Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Малишка Андрія Івановича*
(ПІБ)

академічної групи *121-21ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для вибору інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І.С.*			
розділів:				
спеціальний	*проф. Лактіонов І.С.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Мартиненко А.А.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-21ск-1* *Малишка А.І.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для вибору інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation*

затверджена наказом ректора НТУ «ДП» від *23.04.2024* № *375-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***01.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***06.06.2024 р.***

Завдання видав _____ *проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Малишко А.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 101 с., 36 рис., 3 табл., 3 дод., 29 джерел.

Об'єкт розробки: програмне забезпечення для вибору інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation.

Мета кваліфікаційної роботи: створення програмного забезпечення для вибору інформативних ознак від результатів агромоніторингу на основі метаевристичного алгоритму Whale Optimisation, що може більш точно визначити необхідні стовпці ознак, оптимізація роботи агрокомплексу, використовуючи техніки машинного навчання, видача інформативного результату для подальшого аналізу та прогнозування стану агрокультур.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного забезпечення, що надає можливість використовувати як готовий продукт з графічним інтерфейсом, так і як API за допомогою якого можна покращити власний програмний продукт, зменшивши кількість інформативних ознак та заощадити на обчислювальних потужностях власних технічних засобів.

Актуальність даного програмного продукту визначається постійним попитом на програмні продукти, що могли покращити роботу агромоніторингу, то розробка, оскільки потрібно постійно вдосконалювати свою роботу, щоб бути більш конкурентноспроможним.

Список ключових слів: АГРОМОНІТОРИНГ, ВИБІР ІНФОРМАТИВНИХ ОЗНАК, API, ВЕБІНТЕРФЕЙС, СЕРВЕР, КЛІЄНТ, WHALE OPTIMISATION ALGORITHM, СЕРВІС, МАШИННЕ НАВЧАННЯ.

ABSTRACT

The explanatory note refers to page 101, figure 36, table 3, and a bibliography of 29 referenced sources.

The object of developing is software creation for informative feature selection from results of agricultural monitoring based on Whale Optimisation metaheuristic algorithm.

The objective of qualifying work is software creation for informative feature selection from results of agricultural monitoring based on Whale Optimisation algorithm that can identify needed features columns, optimise agricultural complex work using machine learning techniques, give the informative result for future analysis and predicting of agriculture state.

In the introduction, the analysis and current state of the problem is considered, the objective of the qualifying work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first section, the subject area is analysed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, available solutions are analysed, platforms for development are chosen, program design and development is performed, program operation, algorithm and structure of its functioning are described, as well as program calling and loading, input and output data are determined, the composition of technical means parameters is characterized.

In the economic section, the labour intensity of the developed information system is determined, the cost of work on creating the program is calculated, and the time for its creation is calculated.

The practical value is to create software that provides an opportunity to use both a ready-made product with a graphical interface and as an API with the help of which you can improve your own software product, reducing the number of informative features and saving on the computing power of your own technical means.

The relevance of this software product is determined by the constant demand for software products that could improve the work of agricultural monitoring, then development, because you need to constantly improve your work in order to be more competitive.

The list of key words: AGRICULTURAL MONITORING, INFORMATIVE FEATURE SLECTION, API, WEB INTERFACE, SERVER, CLIENT, WHALE OPTIMISATION ALGORITHM, SERVICE, MACHINE LEARNING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

МН – Машинне навчання

ПЗ – Програмне забезпечення

ПК – Персональний комп'ютер

ШІ – Штучний інтелект

CSS – Cascading Style Sheets

DOM – Document Object Model

HMR – Hot Module Replacement

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

HTTPS – Hyper Text Transfer Protocol Secure

JSON – JavaScript Object Notation

MVT – Model-View-Template

API – Application Programming Interface

CSV – Comma-Separated Values

IoT – Internet of Things

UML – Unified Modeling Language

WOA – Whale Optimisation Algorithm

ЗМІСТ

ВСТУП

На сьогодні існує величезний обсяг даних, особливо якщо це стосується агромоніторингу, які необхідно обробити та виокремити ті, що можуть надати нам відповідь на наступні питання як:

- фактори, що впливають на врожайність;
- зони поля, що потребують додаткового зрошення чи внесення добрив;
- зміни навколишнього середовища, що можуть змінити розвиток культур.

Однак, через те, що таку кількість даних важко обробити вручну, тим паче виділити з них ті ознаки, які впливають на ту чи іншу проблематику, щоб потім можна було спрогнозувати можливі зміни в агрокультурах або виявити можливі залежності в певних ознаках для визначення хвороби, від якої можуть страждати агрокультури.

Щоб зменшити час на визначення певних інформативних ознак, що несуть в собі дані отримані від агромоніторингу більш раціонально використовувати програмне забезпечення, яке змогло б виокремити їх та допомогти користувачеві отримати результат за досить малий проміжок часу та надати певну звітність про результат обробки даних, з чого можна провести певний аналіз та дати прогноз про можливий стан агрокультур на наступний досліджуваний період. Для оптимізації процесу визначення певних інформативних ознак в програмному забезпеченні можуть використовуватись різні алгоритми вибірки ознак.

Алгоритм вибірки ознак – це процес обирання підмножини доречних ознак (змінних) з великого набору даних для використання в побудові моделі. Метою такого вибору є спрощення моделей, зниження розмірності, уникнення перенавчання та покращення узагальнення [1].

З поміж багатьох алгоритмів, окремо необхідно виділити про Whale Optimisation (WOA), оскільки саме цей алгоритм використовується про розробці програмного забезпечення, за допомогою якого буде виконуватись вибірка інформативних ознак з результатів агромоніторингу. Цей алгоритм, натхнений поведінкою горбатих китів під час полювання, що надає йому певну особливість

та перевагу при швидкого пошуку ознак при величезних обсягах даних. WOA відзначається швидкою збіжністю та меншою кількістю обчислень, що робить його привабливим для великих наборів даних, а також, за потреби, може бути гібридизований з іншими алгоритмами, що дозволяє поєднувати його переваги з іншими методами вибірки ознак.

Мета кваліфікаційно роботи полягає у розробці програмного забезпечення, яке здатне ефективно вибирати інформативні ознаки з даних, отриманих в результаті агромоніторингу, з використанням алгоритму Whale Optimisation. Це програмне забезпечення має на меті спростити процес аналізу даних для користувачів, які працюють в аграрній галузі, забезпечуючи їм інструмент для швидкого та точного визначення ключових параметрів, які впливають на врожайність та стан культур.

Задачі, які стоять перед програмним забезпеченням:

- створення інтерфейсу, який прийматиме дані від користувача про стан агрокультур та за допомогою WOA здійснить вибірку необхідних ознак для подальшого прогнозування майбутнього продукту або виявлення можливих хвороб;

- видача результату про вибрані ознаки;

- полегшити роботу агромоніторингу для фахівців, що працюють в цілому в аграрній галузі та займаються аналітикою даних відповідно у даній галузі.

В Україні існують ряд компаній, що розроблюють програмне забезпечення для аграрної галузі серед них Kernel та ELEKS, створюють програмні засоби для сільського господарства, які допомагають автоматизувати бізнес-процеси, включаючи сезонне планування, моніторинг, польові роботи та продаж [2].

