізнес-моделі

<table>
<tr>
<td>Проблема

Обмеженість аудиторії, залученої до збору; відсутність контролю успіху збору; складність прогнозування успішності.</td>
<td>Інноваційна технологія

Прогнозування успішності зборів; сортування зборів за тематикою та репутацією волонтера.</td>
<td rowspan="2">Унікальна ціннісна пропозиція

Спрощений вхід у царину волонтерства; система репутації як заохочення до добросовісної роботи; простота використання для клієнта.</td>
<td>Зацікавлені сторони

Волонтери, меценати, благодійні фонди, бенефіціари.</td>
<td>Ринок

Відсутні аналогічні програмні рішення.</td>
</tr>
<tr>
<td>Рішення

Створення централізованої системи з прогнозуванням успіху зборів та інтелектуальним пошуком зборів.</td>
<td>Програмний продукт

Вебдодаток зі складовими для взаємодії меценатів та волонтерів.</td>
<td>Конкурентні переваги

Оптимізація процесів створення, пошуку, верифікації та прогнозування зборів.</td>
<td>Канал збуту

B2B: взаємодія з благодійними фондами та організаціями.
B2C: функціональність для роботи волонтерів та меценатів.</td>
</tr>
</table>

Продовження табл. 5.12

Структура витрат	Потоки доходів
Заробітна плата; оренда приміщень; реклама; операційні витрати.	Комісія з кожної транзакції.

Так, канва бізнес-моделі містить такі елементи:

1. Проблема. На ринку волонтерської діяльності склалась низка проблем у взаємодії між волонтерами, бенефіціарами та меценатами, що прагнуть докласти коштів до закупівель, організованих волонтерами. Основними з них є складність просування інформації про збори, відсутність установлених норм контролю успішності та виконання зборів і, зрештою, нездатність волонтерами самотужки передбачити динаміку просування збору.
2. Інноваційна технологія. При виконанні проекту буде запроваджено принципово нову для ринку функціональність: прогнозування ризику зборів, відстеження добросовісності волонтера та сортування результатів пошуку мецената, спираючись на неї.
3. Зацікавлені сторони. Основними зацікавленими сторонами проекту є меценати (як фізичні особи, так і організації), індивідуальні волонтери, благодійні фонди та бенефіціари зборів (наразі – переважно, військові).
4. Рішення. Головним рішенням при створенні проекту є розроблення архітектури програмного забезпечення, що регулярно за розкладом виконує переобчислення прогнозованої динаміки зборів, наявних наразі, спираючись на стан ринку.
5. Ринок. Для створення проекту було обрано доволі специфічний ринок волонтерської діяльності. Так, він апріорі не має комерційної конкуренції, в той час, як аналогічні програмні рішення, в цілому відсутні.

6. Програмний продукт. Створений програмний продукт виконано у формі вебдодатку з окремим програмним компонентом, відповідальним за регулярне оновлення прогнозованих даних про збори.
7. Канали збуту. Наявні канали збуту: для індивідуальний клієнтів та для організацій. Так, до першого типу належать створення зборів волонтерами та вкладання у них коштів (з урахуванням комісії) індивідуальних меценатів. Водночас, другий тип працює за схожими принципами, але за особливої взаємодії з представником проекту задля реалізації конкретних проектів.
8. Унікальні ціннісна пропозиція. Унікальна ціннісна пропозиція полягає, передусім, у створенні спрощених умов для волонтерів на початку своєї діяльності з доволі чітким орієнтиром для наступних дій у формі прогнозів успішності зборів. Водночас, сюди також наводимо спрощений процес пошуку та внесення коштів на бажані збори індивідуальних меценатів з додатковими гарантіями їх цільового застосування, завдяки системи репутації волонтерів.
9. Конкурентні переваги. Конкурентні переваги полягають в загальній оптимізації процесів волонтерських зборів, зокрема їх оцифруванні заради спрощення взаємодії. Так, на ринку відсутні системи, що надавали б меценатам зважені рекомендації зі зборів за бажаним напрямком, як і прогнози успішності зборів волонтерам.
10. Потоки доходів. Основним джерелом доходів виступає комісія до кожного платежу з внесення коштів на збір.
11. Структура витрат. Щоквартальні витрати проекту складаються з заробітної плати працівників, оренди робочого приміщення, витрат на маркетинг та, зрештою, операційні витрати, пов’язані з підтримкою сервісу.

5.9. Висновки до розділу

В даному розділі було розглянуто розроблену систему в якості стартапу. Так, це дозволило детальніше дослідити низку навколишніх чинників проекту не як ізольованого програмного забезпечення, а, радше, як комерційного продукту. Передусім, було виконано розгляд системи з точки зору споживача, було виділено наявні незадоволені потреби та співставлено їх з відповідними зацікавленими сторонами проекту. Також було виконано дослідження поточного ринку: вже наявних програмних рішень з виділенням серед них прямих конкурентів, їх недоліків та переваг проектованої системи. Зрештою, спираючись на отримані дані, було докладено зусиль до виділення саме переваг розробленої системи, її унікальної ціннісної пропозиції. Як наслідок, було виконано симуляцію обігу коштів як результату запуску системи з виділенням потенційної прибутковості проекту.

Отже, спираючись на розглянуті риси проекту як бізнес-проекту та розроблену для цього канву бізнес-проекту, можемо стверджувати, що було створено чітке уявлення про продукт як комерційний проект. Так, наведені вище прогнози прибутковості стверджують про комерційну доцільність проекту, отже маємо всі підстави виконувати реалізацію програмного рішення з орієнтацією на комерційне застосування.

ВИСНОВКИ

Метою даної магістерської дисертації була розробка програмного методу передбачення успішності волонтерських зборів. Так, вона передбачала дослідження наявних методів та алгоритмів, виділення їх переваг і недоліків та вдосконалення того, що було обрано за базовий. Очікуваним результатом при цьому було також розроблене програмне забезпечення, що спирається на згаданий метод.

У першому розділі було проведено аналіз наявних методів передбачення успішності волонтерських зборів, а також комерційних систем, що їх застосовують. Було виділено їх переваги та недоліки з метою систематизації розуміння предметної області та виявлення тих методів, використання яких є можливим для здійснення поставленої задачі, з урахуванням їх недоліків, що слід усунути за розробки власного методу.

У другому розділі магістерської дисертації було наведено модифікований програмний метод, що пропонується до втілення в межах роботи. Так, було запропоновано базовий програмний метод, перелічено його переваги, обґрунтовуючи його доречність для виконання поставленого завдання; а також названо недоліки, що необхідно усунути задля виконання завдання, разом зі шляхами їх усунення. Так, суттєва відмінність запропонованого методу від базового полягає у застосуванні паралельної асинхронної обробки оновлень про збори за допомогою архітектури, заснованої на обміні повідомлень.

У третьому розділі було здійснено опис цільової архітектури програмного забезпечення, що спирається на модифікований метод передбачення. Так, задля майбутньої масштабованості системи реалізація паралельної обробки із застосуванням повідомлень реалізована, спираючись на безсерверні сервіси “Amazon Web Services”: “Lambda” та “Simple Queue Service”. Так, вони надають можливості асинхронного надсилання повідомлень на обробку компонентою вебдодатку на подальшу

обробку в брокер повідомлень, який, у свою чергу, перенаправляє повідомлення на обробку наявним оточенням лямбди, що містить код безпосердньо обробки даних з передбаченням успішності збору. Водночас, було обрано набір технологій для реалізації коду, що передбачає інтеграцію з вищезгаданими сервісами. Запропонована архітектура забезпечує високу пропускну здатність для даних про оновлення збору з можливістю паралельної, незалежної їх обробки реалізованим кодом, чим задовольняє вимоги, поставлені до завдання, що виконується.

Беручи до уваги час обробки всіх повідомлень у наборі оновлень зборів як основний критерій оцінки швидкодії реалізованого програмного методу, у четвертому розділі було виконано аналіз та порівняння між собою архітектур, що використовує базовий метод, а також такої, що використовує вдосконалений. Попри отримані аналогічні показники часу обробки обома архітектурами для набору повідомлень, кількість яких не перевищує 10, можна стверджувати про суттєве скорочення обсягу необхідного часу для обробки більших наборів. Зокрема, запровадження згаданої архітектури забезпечує зменшення часу обробки наборів зборів кількістю понад 30, на 55-81%, порівняно з послідовною обробкою, передбаченою базовою архітектурою. Така різниця в результатах, залежно від кількості повідомлень, пояснюється холодним стартом обробників повідомлень, остаточна ініціалізація останніх серед яких у часі співпадає з завершенням обробки набору повідомлень архітектурою з послідовною обробкою. Натомість, збільшення кількості повідомлень, а також використання заздалегідь ініціалізованих оточень мінімізують частину від загального часу обробки, що виділено саме на запуск обробника. Таким чином, запропонований програмний метод є оптимізованим саме для обробки даних у сервісах, що передбачають високу їх пропускну здатність, що, без сумніву, є комерційною перевагою за поточного стану волонтерства.

П’ятий розділ даної магістерської дисертації містить опис бізнес-моделі розробленого на основі запропонованого програмного методу, як

комерційного продукту. Зокрема, було обґрунтовано наявність на ринку попиту на функціональність, що надається розробленим у межах роботи ПЗ, його цільову аудиторію та можливість отримання прибутків, надаючи сервіси з використанням згаданої функціональності. Спираючись на це, було створено канву бізнес-моделі стартапу, що передбачає виділення комерційних переваг продукту, зацікавлені сторони, можливі ринки збуту, пов’язані витрати та доходи. Спираючись на отримані результати, можна стверджувати про можливість комерційного запровадження стартапу, що спирається на розроблений метод та програмне забезпечення.

Отже, можна зробити висновок про те, що розроблений в даній роботі програмний метод передбачає суттєве покращення швидкодії обробки волонтерських зборів, за рахунок чого забезпечується велика пропускна можливість та конкурентоспроможність комерційного програмного забезпечення, створеного на його основі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Machine Learning Guide for Oil and Gas Using Python: A Step-by-Step Breakdown with Data, Algorithms, Codes, and Applications, ISBN: 0128219297,9780128219294, 2021.
2. Misik, S., Cela, A., & Bradac, Z. (2016). Distributed Systems - A brief review of theory and practice. IFAC-PapersOnLine, 49(25), 318–323.
3. Navarro, C. A., Hitschfeld-Kahler, N., & Mateu, L. (2014). A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures. Communications in Computational Physics, 15(02), 285–329.
4. Elteir, M., Lin, H., & Feng, W. (2010). Enhancing MapReduce via Asynchronous Data Processing. 2010 IEEE 16th International Conference on Parallel and Distributed Systems.
5. Hornsby, A. (2011). XMPP message-based MVC architecture for event-driven real-time interactive applications. 2011 IEEE International Conference on Consumer Electronics (ICCE).
6. Guy Süß, J., & Mewes, M. (2001). An Architecture Proposal for Enterprise Message Brokers. Lecture Notes in Computer Science, 27–42.
7. Clark, T., & Barn, B. S. (2011). Event driven architecture modelling and simulation. Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE).
8. L. Mezzalira, L. Hyatt, V. Denti, Z. Laupaj (2023). Let’s Architect! Designing event-driven architectures. Режим доступу: https://aws.amazon.com/blogs/architecture/lets-architect-designing-event-driven-architectures/.
9. AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda. AWS Whitepaper. Режим доступу: https://d1.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Architectures.pdf.
10. Scholz, M., & Wimmer, T. (2020). A comparison of classification

methods across different data complexity scenarios and datasets. Expert Systems with Applications, 114217.