Оскільки в Україні досить розвинена аграрна галузь, то розробка програмного забезпечення, яке здатне оптимізувати процес обробки агрокультур, має місце та є актуальним.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Агромоніторинг має ключове значення у сучасному аграрному секторі, який дозволяє оптимізувати виробництво та підвищити його ефективність.

Сучасний стан проблеми агромоніторингу в Україні характеризується рядом викликів та можливостей. З одного боку, аграрний сектор є одним з ключових напрямків економіки країни, що вимагає постійного вдосконалення методів моніторингу та аналізу даних для підвищення ефективності виробництва. З іншого боку, існують проблеми, які стримують розвиток цієї галузі [3].

Спостерігається тенденція наростаючого технічного та технологічного відставання вітчизняних агропромислових підприємств від рівня виробництва конкурентів з розвинутих країн. Це стосується як обладнання для збору та аналізу даних, так і програмного забезпечення для їх обробки [4].

Погіршення екологічного стану природних ресурсів, зокрема ґрунтів, є серйозною проблемою. Це впливає на якість сільськогосподарської продукції та вимагає впровадження ефективних систем агромоніторингу для своєчасного виявлення та реагування на негативні зміни [4].

Ступінь розв’язання завдань у сфері агромоніторингу в Україні має певні успіхи, що поліпшили збір даних про агрокультури:

- використання Інтернет речей (IoT) для збору даних;
- використання штучного інтелекту (ШІ) для аналізу даних та прогнозів;
- точне землеробство та роботехніка;
- агродрони для моніторингу та обробки.

Нижче наведені наступні ключі успіхи, які досягнуто на 2024 рік в Україні.

Автоматизований збір даних став невід’ємною частиною сільськогосподарського процесу, де IoT-інструменти збирають і передають дані

в режимі реального часу, що дозволяє фермерам відстежувати важливі показники стану ґрунту, рівень опадів, стан сільськогосподарського обладнання та якісні характеристики врожаю [5].

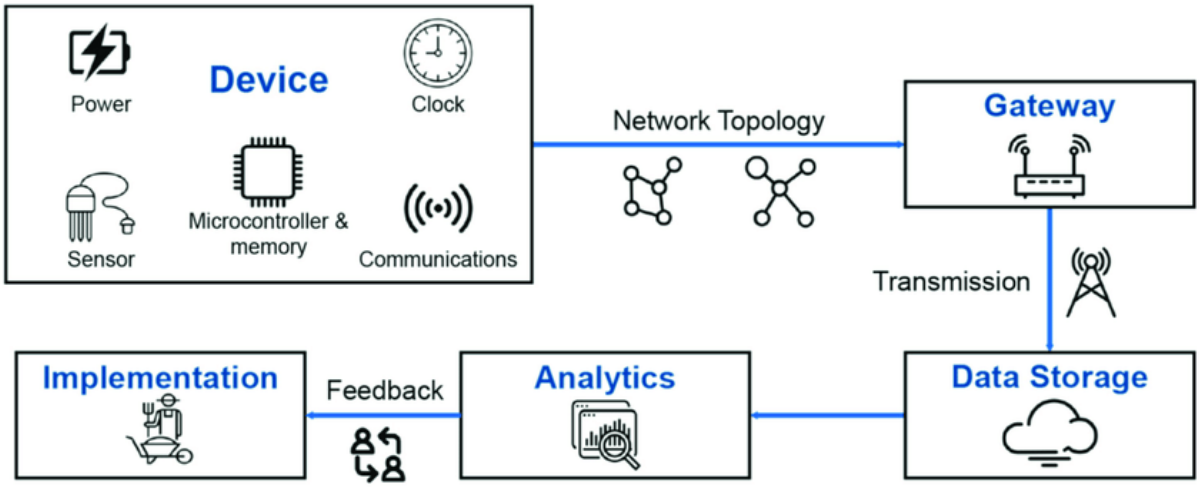


Рис. 1.1. Загальна схема роботи Інтернет речей в агросекторі

ШІ використовується для обробки даних, отриманих з IoT-пристроїв, реагує на зміни в показниках та за допомогою машинного навчання дає рекомендації для покращення сільськогосподарських процесів [5].

Рис. 1.2. Основні завдання штучного інтелекту в агросекторі

Впровадження технологій точного землеробства та робототехніки дозволяє підвищити точність роботи та відстеження даних, що сприяє економії ресурсів, покращенню стану ґрунту та збільшенню врожайності [6].

Рис.1.3. Поливальний робот

Використання дронів для моніторингу полів та внесення добрив стає все більш популярним, оскільки це дозволяє здійснювати точніше внесення ресурсів та швидше реагувати на зміни в стані рослин [6].

Рис. 1.4. Дрон, що вносить добрива на агрокультури

Ці технології використовують українські аграрії для поліпшення продуктивності та ефективності ведення сільського господарства, забезпечуючи досить стійке та екологічно чисте виробництво, яке в свою чергу може зробити українську агропродукцію більш конкурентною на світовому ринку та привабливою для імпортерів.

Проблеми і виклики, які існують незважаючи на технологічний прогрес, пов'язані з інтеграцією різних джерел даних, необхідністю вдосконалення методів обробки та аналізу даних, а також потребою в підвищенні точності прогнозів.

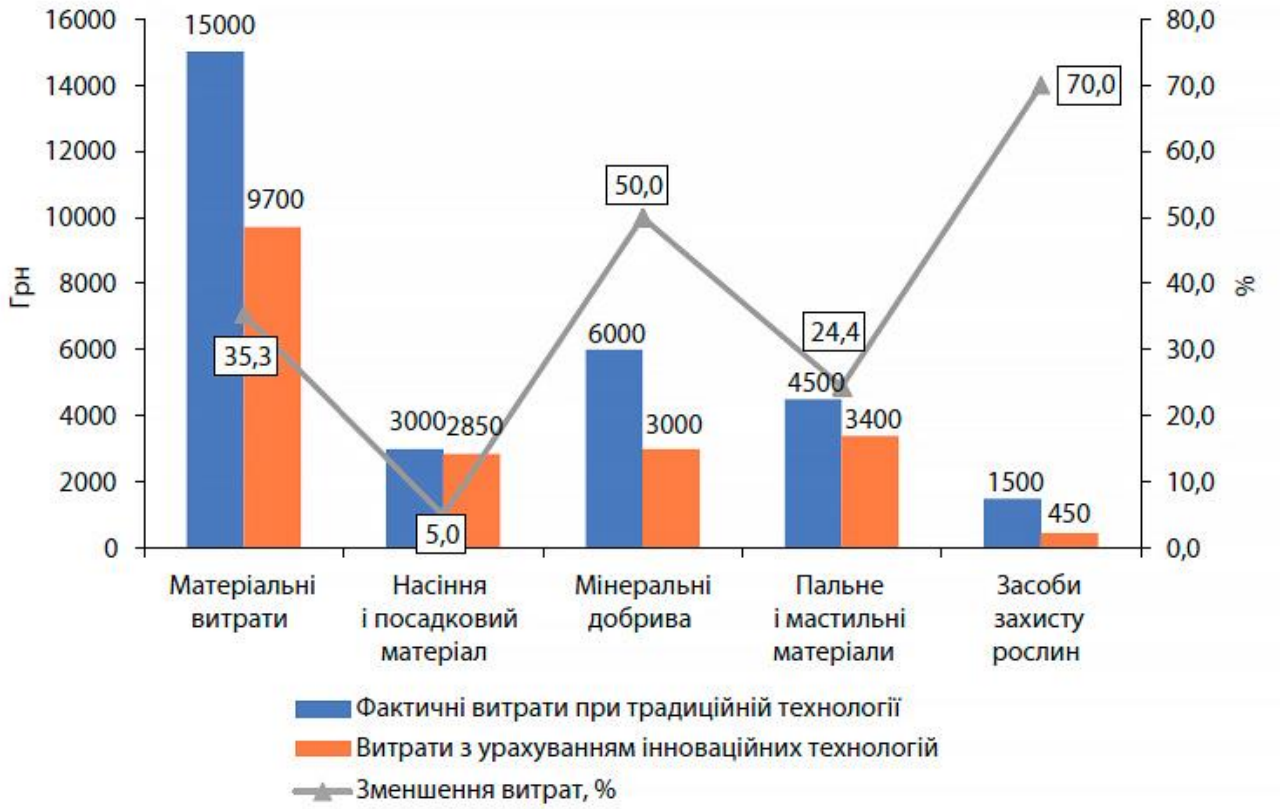


Рис. 1.5. Ефективність запровадження інноваційних технологій для вирощування озимої пшениці на прикладі господарств Вінницької області у 2021 році

В перспективі ці проблеми можуть бути вирішенні впровадженням більш досконалих методів та алгоритмів машинного та/або глибинного навчання в

обробку даних агромоніторингу для підвищення ефективності аналізу та прийняття рішень.

В Україні та у світі існує багато компаній, які розробляють програмні продукти для агромоніторингу та використовують технології машинного навчання та глибинного навчання.

ELEKS – компанія, яка пропонує розробку програмного забезпечення для агросектору, включаючи інтеграційні та автоматизаційні рішення, а також великі дані та розширений аналіз.

Представництва компанії є у 13 країнах світу, включаючи США, Великобританію, Польщу, Німеччину, Канаду, Швейцарію, Японію, ОАЕ та Саудівську Аравію [7].

Послуги, які вони надають аграріям:

- інтеграційні та автоматизаційні рішення;
- великі дані та розширений аналіз.

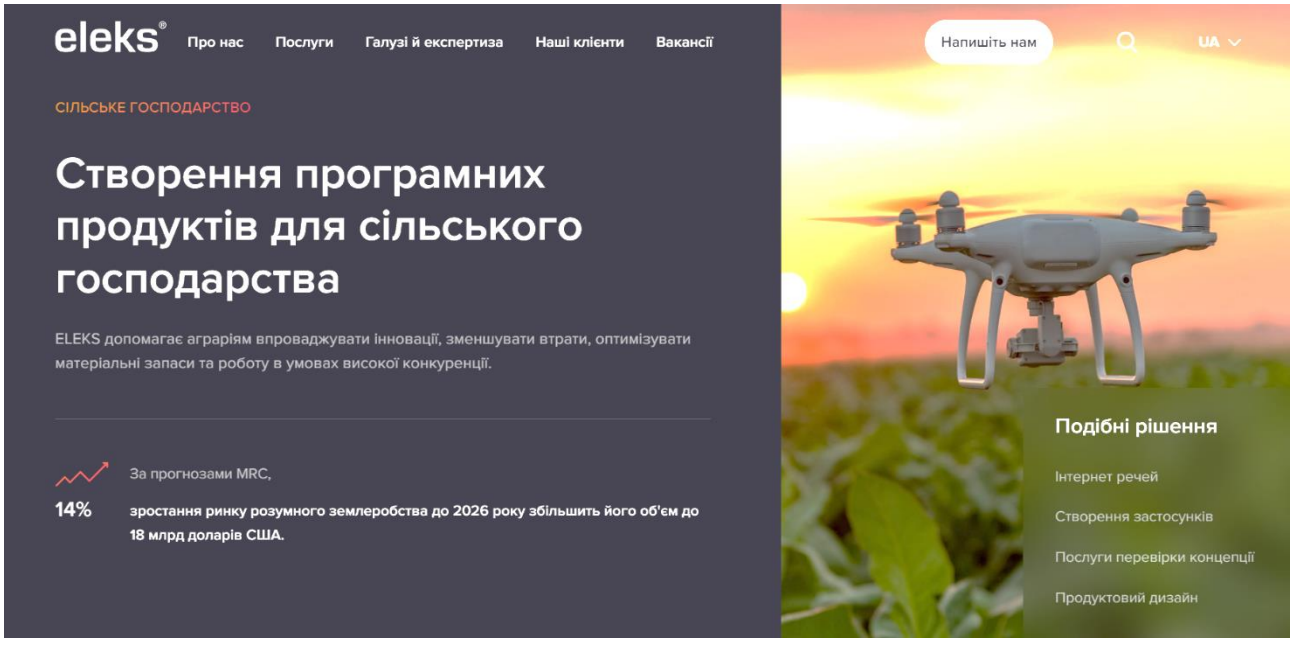


Рис. 1.6. Скриншот сторінки Створення програмних продуктів для сільського господарства | ELEKS

ELEKS допомагає з'єднувати та автоматизувати IT-інфраструктуру аграрного підприємства, щоб виявити прогалини в процесах та комунікації, які

можуть уповільнювати виробництво. Вони пропонують рішення для зниження витрат ресурсів, забезпечуючи безперервний обмін даними між всіма вашими системами.

ELEKS використовує експертизу з моделювання даних та розробки машинного навчання для оптимізації аналітики даних (діагностичної, описової та прогнозної) та обчислення потоку навантаження.

Основні технології, які застосовуються компанією:

- Data Science і предиктивне моделювання;
- аналітика та візуалізація;
- платформи й мікросервіси;
- Інтернет речі;
- мобільні технології.

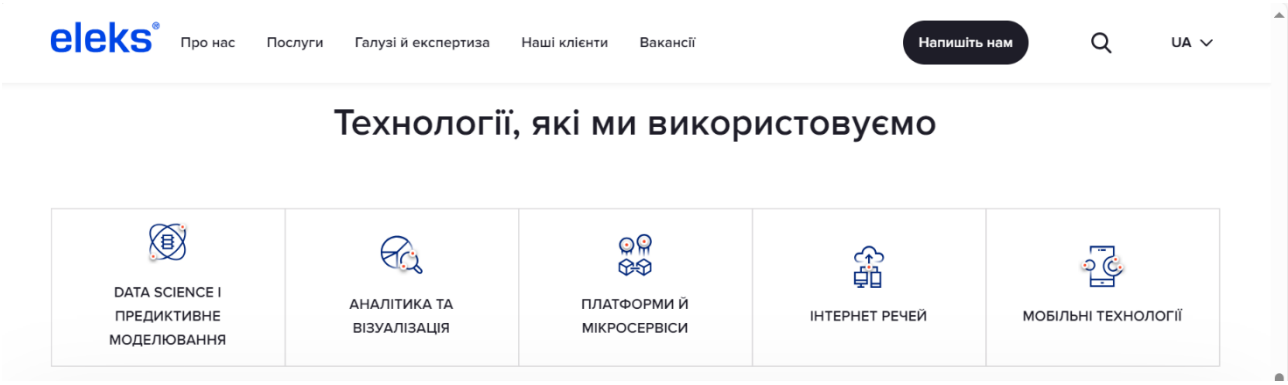


Рис. 1.7. Скриншот секції сторінки про використані технології

Програмне забезпечення, яке компанія створює для аграрних підприємств на замовлення може забезпечити наступне:

- оптимізувати виробництво та збільшити прибуток;
- планувати виробництво на основі накопичених даних;
- об’єднати технологічні процеси за допомогою інтернету речей;
- збалансувати запаси матеріальних засобів і заощадити кошти;
- покращити взаємодію за допомогою мобільних технологій.