11. Wang, W., Zheng, H., & Wu, Y. J. (2020). Prediction of fundraising outcomes for crowdfunding projects based on deep learning: a multimodel comparative study. Soft Computing.
12. M. S. Oduro, H. Yu and H. Huang, "Predicting the Entrepreneurial Success of Crowdfunding Campaigns Using Model-Based Machine Learning Methods," in International Journal of Crowd Science, vol. 6, no. 1, pp. 7-16, April 2022.
13. Ahmad, F. S., Tyagi, D., & Kaur, S. (2017). Predicting crowdfunding success with optimally weighted random forests. 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS).
14. Greenberg, M. D., Pardo, B., Hariharan, K., & Gerber, E. (2013). Crowdfunding support tools: Predicting success & failure. In Proceedings of Extended Abstracts on Human Factors in Computing Systems (pp. 1815-1820).
15. Yu, P.-F., Huang, F.-M., Yang, C., Liu, Y.-H., Li, Z.-Y., & Tsai, C.-H. (2018). Prediction of Crowdfunding Project Success with Deep Learning. 2018 IEEE 15th International Conference on e-Business Engineering.
16. Ahmad, F. S., Tyagi, D., & Kaur, S. (2017). Predicting crowdfunding success with optimally weighted random forests. In Proceedings of IEEE 2017 International Conference on Infocom Technologies and Unmanned Systems (pp. 770–775). IEEE.
17. Jhaveri, S., Khedkar, I., Kantharia, Y., & Jaswal, S. (2019). Success Prediction using Random Forest, CatBoost, XGBoost and AdaBoost for Kickstarter Campaigns. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC).
18. Wang, B., Wu, W., Zheng, W., Liu, Y., & Yin, L. (2020). Recommendation Algorithm of Crowdfunding Platform Based on

Collaborative Filtering. Journal of Physics: Conference Series, 1673(1), 012030.

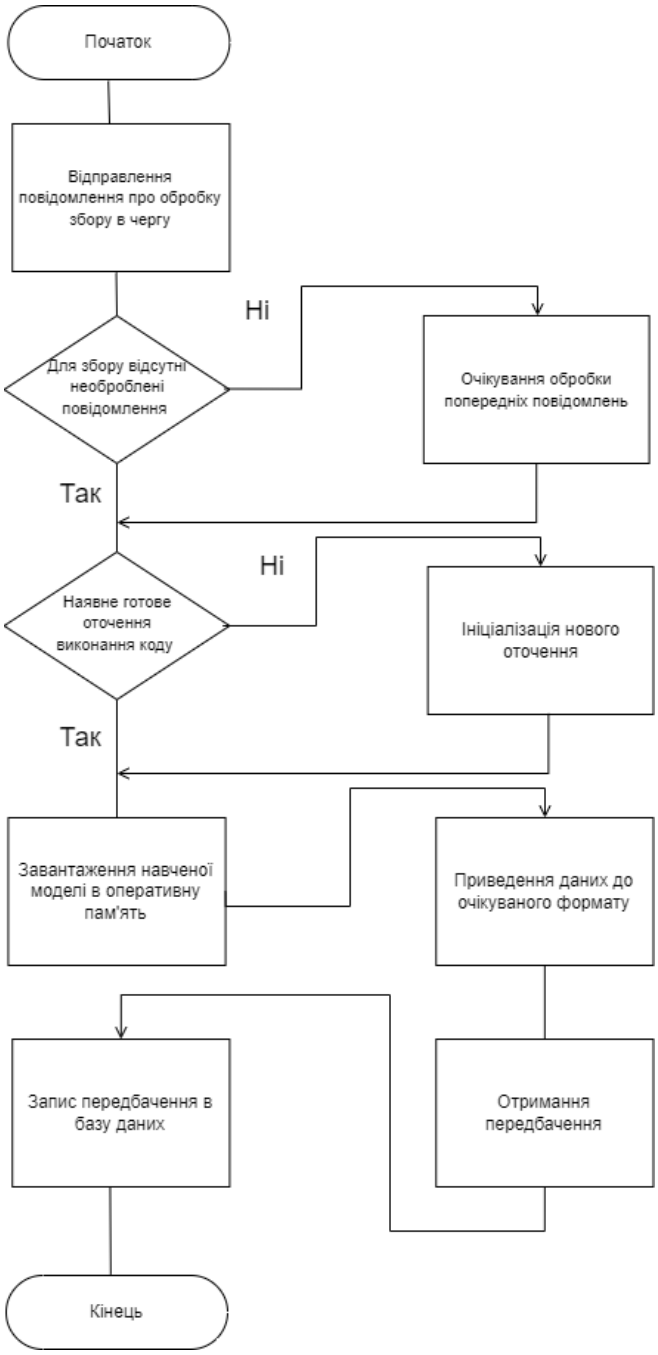
19. Lee, S., Lee, K., & Kim, H. (2018). Content-based Success Prediction of Crowdfunding Campaigns. Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW ’18.
20. Tapia, F., Mora, M. Á., Fuertes, W., Aules, H., Flores, E., & Toulkeridis, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. Applied Sciences, 10(17), 5797.
21. Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS. AWS Whitepaper. Режим доступу: https://docs.aws.amazon.com/whitepapers/latest/availability-and-beyond-improving-resilience/designing-highly-available-distributed-systems-on-aws.html.
22. Magnoni, L. (2015). Modern Messaging for Distributed Sytems. Journal of Physics: Conference Series, 608, 012038.
23. TAN, Yee Heng and REDDY, Srinivas K.. Crowdfunding digital platforms: Backer networks and their impact on project outcomes. (2021). Social Networks. 64, 158-172.
24. Chen, Kevin, Brock Jones and Isaac Kim. “KickPredict : Predicting Kickstarter Success.” (2013).
25. Li, Y.-M., Wu, J.-D., Hsieh, C.-Y., & Liou, J.-H. (2019). A social fundraising mechanism for charity crowdfunding. Decision Support Systems, 113170.
26. T. R. Mahesh, V. Vinoth Kumar, V. Muthukumaran, H. K. Shashikala, B. Swapna, Suresh Guluwadi, "Performance Analysis of XGBoost Ensemble Methods for Survivability with the Classification of Breast Cancer", Journal of Sensors, vol. 2022, Article ID 4649510, 8 pages, 2022.
27. Документація “AWS SDK for .NET”. Режим доступу: https://aws.amazon.com/sdk-for-net/.

28. High Performance Computing Lens. AWS Well-Architected Framework. Queue - Based Architecture. Режим доступу: https://docs.aws.amazon.com/wellarchitected/latest/high-performance-computing-lens/queue-based-architecture.html.
29. Набір тренувальних даних “Kickstarter Projects”. Режим доступу: https://www.kaggle.com/datasets/kemical/kickstarter-projects/data.
30. Матриця відповідності технічних характеристик віртуальних процесорів “AWS EC2” фізичним аналогам. Режим доступу: https://aws.amazon.com/ec2/instance-types/.
31. J. Dantas, H. Khazaei and M. Litoiu, "Application Deployment Strategies for Reducing the Cold Start Delay of AWS Lambda," 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), Barcelona, Spain, 2022, pp. 1-10.
32. Chung, Jinwook, "Long-Term Study of Crowdfunding Platform: Predicting Project Success and Fundraising Amount" (2015). All Graduate Theses and Dissertations. 4440.
33. Huan Liu and Dan Orban. Gridbatch: Cloud computing for large-scale data-intensive batch applications. In Cluster Computing and the Grid, 2008. CCGRID ’08. 8th IEEE International Symposium on, pages 295–305, 2008.
34. M. Zhou et al., “Project description and crowdfunding success: an exploratory study” Information Systems Frontiers, Springer, December 7, 2016, pp. 1-16.
35. Ehm F Running a Reliable Messaging Infrastructure for CERN’s Control System. Proceedings of ICALEPCS2011 (Grenoble, FRANCE).
36. P. Vahidinia, B. Farahani and F. S. Aliee, "Cold start in serverless computing: Current trends and mitigation strategies", 2020 International Conference on Omni-layer Intelligent Systems (COINS), pp. 1-7..
37. K. Solaiman and M. A. Adnan, "Wlec: A not so cold architecture to mitigate cold start problem in serverless computing" (IC2E), 2020.