Наступна компанія, яка надає свої послуги у розробці програмного забезпечення аграріям, – Kernel Digital.

Kernel Digital – це IT-компанія, яка спеціалізується на використанні штучного інтелекту для оптимізації роботи агрономів. Вони розробляють цифрові рішення, які дозволяють агрономам використовувати смартфони та інші мобільні пристрої для підвищення продуктивності на полі.

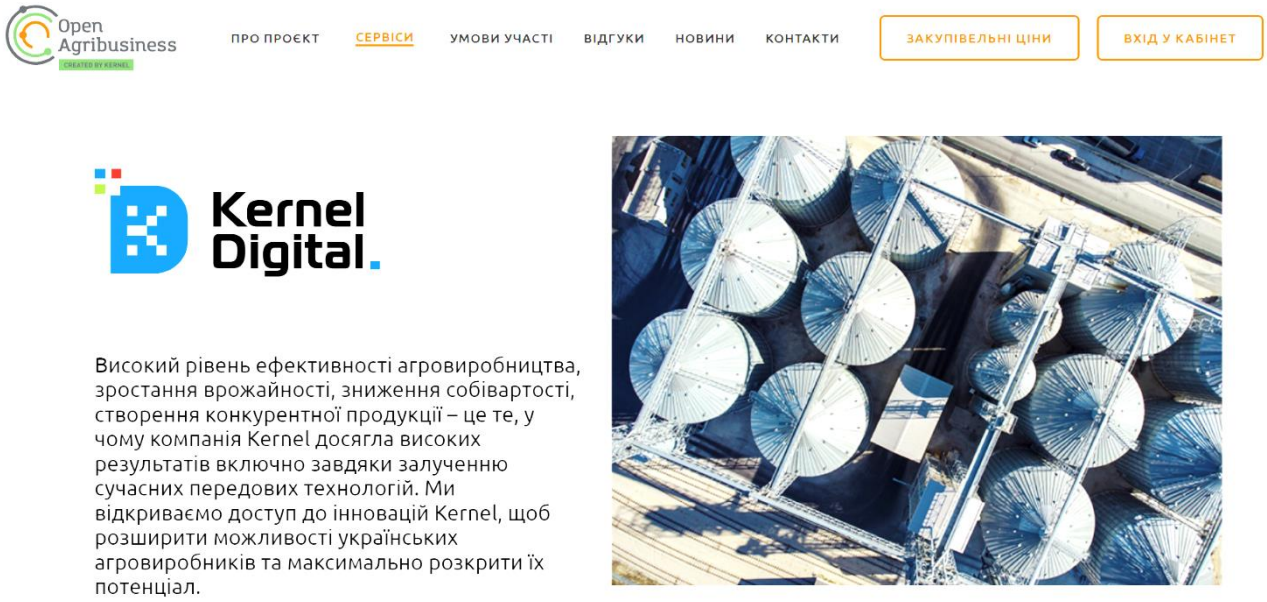


Рис. 1.8. Скриншот сторінки Open Agribusiness

Інструменти та застосунки, які вони розроблюють для агросектору:

- інтелектуальне планування;
- моніторинг у реальному часі;
- паспорт поля;
- калькулятор добрив;
- розширений лабораторний аналіз.

Агрономи отримують рекомендації щодо посіву на основі історичних даних, що дозволяє оптимізувати розподіл ресурсів та підвищити ефективність посіву, збирання врожаю та внесення добрив [8].

Звіти з поля, включаючи відео та аналіз ґрунту, надають інформацію для прийняття обґрунтованих рішень, надаючи актуальні дані про стан полів [8].

Надає огляд важливих показників для кожного поля, надаючи зацікавленим сторонам важливу інформацію [8].

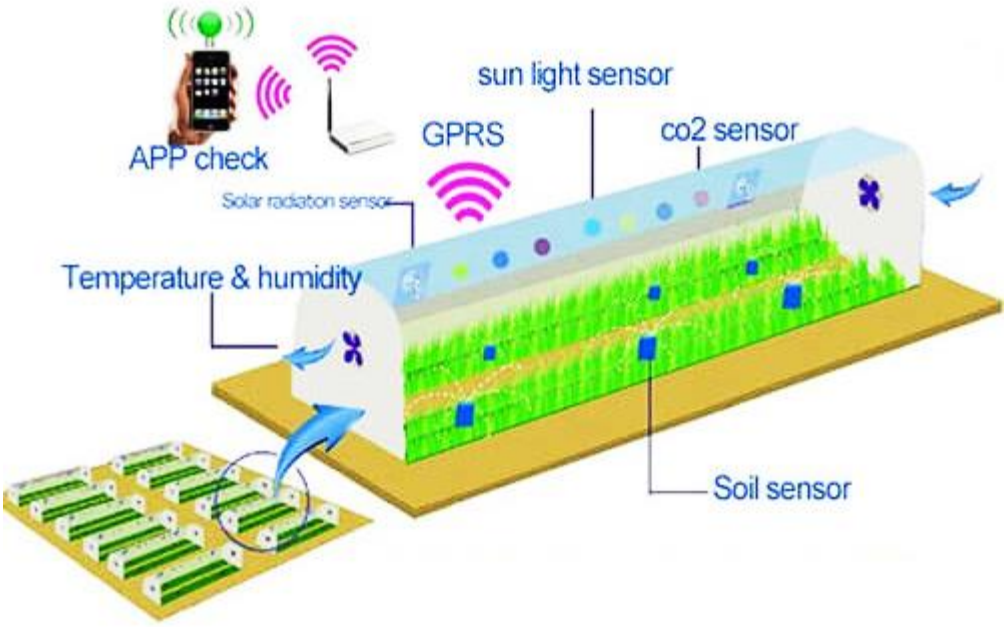


Рис. 1.9. Взаємодія датчиків під час моніторингу даних у теплиці

Розрахунок потреби в активних речовинах для поля на основі численних показників ґрунту (агрохімії), попередньої культури та іншого [8].

Детальний аналіз складу ґрунту та інших показників сприяє обґрунтованій аграрній практиці.

Компанія Kernel Digital активно використовує Data Science для обробки даних агромоніторингу. Це дозволяє їм аналізувати великі обсяги даних, отриманих з різних джерел, таких як сенсори на полях, супутникові знімки, погодні станції та інші.

Завдяки методам Data Science, Kernel Digital може:

- прогнозувати врожайність;
- виявляти захворювання рослин;
- оптимізувати використання ресурсів.

Використовуючи історичні дані та машинне навчання, компанія може прогнозувати потенційну урожайність, що допомагає агрономам планувати та оптимізувати свої ресурси.

Алгоритми Data Science дозволяють швидко ідентифікувати ознаки хвороб на рослинах, що сприяє своєчасному лікуванню та зменшенню втрат.

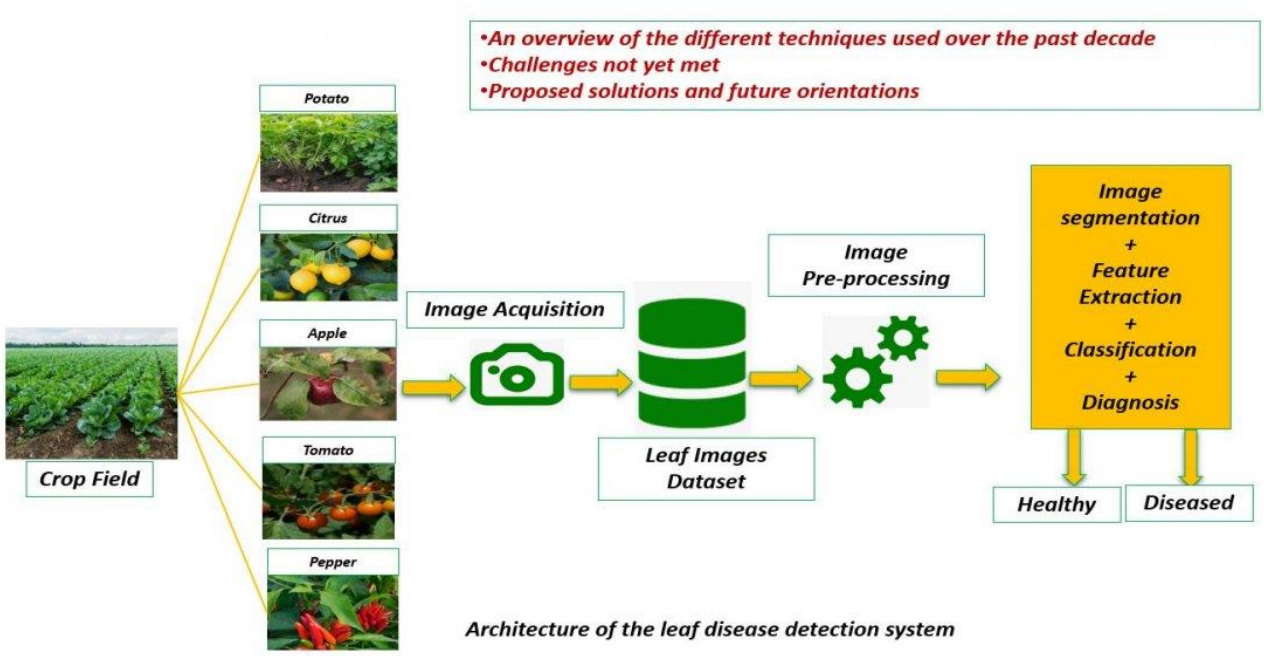


Рис. 1.10. Алгоритм дій для визначення захворювання у рослини, використовуючи її фотографію

Аналізуючи дані про стан ґрунту, вологість, температуру та інші фактори, компанія може рекомендувати найбільш ефективні методи зрошення та внесення добрив.

Ці технології допомагають агрономам не тільки підвищити ефективність роботи на полі, але й сприяють сталому розвитку сільського господарства, зменшуючи вплив на довкілля [9].

Можна підбити підсумки щодо розвитку аграрного сектору впровадженням різних технологій, включно зі штучним інтелектом. В Україні цифровізація сільського господарства продовжується та оптимізує роботу аграріїв, що призводить до збільшення обсягів та якості продукції. Також існують компанії, що можуть надавати послуги в розробці програмного забезпечення, яке виконує цілий спектр задач. Проте це програмне забезпечення створюється конкретно для окремої компанії, тобто прикладної програми, яка змогла бути в доступі для кожного підприємця без прив’язки до певної компанії, не виявлено в ході пошуку, тоді виходить, що кожна компанія, яка працює в аграрному секторі та потребує доступ до оптимізації даних, мусить купувати окрему програму в компанії-розробника. З одного боку, будь-який підприємець,

що має можливість придбати готовий програмний продукт може отримати той, що буде задовольняти всі його вимоги, але з іншого боку, підприємства, які не можуть придбати окремий продукт, – не можуть покращити свій виробничий процес, і більш ефективно для них – мати доступ до вже готового програмного забезпечення, яке могло бути у відкритому доступі або надавати свій функціонал за платною підпискою, що може бути врази дешевше, ніж купувати нову програму під себе.

1.2. Призначення розробки та галузь застосування

Назва розроблювального продукту– «Програмне забезпечення для вибору інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation».

Основна мета розробки – оптимізація отриманих даних агромоніторингу для подальшого виявлення можливих захворювань або для прогнозування можливого стану агрокультур на відповідний період, використовуючи алгоритм вибірки ознак Whale Optimisation, що є метаевристичним алгоритмом (входить до обгорткових методів), що використовується для видалення зайвих ознак, беручи за основу поведінку горбатих китів; також покращення досвіду користувачів у роботі з аналітичними даними агромоніторингу, видача звіту про прогнозовану поведінку агрокультур чи виявленні захворювань в них.

Існує ряд причин для розробки даного продукту такі як:

- підвищення точності аналізу даних;
- ефективність обробки великих обсягів даних;
- оптимізація ресурсів;
- адаптивність до змінних умов;
- сталість сільського господарства.

Використання алгоритму Whale Optimisation дозволяє точніше визначати, які ознаки є найбільш значущими для прогнозування врожайності, виявлення захворювань рослин, а також для оптимізації використання ресурсів.

Агромоніторинг зазвичай генерує великі масиви даних. Алгоритм Whale Optimisation може ефективно обробляти ці дані, виокремлюючи найбільш інформативні ознаки з великої кількості змінних.

Завдяки вибору інформативних ознак, фермери можуть краще розподіляти ресурси, наприклад, вносячи добрива тільки в ті місця, де це необхідно, що зменшує витрати та вплив на довкілля.

Агрономічні умови постійно змінюються, і програмне забезпечення на основі Whale Optimisation може швидко адаптуватися до цих змін, забезпечуючи актуальність інформації [10].

Використання інформативних ознак допомагає впроваджувати сталі методи ведення сільського господарства, знижуючи вплив на довкілля та підвищуючи ефективність використання природних ресурсів.

Метаевристичні алгоритми оптимізації стають все більш популярними в інженерних додатках, оскільки вони:

- спираються на досить прості концепції і прості в реалізації;
- не вимагають інформації про градієнт;
- можуть обійти локальний оптимум;
- можуть бути використаний у широкому спектрі завдань, що охоплюють різні дисципліни.