ДОДАТКИ

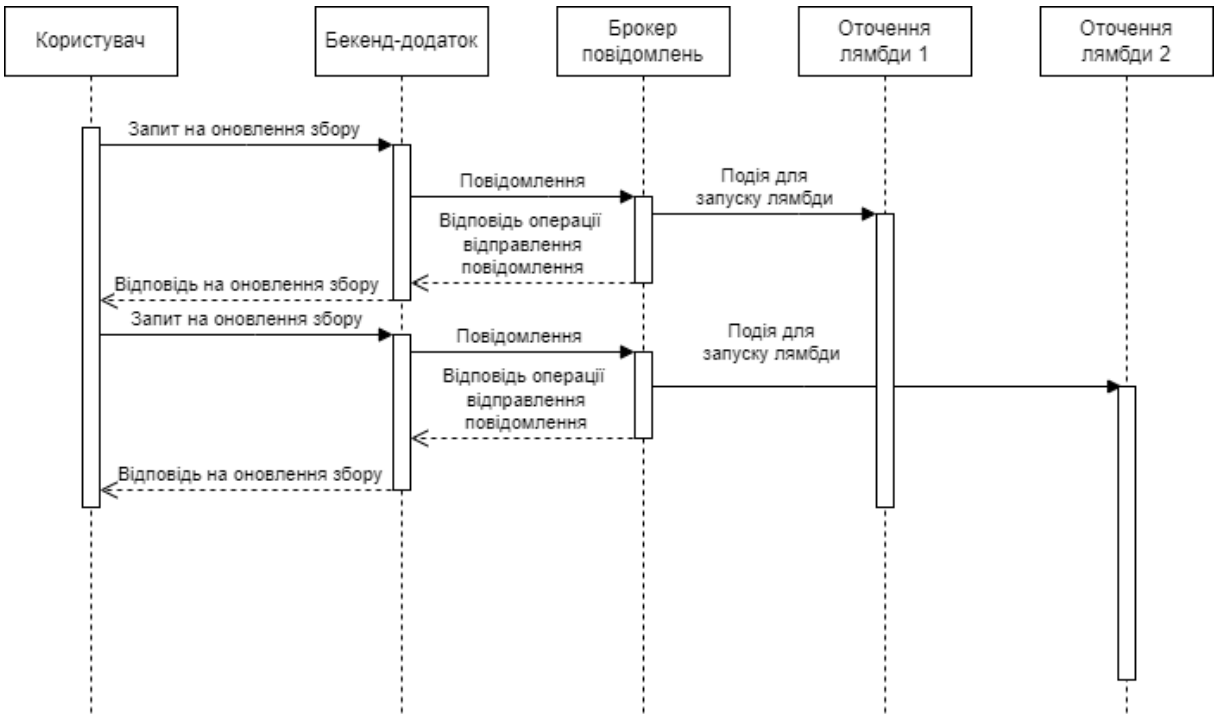
Додаток 1
Копії графічних матеріалів

Блок-схема модифікованого методу передбачення успішності зборів



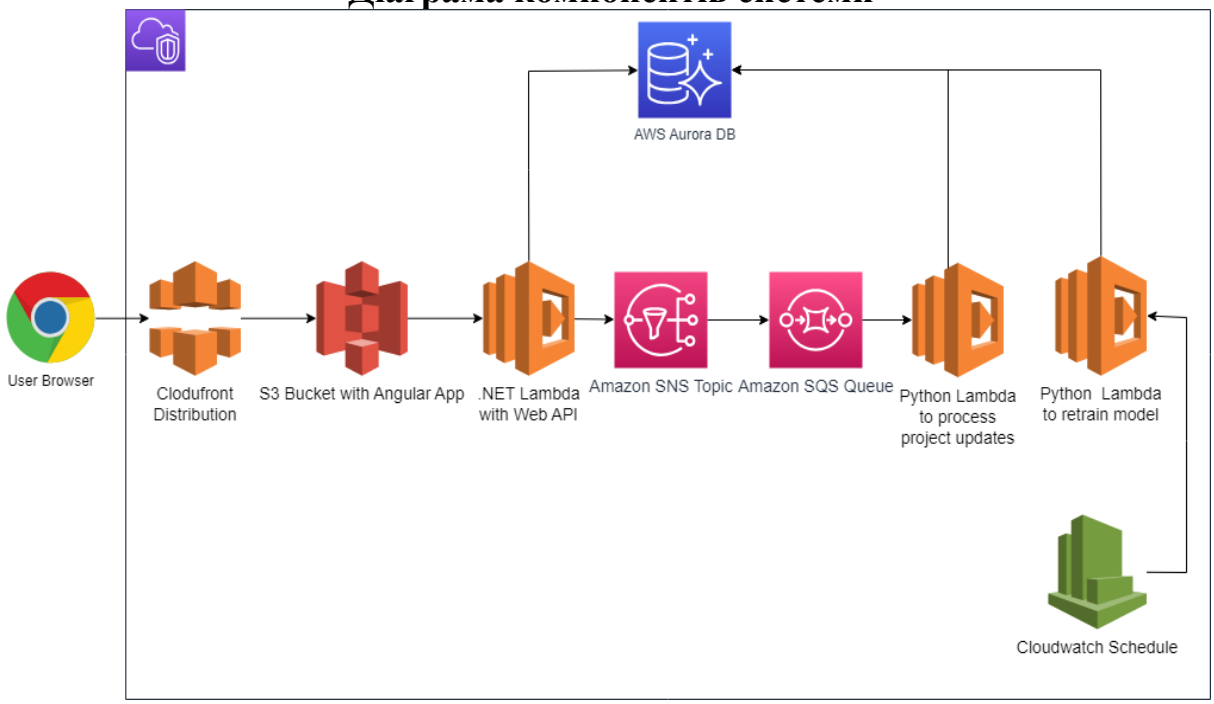
Вадим Лавріненко
група КП-21мп

Діаграма послідовності паралельної обробки двох повідомлень



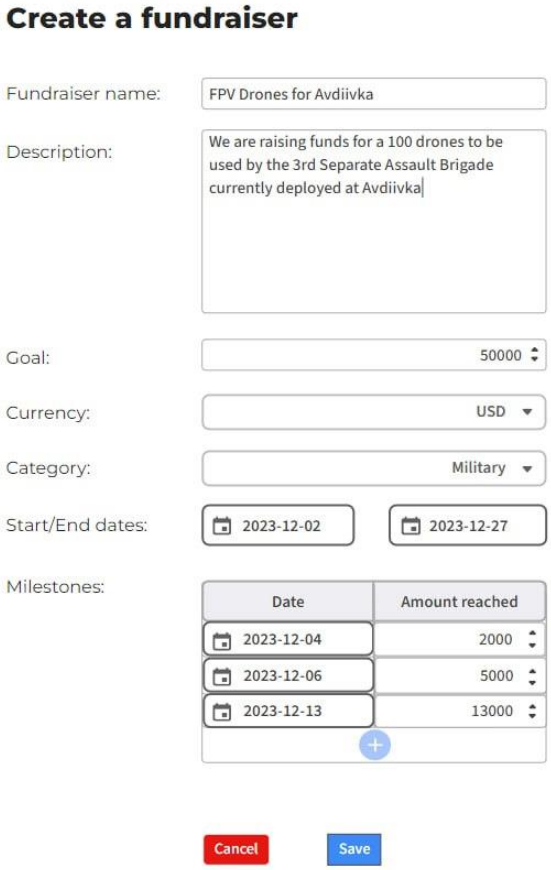
Вадим Лавріненко
група КП-21мп

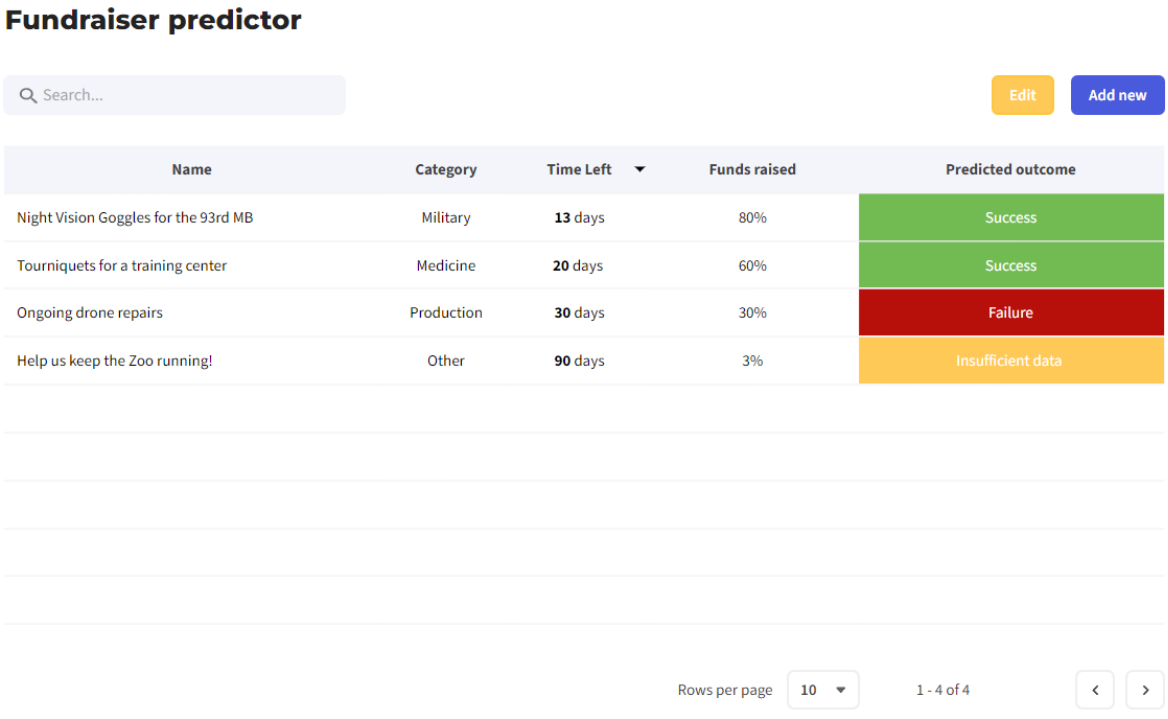
Діаграма компонентів системи



Вадим Лавріненко
група КП-21мп

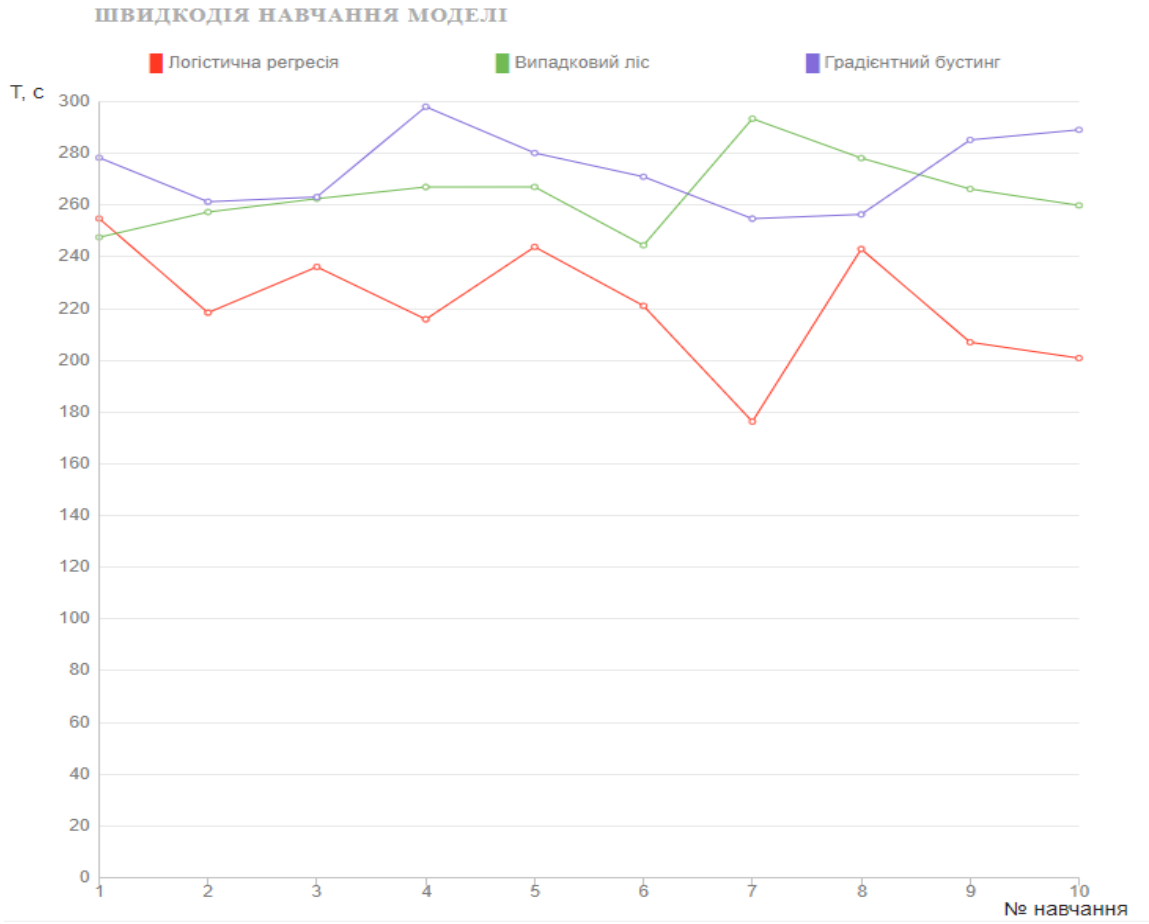
Графічний інтерфейс вебзастосунку





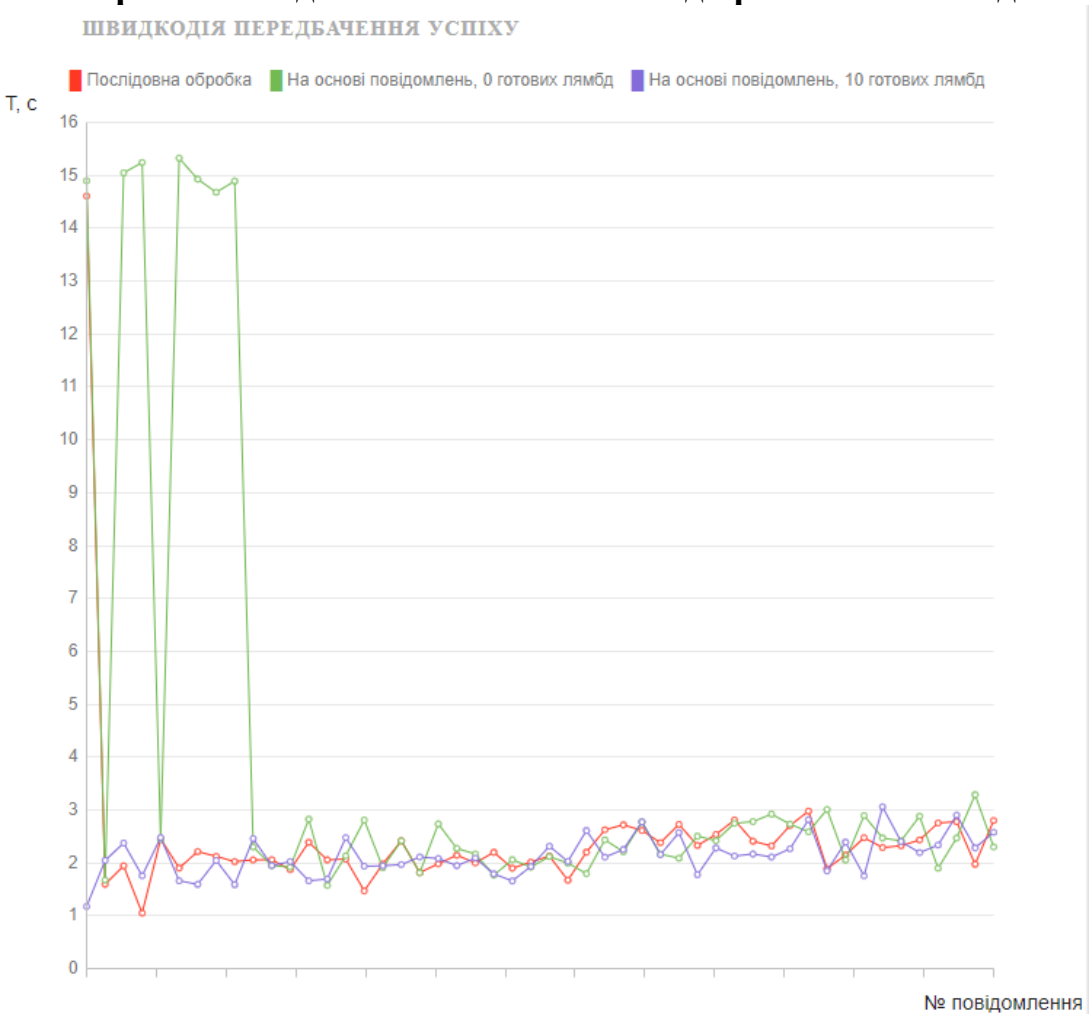
Вадим Лавріненко
група КП-21мп

Час навчання моделей, що спираються на запропоновані базові методи



Вадим Лавріненко
група КП-21мп

Час обробки повідомлень базовим та модифікованим методами



Вадим Лавріненко
група КП-21мп

Додаток 2
Фрагменти коду реалізації запропонованого методу

Лістинг 1. Код файлу training_lambda.py

```
import pandas as pd
import sklearn
from dateutil import parser
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tempfile
import boto3
import joblib
from xgboost import XGBClassifier

def date_time(df):
    days_apart = []
    launched = df["launched"].values
    i = 0
    deadline = df["deadline"].values
    for x in launched:
        days_apart.append((parser.parse(deadline[i]) - parser.parse(x)).days)
        i = i +1
    return days_apart

def lambda_handler(event, context):
    categorical_category = ["main_category" , "currency" , "state", "country"]
    features = ["main_category" , "currency" , "country","goal",
"backers","days_apart"]
    label = "state"
    df = pd.read_csv("s3://dissertation-resources/ks-projects-201801.csv")
    X = df

    df["days_apart"] = date_time(df)

    le = LabelEncoder()
    for x in categorical_category:
        df[x] = le.fit_transform(df[x].values)

    X =  df[features]
    y = df[label]
    train_x,test_x,train_y,test_y = train_test_split(X,y,test_size=0.4)

    model = XGBClassifier()
    model.fit(train_x,train_y)
    y_pred = model.predict(test_x)
    score = sklearn.metrics.accuracy_score(y_pred,test_y)
    print(score)


    s3_client = boto3.client('s3')
    bucket_name = "dissertation-resources"
    key = "xgboost.pkl"

    # WRITE
    with tempfile.TemporaryFile() as fp:
        joblib.dump(model, fp)
        fp.seek(0)
        s3_client.put_object(Body=fp.read(), Bucket=bucket_name, Key=key)
```

Лістинг 2. Код файлу classification_lambda.py

```
import json
import tempfile
import boto3
import joblib
import pandas
import os

s3_client = boto3.client('s3')
bucket_name = 'dissertation-resources'
prediction_method = os.environ['predictionMethod']
key = f'{prediction_method}.pkl'

def lambda_handler(event, context):
    with tempfile.TemporaryFile() as fp:
        s3_client.download_fileobj(Fileobj=fp, Bucket=bucket_name, Key=key)
        fp.seek(0)
        downloaded_model = joblib.load(fp)

    for record in event['Records']:
        fundraiser_json = json.loads(record['body'])
        dataframe = pandas.DataFrame({
            'main_category':fundraiser_json['main_category'],
            'currency':fundraiser_json['currency'],
            'country':fundraiser_json['country'],
            'goal':fundraiser_json['goal'],
            'backers':fundraiser_json['backers'],
            'days_apart':fundraiser_json['days_apart']}, index=[0])