Рис. 1.11. Обгортковий метод обирання ознак

Порівняно з обгортковими, фільтрові методи для вибірки інформативних ознак мають ряд недоліків:

- обмежена здатність до врахування взаємозв'язків між ознаками – фільтрові методи оцінюють значущість ознак на основі їх статистичних характеристик і не враховують взаємозв'язки між ознаками [11];

- нездатність до адаптації під конкретну модель – фільтрові методи не використовують модель прогнозування для вибору ознак, тому вони можуть вибирати ознаки, які не є оптимальними для конкретної моделі;

- не враховують реальну продуктивність моделі – оскільки фільтрові методи не використовують модель для вибору ознак, вони не можуть гарантувати, що вибрані ознаки покращать продуктивність моделі [11];

- висока чутливість до шуму в даних – фільтрові методи можуть вибирати ознаки, які виявляються інформативними лише через шум у даних, що може призвести до перенавчання моделі [11].

Алгоритм Whale Optimisation натхненний поведінкою горбатих китів під час їх унікальної стратегії полювання, відомої як «бульбашкова сітка». Ця стратегія полягає в тому, що кити створюють бульбашки навколо здобичі, формуючи спіраль, щоб заманити рибу до поверхні і легко її спіймати. Whale Optimisation імітує цю поведінку, використовуючи математичні моделі для визначення оптимальних рішень у складних оптимізаційних задачах [12].

Між цими двома стратегіями полювання кити демонструють надзвичайну координацію та співпрацю. Вони здатні синхронізувати свої дії для створення ефективної пастки, що є ключовим для успішного полювання. Аналогічно, Whale Optimisation Algorithm використовує принципи координації та адаптації для визначення найбільш ефективного шляху до оптимального рішення. Це досягається шляхом моделювання поведінки окремих китів у пошуковому просторі та їх взаємодії для досягнення певної згоди щодо найкращого рішення.

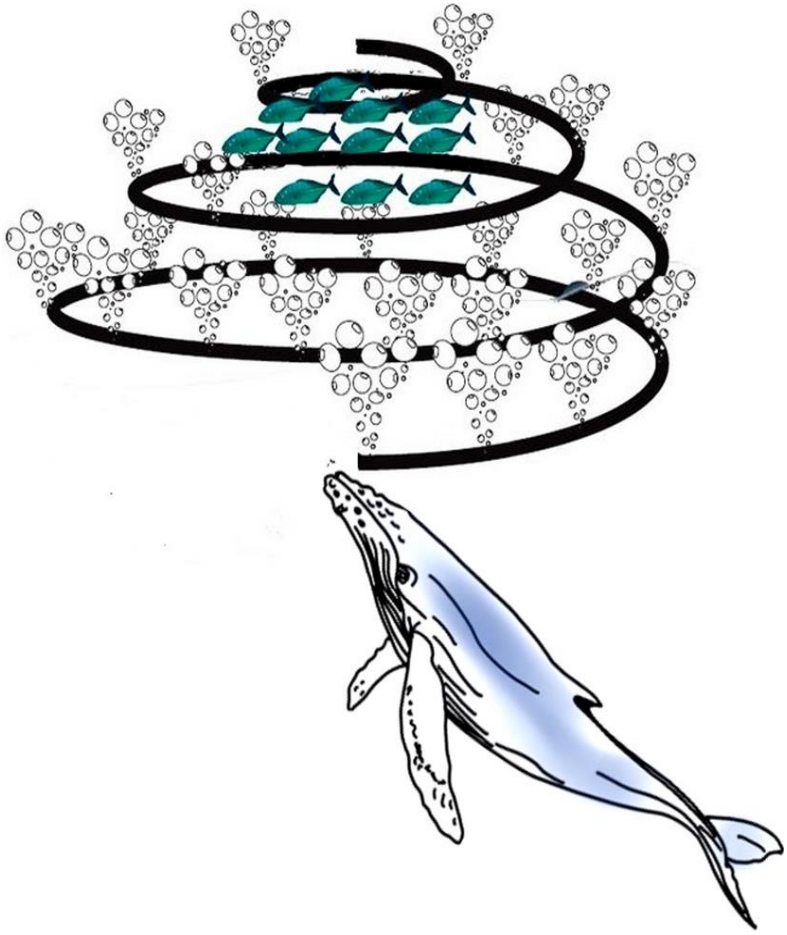


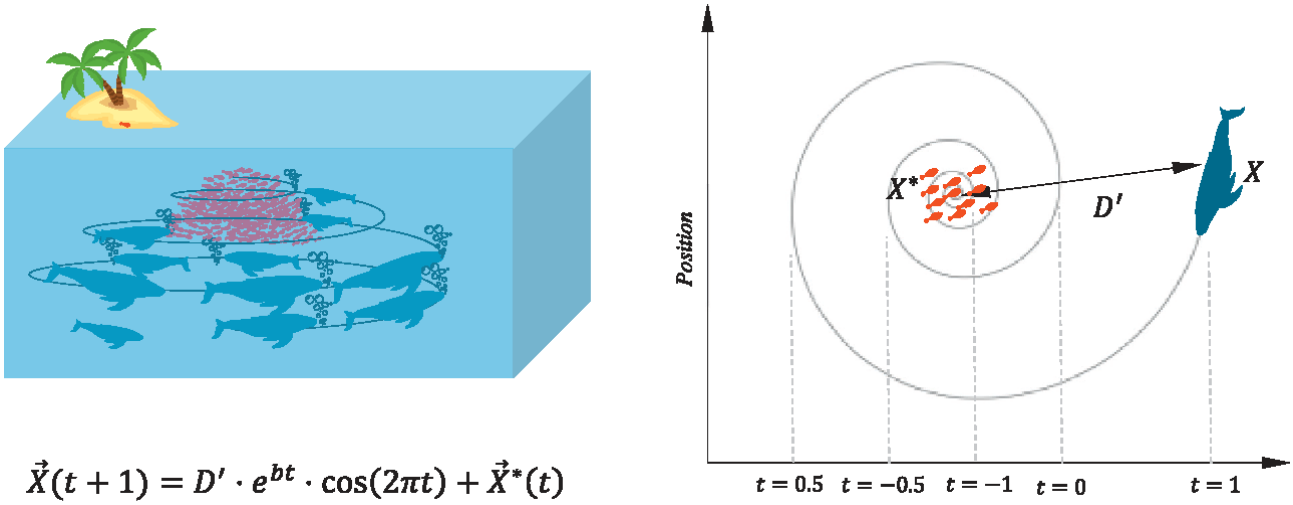
Рис. 1.12. Полювання горбатого кита а допомогою бульбашкової сітки

Два маневри, пов'язані з подачею бульбашкової сітки, - це «висхідні спіралі» і «подвійні петлі».

Під час маневру «вгору по спіралі» горбаті кити пірнають приблизно на 12 м вниз, а потім починають створювати бульбашки у формі спіралі навколо здобичі та підпливають до поверхні.

Маневр «подвійні петлі» включає в себе три різні етапи: коралову петлю, лобтейл і петлю захоплення [12].

Алгоритм складається з двох основних фаз: пошук здобичі та бульбашкова атака. Під час пошуку здобичі, алгоритм випадково переміщує пошукові агенти для глобального пошуку простору рішень. У фазі бульбашкової атаки, алгоритм використовує спіральне оновлення позицій агентів для локального пошуку навколо поточного найкращого рішення [12].



$$\vec{X}(t+1) = D' \cdot e^{bt} \cdot \cos(2\pi t) + \vec{X}^*(t)$$

Рис. 1.13. Унікальні методи живлення горбатих китів бульбашковими сітками та математична модель

Whale Optimisation має унікальну особливість, яка полягає в його здатності до ефективного переходу між глобальним та локальним пошуком. Ця властивість дозволяє алгоритму адаптуватися до різних типів оптимізаційних завдань і знаходити оптимальні рішення навіть у складних умовах.