        fundraiser_id = fundraiser_json['id']
        downloaded_model_prediction = downloaded_model.predict(dataframe)
        print(f'Predicted successfulness for id={fundraiser_id} is
{downloaded_model_prediction}')
```

Лістинг 3. Код файлу SqsClient.cs

```
using System.Net;
using System.Text.Json;
using Amazon.SQS;
using Amazon.SQS.Model;
using FundraiserApi.Infrastructure;
using FundraiserApi.Infrastructure.Configuration;
using Microsoft.Extensions.Options;

namespace FundraiserApi.Integration;

public interface ISqsClient<T>
{
    Task PublishAsync(T message, string messageGroupId);
}

public class SqsClient<T>: ISqsClient<T> where T : class
{
    private readonly IAmazonSQS _client;
    private readonly SqsPublisherConfiguration _configuration;

    public SqsClient(IOptions<SqsPublisherConfiguration>
publisherConfigurationOptions)
    {
        _configuration = publisherConfigurationOptions.Value;
        _client = new AmazonSQSClient();
    }

    public async Task PublishAsync(T message, string messageGroupId)
    {
        var sqsMessageRequest = new SendMessageRequest
        {
            MessageDeduplicationId = Guid.NewGuid().ToString(),
            MessageGroupId = messageGroupId,
            MessageBody = JsonSerializer.Serialize(message,
Utils.DefaultJsonSerializerOptions),
            QueueUrl = _configuration.QueueUrl
        };

        var response = await _client.SendMessageAsync(sqsMessageRequest);
        if (response.HttpStatusCode != HttpStatusCode.OK)
            throw new Exception($"Got status code={response.HttpStatusCode}");
    }
}
```

Лістинг 4. Код файлу PerformanceRunnerOrchestrator.cs

```
namespace FundraiserApi.PerformanceRunner.Services;

public interface IPerformanceRunnerOrchestrator
{
    Task ProcessEventAsync(SQSEvent.SQSMessage sqsMessage);
}

public class PerformanceRunnerOrchestrator: IPerformanceRunnerOrchestrator
{
    private readonly ISqsClient<FundraiserData> _sqsClient;
    private readonly PerformanceSettings _performanceSettings;

    public PerformanceRunnerOrchestrator(ISqsClient<FundraiserData> sqsClient,
IOptions<PerformanceSettings> performanceSettingsOptions)
    {
        _sqsClient = sqsClient;
        _performanceSettings = performanceSettingsOptions.Value;
    }
    public async Task ProcessEventAsync(SQSEvent.SQSMessage sqsMessage)
    {
        try
        {
            await ProcessEventInternalAsync();
        }
        catch (Exception e)
        {
            Console.WriteLine(e);
            throw;
        }
    }
    private async Task ProcessEventInternalAsync()
    {
        var fundraiserData = new FundraiserData
        {
            Backers = 100,
            Category = Category.Fashion,
            Country = Country.US,
            Currency = Currency.USD,
            Duration = 20,
            Goal = 40000
        };

        var publishTasks = new List<Task>();
        var useUseSingleId = _performanceSettings.UseSingleId;
        for (int i = 1; i < _performanceSettings.FundraiserCount + 1; i++)
        {
            fundraiserData.Id = i;
            var publishTask = _sqsClient.PublishAsync(fundraiserData,
useUseSingleId ? "1" : i.ToString());
            Console.WriteLine($"Published message number {i}");
            publishTasks.Add(publishTask);
        }

        await Task.WhenAll(publishTasks);
    }
}
```