Глобальний пошук у Whale Optimisation відповідає за дослідження широкого простору рішень. Це досягається шляхом імітації поведінки китів, коли вони розсіюються в океані в пошуках здобичі.

У цій фазі алгоритм випадково переміщує «китів» (рішення) по простору пошуку, що дозволяє виявити різні потенційні області оптимуму.

Після ідентифікації потенційної області оптимуму, Whale Optimisation переходить до локального пошуку, який імітує поведінку китів під час бульбашкової атаки [13].

Локальний пошук зосереджується на детальному дослідженні обраної області, використовуючи спіральні рухи для точного визначення оптимального рішення.

Whale Optimisation здатний балансувати між глобальним пошуком та локальним пошуком, що є ключовим для знаходження оптимальних рішень.

Завдяки своїй адаптивності, Whale Optimization може застосовуватися у широкому спектрі задач, включаючи інженерні проблеми, де потрібно знаходити оптимальні конструкції або параметри [13].

У глибокому навчанні Whale Optimization може використовуватися для оптимізації гіперпараметрів, таких як швидкість навчання або кількість шарів, що може значно покращити продуктивність нейронних мереж [13].

Алгоритм автоматично регулює свою поведінку в залежності від стадії пошуку, що дозволяє ефективно адаптуватися до складності проблеми.

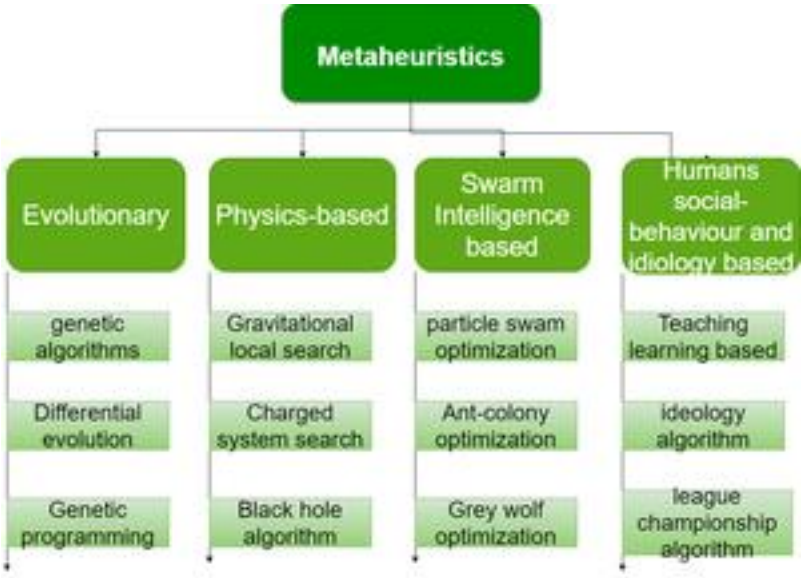


Рис. 1.14. Класифікація метаевристичних алгоритмів

Порівняння Whale Optimisation з Grey Wolf Optimisation та Ant Colony Optimisation. Whale Optimisation, Grey Wolf Optimisation та Ant Colony Optimisation є метаевристичними алгоритмами, які використовуються для оптимізації. Grey Wolf Optimisation натхненний соціальною ієрархією та полюванням сірих вовків, в той час як Ant Colony Optimisation імітує поведінку мурах у пошуку їжі та оптимізації маршрутів. Whale Optimisation вирізняється своєю унікальною стратегією пошуку, яка імітує бульбашкову сітку горбатих китів, що дозволяє ефективно знаходити рішення в багатовимірних просторах [14].

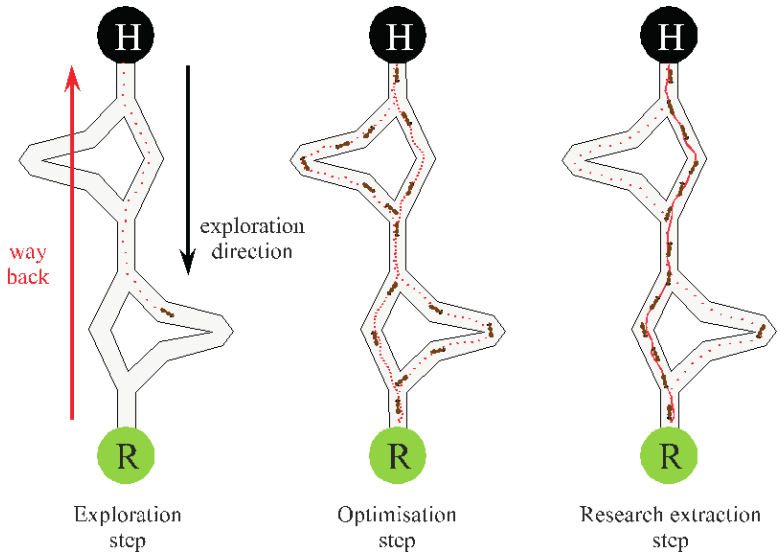


Рис. 1.15. Поведінка колонії мурах в алгоритмі Ant Colony Optimisation

У порівнянні з Grey Wolf Optimisation, Whale Optimisation має менш складну соціальну структуру, що може сприяти швидшій збіжності в деяких задачах. Ant Colony Optimisation, з іншого боку, використовує феромони для накопичення знань про простір рішень, що може бути корисним у задачах з чітко визначеними шляхами, але менш ефективним у непередбачуваних або непостійних середовищах [14].

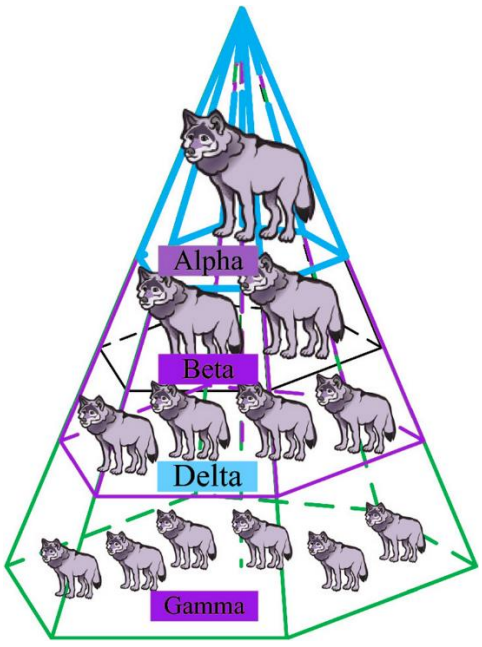


Рис. 1.16. Ієрархія сірих вовків імплементована в алгоритм Grey Wolf Optimisation

Whale Optimisation часто показує кращі результати у випадках, де потрібно балансувати між глобальним та локальним пошуком, тоді як Grey Wolf Optimisation та Ant Colony Optimisation можуть бути більш ефективними у випадках, де потрібна висока точність локального пошуку або коли проблема має чітку структуру [14].

Отже, розробка даного програмного забезпечення значно покращити якість та ефективність роботи в аграрному секторі.