Лістинг 5. Код файлу FundraiserOrchestrator.cs

```
namespace FundraiserApi.Lambda.Services;

public interface IFundraiserOrchestrator
{
    Task CreateAsync(CreateFundraiserDto request);
    Task UpdateAsync(UpdateFundraiserDto request);
    Task<IReadOnlyCollection<ViewFundraiserDto>> GetAllAsync();
}

public class FundraiserOrchestrator : IFundraiserOrchestrator
{
    private readonly IFundraiserRepo _fundraiserRepo;
    private readonly IMapper _mapper;
    private readonly ISqsClient<FundraiserData> _sqsClient;
    private readonly IFundraiserService _fundraiserService;
    private readonly PredictionSettings _predictionSettings;

    public FundraiserOrchestrator(
        IFundraiserRepo fundraiserRepo,
        IMapper mapper,
        ISqsClient<FundraiserData> sqsClient,
        IFundraiserService fundraiserService,
        IOptions<PredictionSettings> predictionSettingsOptions)
    {
        _fundraiserRepo = fundraiserRepo;
        _mapper = mapper;
        _sqsClient = sqsClient;
        _fundraiserService = fundraiserService;
        _predictionSettings = predictionSettingsOptions.Value;
    }

    public async Task CreateAsync(CreateFundraiserDto request)
    {
        var fundraiser = _fundraiserService.CreateFundraiser(request);

        var savedFundraiser = await
_fundraiserRepo.CreateFundraiserAsync(fundraiser);

        if (!_fundraiserService.IsPredictionRequired(savedFundraiser))
            return;

        var fundraiserData =
FundraiserTransformer.TransformToIntegrationModel(savedFundraiser);
        await _sqsClient.PublishAsync(fundraiserData,
savedFundraiser.Id.ToString());
    }

    public async Task UpdateAsync(UpdateFundraiserDto request)
    {
        var existingFundraiser = await
_fundraiserRepo.GetByIdAsync(request.Id);
        var updatedFundraiser = _fundraiserService.UpdateFundraiser(request,
existingFundraiser);

        var savedFundraiser = await
_fundraiserRepo.UpdateFundraiserAsync(updatedFundraiser);
```

```
        if (!_fundraiserService.IsPredictionRequired(savedFundraiser))
            return;

        var fundraiserData =
FundraiserTransformer.TransformToIntegrationModel(savedFundraiser);
        await _sqsClient.PublishAsync(fundraiserData,
savedFundraiser.Id.ToString());
    }

    public async Task<IReadOnlyCollection<ViewFundraiserDto>> GetAllAsync()
    {
        var fundraisers = await _fundraiserRepo.GetAllFundraisersAsync();
        return
_mapper.Map<IReadOnlyCollection<ViewFundraiserDto>>(fundraisers);
    }
}

public interface IFundraiserService
{
    Fundraiser CreateFundraiser(CreateFundraiserDto createFundraiserDto);
    bool IsPredictionRequired(Fundraiser fundraiser);
}

public class FundraiserService : IFundraiserService
{
    private readonly IMapper _mapper;
    private readonly PredictionSettings _predictionSettings;

    public FundraiserService(IMapper mapper, IOptions<PredictionSettings>
predictionSettingsConfig)
    {
        _mapper = mapper;
        _predictionSettings = predictionSettingsConfig.Value;
    }

    public Fundraiser CreateFundraiser(CreateFundraiserDto
createFundraiserDto)
    {
        var fundraiser = _mapper.Map<Fundraiser>(createFundraiserDto);
        var latestPledgedData =
fundraiser.Pledged.ValuesAtPointsInTime.LastOrDefault();
        if (latestPledgedData != null && latestPledgedData.Time ==
createFundraiserDto.EndDateTime)
        {
            fundraiser.Status = latestPledgedData.Value >=
createFundraiserDto.Goal
                ? Status.Successful
                : Status.Failed;
        }
        else
        {
            fundraiser.Status = fundraiser.Backers.Length() <
_predictionSettings.MinimumDataPoints
                ? Status.InsufficientData
                : Status.PredictionPending;
        }

        return fundraiser;
```

```
    }

    public bool IsPredictionRequired(Fundraiser fundraiser)
    {
        var latestPledgedData =
fundraiser.Pledged.ValuesAtPointsInTime.LastOrDefault();
        if (latestPledgedData != null && latestPledgedData.Time ==
fundraiser.EndDateTime)
        {
            return false;
        }

        return fundraiser.Backers.Length() >=
_predictionSettings.MinimumDataPoints;
    }
}
```

Додаток 3
Копія публікації

УДК 519.688

Магістрант Лавріненко В.В., к.т.н., доцент Олещенко Л.М.

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

МЕТОД ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕДБАЧЕННЯ УСПІШНОСТІ ВОЛОНТЕРСЬКИХ ЗБОРІВ

Abstract

Vadym Lavrinenko, master student; Liubov M.Oleshchenko, assoc. prof., PhD

Method and software for volunteer fundraiser success predictions

This paper deals with creating a software method capable of accurately predicting fundraiser success. The proposed message-driven architecture of the distributed software system is capable of handling a constant throughput of fundraiser updates with the provided benefit of effortless scalability to support increased loads on the system.

Вступ

Для досягнення успішних результатів у волонтерських проєктах та оптимізації наявних ресурсів, необхідно мати програмні інструменти, які допоможуть передбачити успішність зборів. У даному дослідженні ми розглянемо модифікований алгоритм прогнозування успішності волонтерських зборів та програмне забезпечення (ПЗ), яке базується на цьому алгоритмі. Основною особливістю запропонованого модифікованого алгоритму є використання градієнтного бустингу для аналізу та прогнозування даних, пов'язаних з волонтерськими зборами. Побудовані з використанням зазначеного методу класифікатори здатні виявляти складні залежності та встановлювати закономірності в даних, що дозволяє зробити більш точні прогнози щодо успішності волонтерських проєктів.

Постановка задачі

Задача полягає у вдосконаленні наявного методу машинного навчання моделі, що спирається на градієнтний бустинг, на історичних даних про збори та застосування моделі передбачення успішності нових волонтерських зборів з використанням паралельних обчислень та розподілу навантаження між процесами, що виконують вищезгадані операції, з метою забезпечення швидкодії та масштабованості програмної системи.

Опис методу

У межах даної роботи розглядається передбачення успішності зборів, отриманих з набору даних, що містить 375765 рядків про фандрейзингові

проєкти. Так, доведено, що доцільний набір рис для передбачення успішності зборів у даному наборі даних, є наступним [1]:

- тривалість збору, виражена у кількості днів від розміщення збору до його кінцевої дати;
- предметна область збору, виражена як цілочисельне значення типу перелічення коштів;
- валюта, в якій збираються кошти для збору, виражена як цілочисельне значення типу перелічення;
- країна, де знаходяться автори збору, виражена як цілочисельне значення типу перелічення;
- цільовий обсяг збору, виражений у вищезгаданій валюті;
- поточна кількість меценатів, що надали кошти для проєкту.

Готовий до застосування у реальному світі метод передбачення, що розроблюється, передбачає окремі кроки навчання мережі, верифікації отриманих результатів та додаткового навчання, спираючись на дані, отримані в ході використання програмного засобу [2,3]. Водночас, модифікація методу передбачення успішності волонтерських зборів полягає в розподілі обчислень з періодичного навчання моделі та отримання прогнозів для заданих зборів, спираючись на застосування черги повідомлень, що асинхронно обробляються множиною екземплярів програми обробника [4,5]. Модифікований алгоритм навчання моделі передбачення успішності має такий вигляд:

1. Створення нового періодичного повідомлення з додаткового навчання нейронної мережі за розкладом.
2. Статична ініціалізація нового оточення виконання коду навчання моделі, або ж повторного використання вже наявного, створеного для попереднього навчання, за умови, що інтервал між виконанням навчання є коротшим, ніж життєвий цикл згаданого оточення.
3. Завантаження даних в оперативну пам’ять та їх нормалізація.
4. Випадковий розподіл набору даних на навчальний та тестовий.
5. Виконання навчання моделі до виконання умови зупинки.
6. Верифікація моделі за допомогою надання довільної вибірки тестових даних як вхідних даних для навченої моделі.
7. Серіалізація навченої моделі та розміщення її в спільній для компонентів навчання та передбачення ділянці пам’яті.
8. Розміщення у розкладі наступного виконання процесу додаткового навчання.
9. Виконання кроку 1 за досягнення вищезгаданої точки у часі.

Маючи навчену для заданого періоду часу модель, слід застосувати наведену далі послідовність дій для прогнозування успіху окремого волонтерського збору:

1. Отримання запиту на створення збору.
2. Формування та відправлення повідомлення про прогнозування для даного збору в чергу повідомлень, з дотриманням послідовності

повідомлень для кожного окремого збору за допомогою виділення йому логічно ізольованого контейнера.
3. Отримання повідомлення обробником в окремому потоці.
4. Статична ініціалізація нового оточення виконання коду прогнозування, або ж повторне використання вже наявного, створеного для обробки попереднього збору, що, зокрема, вже має завантажену модель.
5. Завантаження актуальної навченої моделі, в разі створення нового оточення.
6. Трансформація даних про збір у кортеж вхідних значень формату, для якого було виконано навчання.
7. Надання даних збору на вхід моделі та отримання передбачення.
8. Запис отриманого прогнозу в базу даних задля подальшого відображення клієнту.

Задля масштабованості програмної системи, розгортання коду навчання та передбачення було виконано за допомогою хмарного сервісу «AWS Lambda», що дозволяє не лише оброблювати кількість зборів, що зростає в реальному часі, за допомогою ініціалізації нових екземплярів лямбда-функцій, а й забезпечує інтеграцію з брокером повідомлень з задовільними показниками швидкодії та можливістю повторного використання наявних оточень виконання коду задля скорочення затримки між відправленням повідомлення та його обробкою [6].

Розподілення відповідальностей програмних компонентів програмної системи було виконано за допомогою черги повідомлень, наданої сервісом «Amazon SQS», що передбачає можливість інтеграції компонентів на рівні інфраструктури з великою пропускною спроможністю для отриманих повідомлень. Водночас, задля забезпечення подальшого масштабування програмної системи та забезпечення можливості виконання передбачень за допомогою різних обробників повідомлень, що базуються на різні моделі, залежно від вимог до тестування системи або ж оновлення комерційно застосованої системи, інфраструктура передбачає також застосування сервісу «Amazon SNS» з метою паралельного відправлення окремого повідомлення всім можливим обробникам. Запропонований метод було втілено у програмній системі, архітектуру якої наведено на рис. 1.

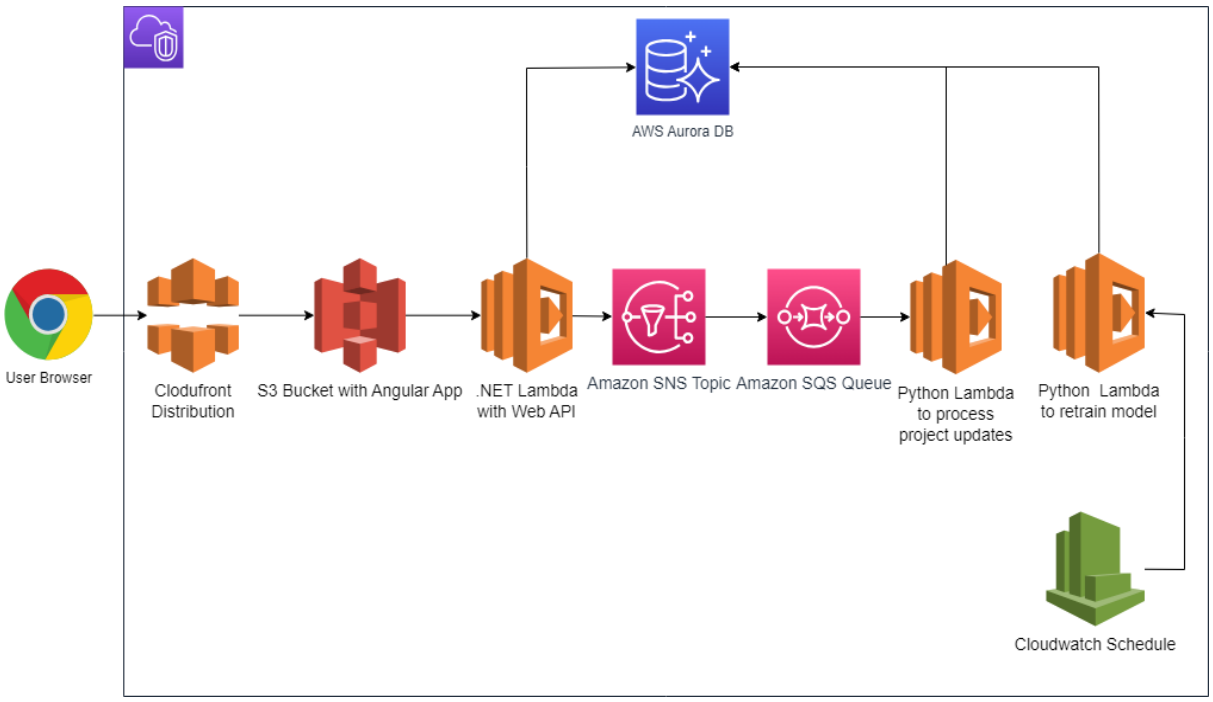


Рис. 1. Запропонована архітектура для програмної реалізації метода

Результати порівняння швидкодії системи, що спирається на послідовну обробку оновлень зборів, та системи, що використовує вищезгадану архітектуру, наведені у таблиці 1.

Табл. 1. Час обробки даних із застосуванням класичної та запропонованої архітектури

Використане програмне забезпечення	ПЗ, що базується на архітектурі з послідовною обробкою	ПЗ, що базується на запропонованій архітектурі з асинхронною обробкою повідомлень
Час обробки 10 зборів, с	29	30
Час обробки 50 зборів, с	81	36
Час обробки 100 зборів, с	156	49
Час обробки 500 зборів, с	813	149

Висновки

Аналізуючи результати прогнозування з використанням запропонованого модифікованого методу, маємо суттєво збільшену швидкодію системи, та, як наслідок, збільшену пропускну здатність програмного компоненту з обробки запитів клієнта, обчислювальні ресурси якого є окремими від асинхронного обробника повідомлень.

Час обробки набору повідомлень, кількість яких не перевищує 10, є аналогічним для запропонованої та класичної архітектури програмного забезпечення. Водночас, ПЗ із запропонованою архітектурою забезпечує зменшення часу обробки наборів повідомлень більшої кількості, у порівнянні з архітектурою, що передбачає послідовну обробку, на 55-81%.

Надалі доцільним є підлаштування створеного програмного засобу під набір даних вітчизняних систем з організації саме волонтерських зборів, а не фандрейзингових кампаній, дані яких через відсутність публічно доступних даних, було використано в межах даної роботи.

Література

1. S. Khosla, J. Reinecke, B. Wittenbrink. (2021). Crowdfunding: Predicting Kickstarter Project Succes, pp. 1-2.
2. W. Wang, H. Zheng, Y. J. Wu. (2020). Prediction of fundraising outcomes for crowdfunding projects based on deep learning: a multimodel comparative study. Soft Computing - A Fusion of Foundations, Methodologies and Applications Volume 24, Issue 11, pp. 8323 – 8341.
3. S. Kao. (2021). A crowdfunding prediction model: a data-driven approach. MISNC '21: Proceedings of the 8th Multidisciplinary International Social Networks Conference, pp. 63 – 70.
4. S. Raje. (2019). Performance Comparison of Message Queue Methods, pp. 33 – 39.
5. Chaney D. (2019). A principal–agent perspective on consumer co-production: crowdfunding and the redefinition of consumer power. Technol Forecast Soc Chang, 141:74–84.
6. J. Dantas, H. Khazaei, M. Litoiu. (2022). Application Deployment Strategies for Reducing the Cold Start Delay of AWS Lambda. IEEE 15th International Conference on Cloud Computing, pp. 4 – 5.

Додаток 4
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Метод та програмне забезпечення для передбачення успішності волонтерських зборів

Студент: Вадим ЛАВРІНЕНКО

Науковий керівник: доцент каф. ПЗКС, к.т.н., доцент Олещенко Л.М.

Київ – 2024

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- Розширення галузі волонтерства зумовлює потребу в можливості аналізу ризику недофінансування зборів.
- Велика кількість зборів, що з'являються повсякчас, потребують методу обробки, що забезпечує велику пропускну здатність та масштабованість програмного забезпечення.

Об'єкт дослідження: процес передбачення успішності волонтерських зборів.

Предмет дослідження: програмні методи та алгоритми розподілених обчислень для класифікації та передбачення успішності волонтерських зборів.

2

Наукове завдання: розробка програмного методу передбачення успішності волонтерських зборів.

Мета дослідження: зменшити час обробки даних зборів за допомогою впровадження архітектури із асинхронною обробкою повідомлень.

Окремі завдання

1. Дослідити наявні методи передбачення успішності волонтерських зборів.
2. Розробити програмний метод, що дозволить передбачати успішність волонтерських зборів.
3. Виконати програмну реалізацію методу.
4. Порівняти отримані результати з результатами аналогічних методів та зробити висновки.
5. Розробити бізнес-модель програмного продукту.

3

НАЯВНІ РІШЕННЯ НАУКОВОЇ ПРОБЛЕМИ

1. Прогнозування часових рядів.
2. Логістична регресія.
3. Застосування дерев ухвалення рішень.
4. Градієнтний бустинг.
5. Колаборативна фільтрація для передбачення успішності збору на основі історії вкладів користувача.

4

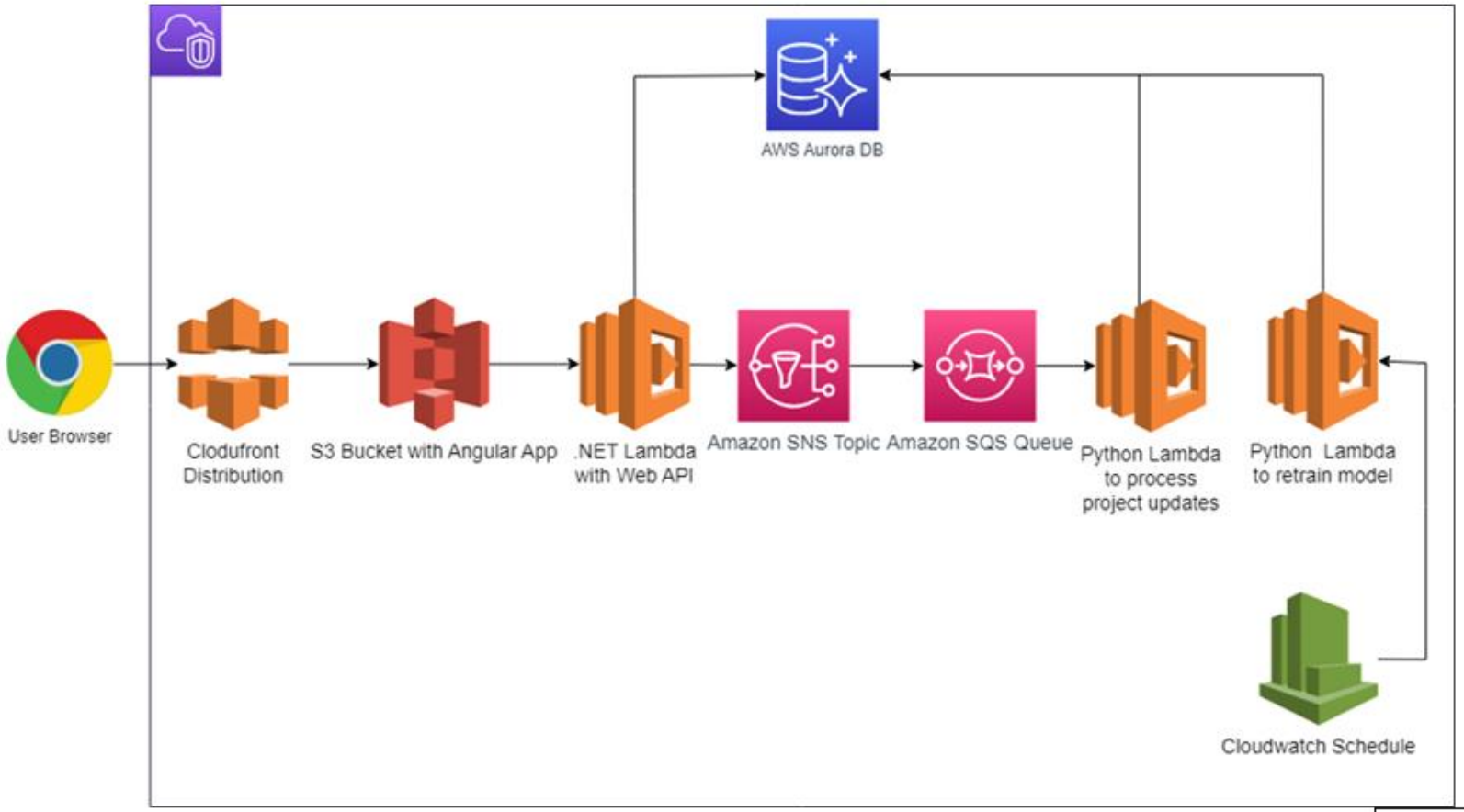
ГІПОТЕЗА

Розробка методу прогнозування успішності зборів у вигляді розподіленої системи збільшить пропускну здатність системи та зменшить час обробки окремих зборів, за умови одночасного отримання більше 30 зборів на обробку, порівняно з архітектурою, що передбачає послідовну обробку.

ТЕХНОЛОГІЇ

PostgreSQL

АРХІТЕКТУРА СИСТЕМИ



7

ДАНІ

В межах роботи розглядається передбачення успішності зборів, отриманих з набору даних про фандрейзингові проекти, що містить 375765 рядків та 15 колонок.

Detail Compact Column 10 of 15 columns

ID	name	category	main_category	currency	deadline	goal	launched	pledged
						Goal amount in project currency		Pledged amount in th project currency
5971 2.15b	375765 unique values	Product Design 6% Documentary 4% Other (340208) 90%	Film & Video 17% Music 14% Other (263158) 69%	USD 78% GBP 9% Other (49164) 13%	2009-05-03 2018-03-03	0.01 100m	1970-01-01 2018-01-02	0
1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP	2015-10-09	1000.00	2015-08-11 12:12:28	0.00
1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD	2017-11-01	30000.00	2017-09-02 04:43:57	2421.00
1000004038	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26	45000.00	2013-01-12 00:20:50	220.00
1000007540	ToshiCapital Rekordz Needs Help to Complete Album	Music	Music	USD	2012-04-16	5000.00	2012-03-17 03:24:11	1.00
1000011046	Community Film Project: The Art of Neighborhood Filmmaking	Film & Video	Film & Video	USD	2015-08-29	19500.00	2015-07-04 08:35:03	1283.00
1000014025	Monarch Espresso Bar	Restaurants	Food	USD	2016-04-01	50000.00	2016-02-26 13:38:27	52375.00

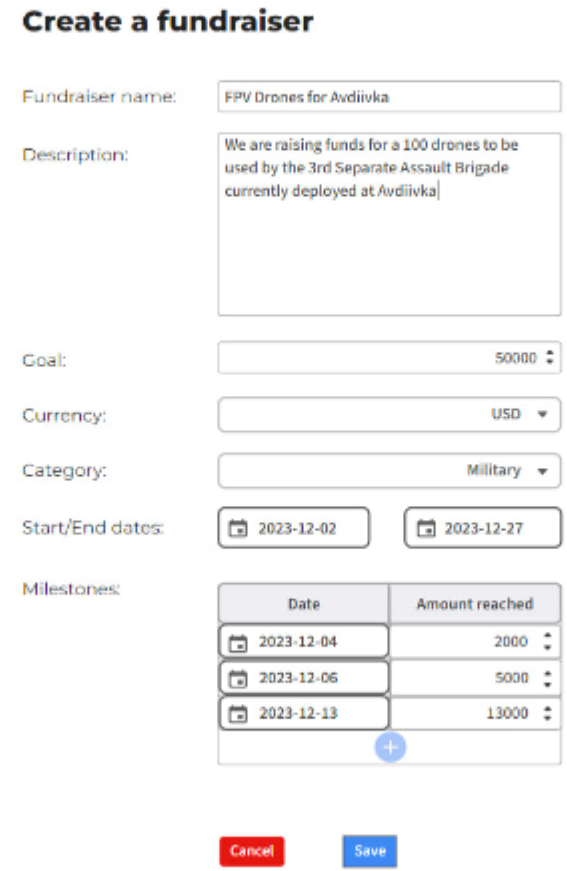
8

НАБІР ОЗНАК

Набір ознак, що було використано для передбачення успішності зборів у даному наборі даних:

- тривалість збору;
- предметна область збору;
- валюта, в якій збираються кошти для збору;
- країна, де знаходяться автори збору;
- цільовий обсяг збору;
- поточна кількість меценатів.

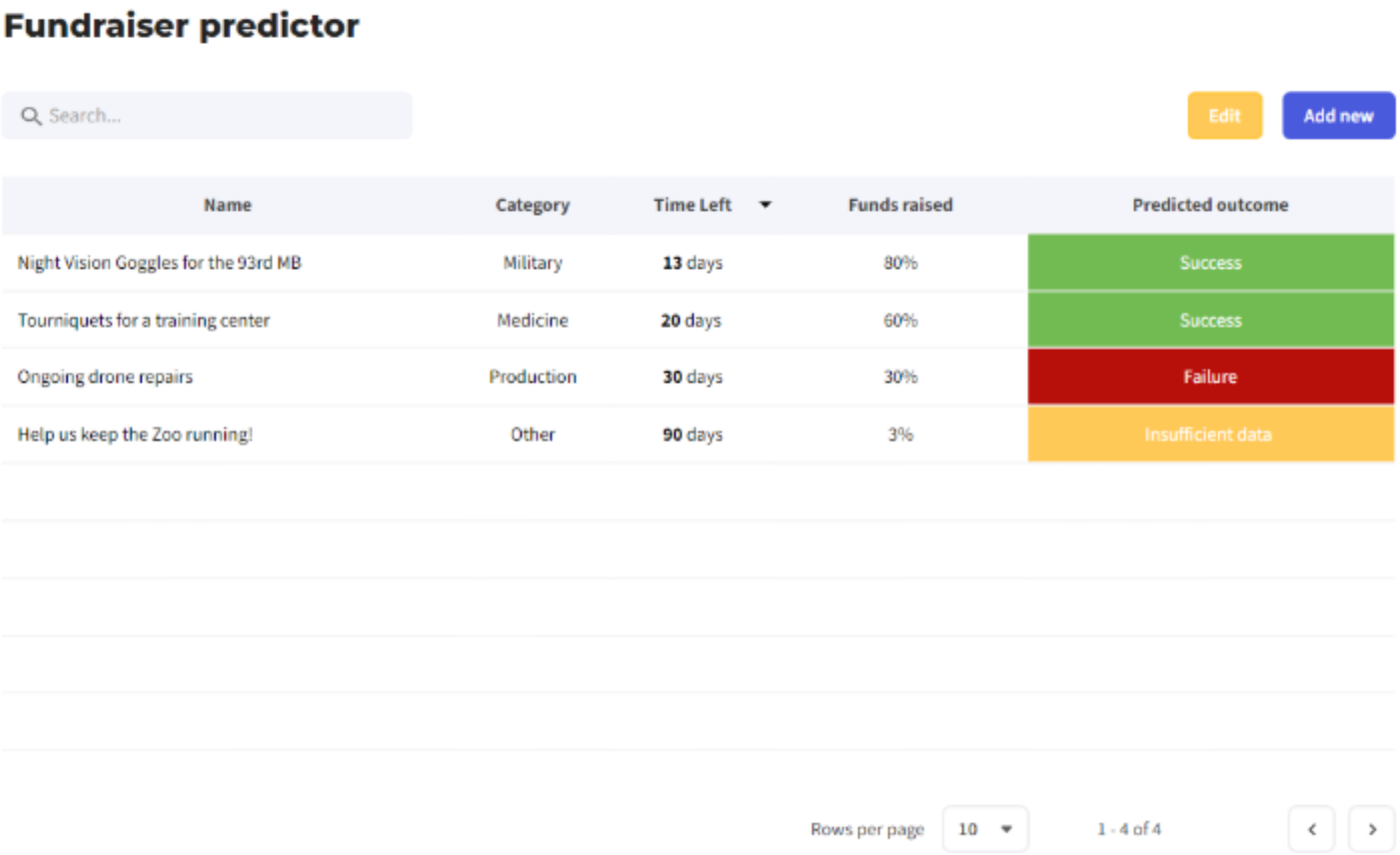
ВЕБЗАСТОСУНОК



Меню додавання нового збору

10

ВЕБЗАСТОСУНОК



Перелік зборів з передбаченнями успішності

11

МЕТОД ПЕРЕДБАЧЕННЯ

Задача передбачення успішності збору належить до задач бінарної класифікації, де заданий збір належить одному з двох класів: успішний або неуспішний.

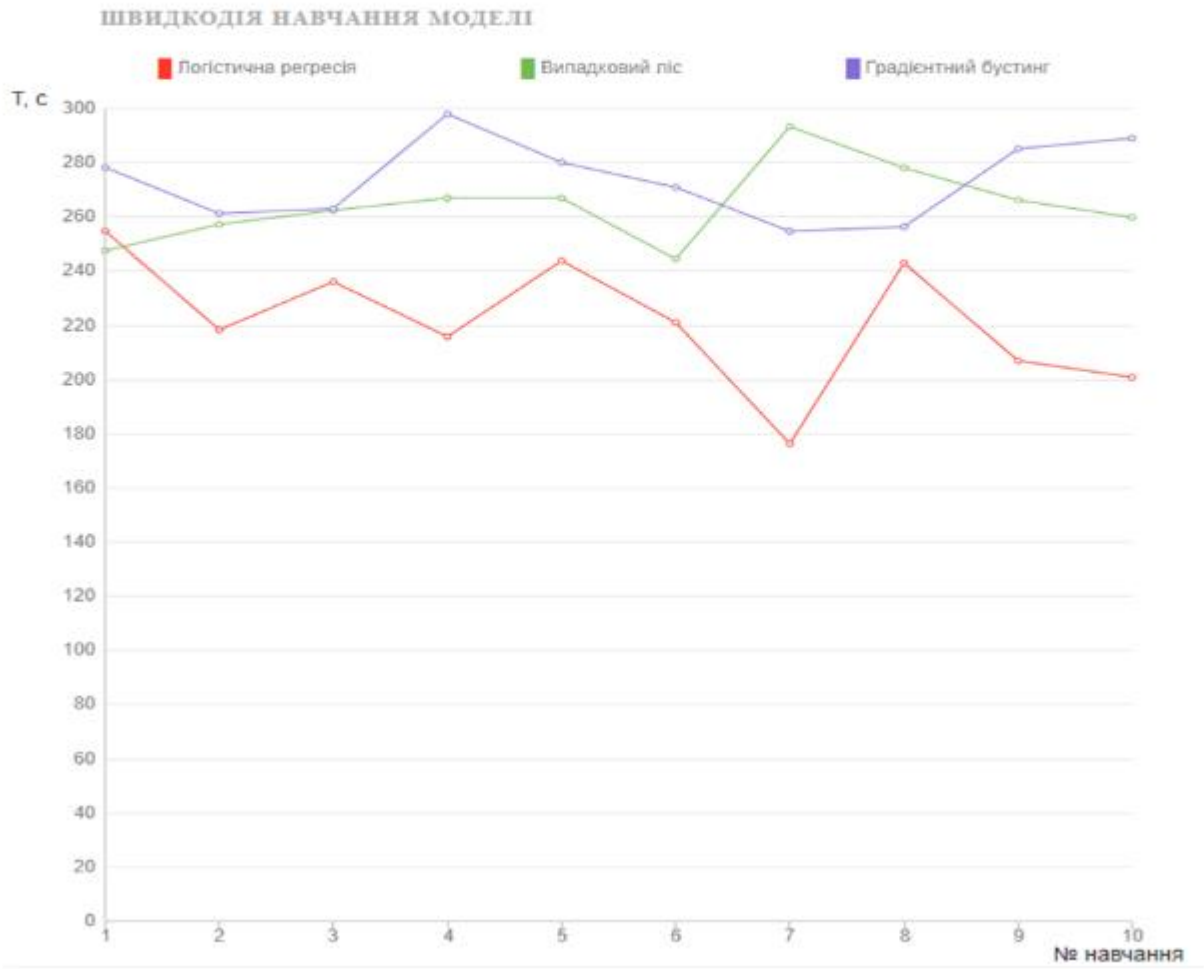
За базовий метод було обраного градієнтний бустинг, навчання моделі за якого складається з кроків:

1. Створення слабкої моделі - дерева рішень.
2. Обчислення помилки поточної моделі.
3. Створення нової слабкої моделі, цільова змінна якої - помилка з попереднього кроку.
4. Обчислення поточного передбаченого значення цільової змінної як суми значень, передбачених минулими 2 кроками.
5. Кроки 2-4 виконуються до досягнення умови зупинки навчання.

ВИБІР БАЗОВОГО МЕТОДУ

Метод	Точність, %
Логістична регресія	76
Випадковий ліс	79
Градієнтний бустинг	83

ВИБІР БАЗОВОГО МЕТОДУ



14

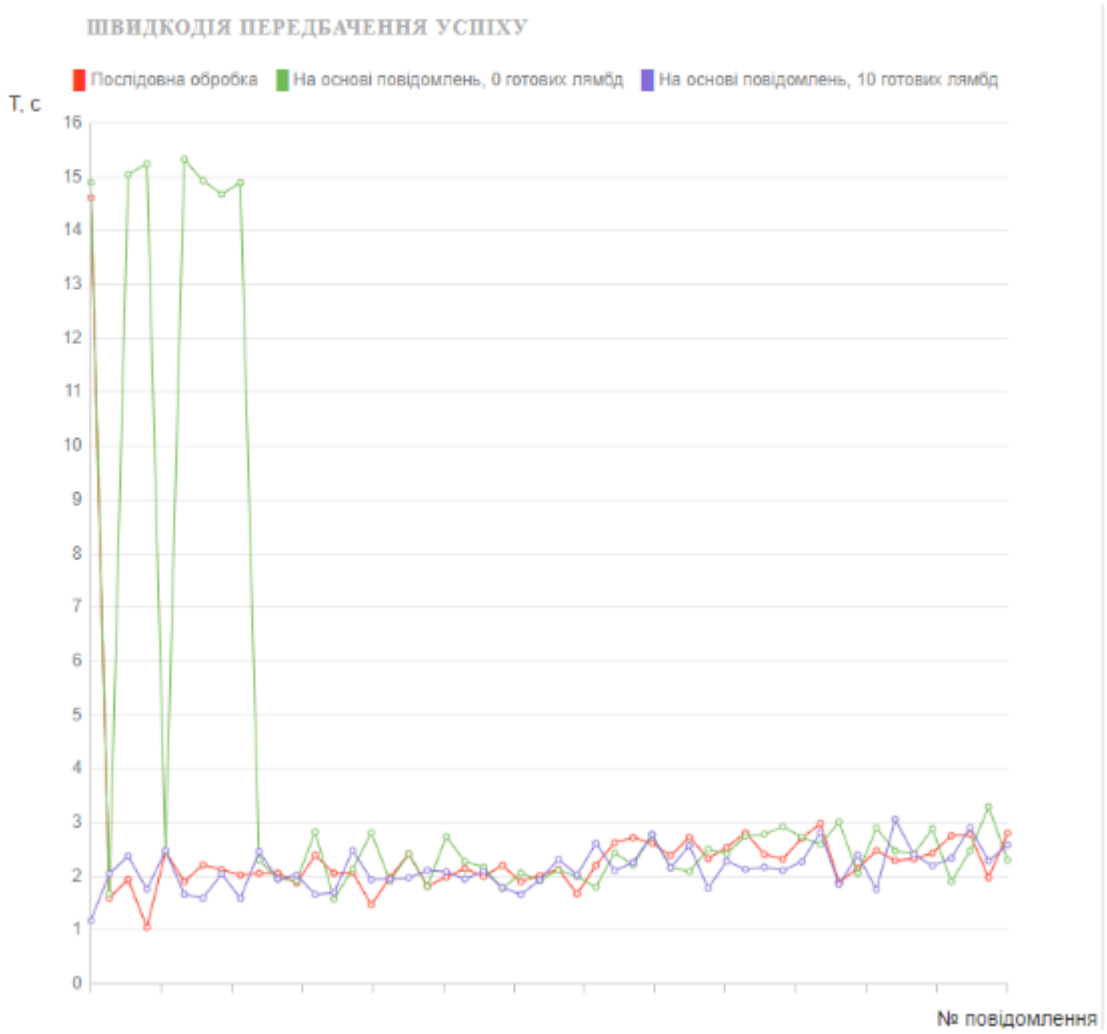
АЛГОРИТМ ПЕРЕДБАЧЕННЯ

15

РЕЗУЛЬТАТИ

Використане програмне забезпечення	ПЗ, що базується на архітектурі з послідовною обробкою	ПЗ, що базується на запропонованій архітектурі з асинхронною обробкою повідомлень
Час обробки 10 зборів, с	29	30
Час обробки 50 зборів, с	81	36
Час обробки 100 зборів, с	156	49
Час обробки 500 зборів, с	813	149

РЕЗУЛЬТАТИ


ШВИДКОДІЯ ПЕРЕДБАЧЕННЯ УСПІХУ
Послідовна обробка
На основі повідомлень, 0 готових лямбд
На основі повідомлень, 10 готових лямбд
T, c
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
№ повідомлення

БІЗНЕС МОДЕЛЬ

Проблема Обмеженість аудиторії, залученої до збору; відсутність контролю успіху збору; складність прогнозування успішності.	**Інноваційна технологія** Прогнозування успішності зборів; сортування зборів за тематикою та репутацією волонтера.	**Унікальна ціннісна пропозиція** Спрощений вхід у царину волонтерства; система репутації як заохочення до добросовісної роботи; простота використання для клієнта.	**Зацікавлені сторони** Волонтери, меценати, благодійні фонди, бенефіціари.	**Ринок** Відсутні аналогічні програмні рішення.
Рішення Створення централізованої системи з прогнозуванням успіху зборів та інтелектуальним пошуком зборів.	**Програмний продукт** Веб-додаток зі складовими для взаємодії меценатів та волонтерів.		**Конкурентні переваги** Оптимізація процесів створення, пошуку, верифікації та прогнозування зборів.	**Канал збуту** B2B: взаємодія з благодійними фондами та меценатами - організаціями. B2C: функціонал для роботи волонтерів та меценатів.
Структура витрат Заробітна плата; оренда приміщень; реклама; операційні витрати.			**Потоки доходів** Комісія з кожної транзакції.	

18

НАУКОВА НОВИЗНА

У даній роботі вперше застосовано модифікований метод, що спирається на архітектуру з використанням асинхронної обробки повідомлень та дозволяє зменшити час обробки даних про збори кількістю понад 30, на 55-81%, порівняно з використанням послідовної обробки.

ПРАКТИЧНА ЦІННІСТЬ

Практична цінність роботи полягає у запропонованій архітектурі розподіленої системи для розміщення оголошень про волонтерські збори та асинхронного передбачення їх успішності, що є конкурентоспроможною та готовою до розгортання у вигляді комерційного продукту.

АПРОБАЦІЯ

Основні положення і результати даної роботи були представлені на науковій конференції магістрантів та аспірантів

“Прикладна математика та комп’ютинг ПМК-2023”, та опубліковано у збірнику тез доповідей: Лавріненко В.В., Олещенко Л.М. Метод та програмне забезпечення для передбачення успішності волонтерських зборів. Прикладна математика та комп’ютинг ПМК-2023” (Київ, 28 - 30 листопада 2023 р.), с. 486-490.

21

ВИСНОВКИ

1. У магістерській дисертації досліджено наявні методи передбачення успішності волонтерських зборів, серед яких у якості базового було обрано градієнтний бустинг.
2. Було вперше застосовано архітектуру з використанням асинхронної обробки повідомлень для задачі передбачення успішності волонтерських зборів. Як наслідок, було спроектовано конкретну архітектуру розроблюваної системи.
3. Було виконано програмну реалізацію спроектованої архітектури.
4. Було виконано порівняння швидкодії передбачень успішності наборів зборів різної кількості програмною системою, що передбачає запропоновану архітектуру, та програмною системою, що передбачає послідовну обробку. При цьому запропонована архітектура забезпечила зменшення часу обробки на 55-81%.
5. Розроблено бізнес-модель запропонованого програмного продукту.

UNICHECK
by Turnitin

User name:
Олещенко Любов Михайлівна

Check ID:
1016051551

Check date:
09.01.2024 19:40:06 EET

Check type:
Doc vs Internet + Library

Report date:
09.01.2024 22:34:00 EET

User ID:
84299

File name: **Лавріненко_Вадим_КП_21мп**

Page count: **58** Word count: **12120** Character count: **94627** File size: **83.87 KB** File ID: **1015752150**

1.43% Matches

Highest match: **0.22%** with Library source (File ID: **1005741299**)

0.51% Internet sources	84	Page 60
1.36% Library sources	175	Page 60

0% Quotes

No quotes found

Exclusion of references is off

0% Exclusions

23

Дякую за увагу!