1.3. Підстава для розробки

Темою розробки є «Розробка програмного забезпечення для вибірки інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation».

Організація, що затверджує цей документ є Національний технічний університет «Дніпровська політехніка».

Підставою для розробки є наказ по університету № 375-с від 29.04.2024 р.

1.4. Постановка завдання

Мета і призначення: вибрати інформативні ознаки із результатів агромоніторингу; оптимізувати роботу сільського господарства й підвищити точність та аналіз даних, що в свою чергу, може знизити ризики та витрати, пов’язані з вирощуванням культур.

Техніко-економічна сутність: використання алгоритму Whale Optimisation дозволяє ефективно обирати найбільш значущі ознаки з великої кількості даних, що надходять у вигляді CSV файлів, які містять дані від моніторингових систем, що їх фіксують. Це сприяє зменшенню витрат на добрива та зрошення, а також підвищенню врожайності.

Вихідним результатом є дані у JSON-формату, які можуть використовуватись окремо для розробників програмних продуктів для обробки

даних під власні цілі, або готовий звіт у вебформаті для користувачів, що використовують готовий користувацький інтерфейс, за допомогою чого можна провести аналіз та сформувати інформацію про:

- рекомендації по внесенню добрив – оптимальні дози та час внесення;
- прогноз врожайності – оцінка потенційної урожайності на основі аналізу даних;
- виявлення проблемних зон – зони з підвищеним ризиком втрати врожаю.

Розподіл функцій:

- персонал – моніторинг стану системи, прийняття рішень на основі аналізу даних.
- технічні засоби – автоматичний збір та обробка даних, видача обробленого набору даних.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмне забезпечення представляє собою API, до якого можна підключитись використовуючи відповідний домен та кінцеву точку (endpoint), тобто користувачі можуть використовувати його як додатковий сервіс до свого програмного забезпечення для вибірки інформативних ознак, так і як повноцінний вебзастосунок.

До програмного забезпечення додається користувацький інтерфейс, який містить опис про програмне забезпечення, домен та кінцеву точку, дані, що приймаються, дані, як повертаються, форму для заповнення необхідних даних (включно з файлом формату CSV), розділ результату, що включатиме необхідну інформацію про виконання обробки набору даних.

1.5.2. Вимоги до інформаційної безпеки

Оскільки дане програмне забезпечення є прототипом. Програмне забезпечення виконуватиме роботу на локальному вебсервері, тому безпекових вимог можна додати:

- перевірка вхідних даних для запобігання обробки помилкової або шкідливої інформації;

- логування всіх операцій з даними для забезпечення можливості аудиту та відстеження.

Якщо програмне забезпечення буде розгорнуто на певній хмарній платформі такій як AWS або Heroku, то тут додаються наступні вимоги:

- використання HTTPS для шифрування даних, що передаються;

- регулярне оновлення програмного забезпечення для усунення відомих вразливостей;

- моніторинг системи на предмет нестандартної поведінки або спроб вторгнення.

1.5.3. Вимоги до складу та параметрів технічних засобів

Основні вимоги до технічних засобів:

- мати мережеву карту (Ethernet, Wi-Fi, 3G, 4G) для роботи з програмним забезпеченням у глобальній мережі Інтернет;

- мати периферійні пристрої, що здатні візуалізувати роботу з програмним забезпечення (монітор або вбудований дисплей);

- мати периферійні пристрої для введення інформації (клавіатура, маніпулятор, сенсорний дисплей).

1.5.4. Вимоги до інформаційної та програмної сумісності

Вебінтерфейс запропонований до програмного забезпечення працює з усіма браузерами без прив'язки до конкретної версії.

Якщо користувач використовує програмне забезпечення як сервіс RESTful API, то основна вимога – це використовувати GET, POST, PUT, та DELETE запити для взаємодії з сервісом, що забезпечує гнучкість та широкий спектр функціональних можливостей.

Висновки до першого розділу

Агромоніторинг в Україні має сталий розвиток і впроваджує нові технології для покращення сільсько-господарської продукції. В даній галузі активно застосовуються IoT-технології, ШІ, робототехніка та агродрони. Завдяки IoT-технологіям аграрії відстежують показники ґрунту, рівень опадів тощо; ШІ використовується для обробки даних; робототехніка дозволяє підвищити точність роботи; дрони здійснюють швидке реагування на зміни стану рослин. Найбільш популярні компанії, що розробляють програмні продукти для сільського господарства, – ELEKS та Kernel Digital. Їхні продукти забезпечують оптимізацію виробництва, збалансування запасів матеріальних засобів, розширений лабораторний аналіз. Відсутність програмного забезпечення з відкритим доступом або за платною підпискою для оптимізації даних в аграрному секторі змушує підприємства або купувати прикладне програмне забезпечення під власні потреби, або взагалі залишатися без можливості оптимізувати свої виробничі процеси. Розроблювальне ПЗ має на меті оптимізувати дані для подальшого виявлення можливих захворювань або для прогнозування можливого стану агрокультур на відповідний період, використовуючи алгоритм Whale Optimisation, що вирізняється своєю унікальною стратегією пошуку. Програмне забезпечення представляє собою API, яке видає результат у JSON-форматі.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Програмне забезпечення приймає дані від користувача:

- назву дослідження, яке порушує певне питання (наприклад, дослідження росту пшениці при використанні нових добрив);

- набір даних, сформований на основі результатів агромоніторингу та оформлений у вигляді файлу CSV;

- кількість китів, які здійснюють вибірку даних;

- кількість ітерацій, за кожною з яких буде виконуватись переоцінка та повторна вибірка ознак з набору даних.

Функціональне призначення ПЗ: отримувати сформований набір даних, виконати обробку та видати у вигляді JSON результат про виконану роботу, що містить наступні показники як:

- найкраще рішення;

- найкращу допасовість;

- назви стовпців ознак, від яких має залежність залежна змінна;

- оброблений набір даних ознак (якщо вихідний набір не мав певні ознаки під час певних дослідів, то вони замінюються на середнє значення за стовпцем ознаки);

- набір даних залежної зміни.

Експлуатаційне призначення ПЗ: можливість інтегрувати функціонал програми в іншу, наприклад для регресивного моделювання, щоб проаналізувати врожайність агрокультур, також для візуального аналізу даних при використанні готового інтерфейсу користувача, що надає позитивний досвід при роботі з API.

Перед початком роботи користувач переходить на сторінку вебсторінки, що містить інформацію про API як для розробника, так і готову та розділ

результату, де куди буде виведено вихідну інформацію, що надає вебсервер у відповідь. Також сторінка містить навігаційну панель, якою зручно переходити до необхідних розділів.

2.2. Опис застосованих математичних методів

В основі програмного забезпечення міститься алгоритм WOA, що використовує цільову функцію або функцію допасованості, яка представляє математичний критерій, який потрібно оптимізувати для знаходження найкращих рішень.

Функція допасованості (fitness function) – це функція, яка використовується для оцінки якості рішень. Алгоритм намагається знайти рішення, що максимізує або мінімізує цю функцію в залежності від задачі оптимізації [15].

В даному випадку знаходиться рішення, що мінімізує, оскільки в середині функції використовується модель машинного навчання Random Forest Regressor, що має наступні особливості:

- забезпечення більш точної оцінки важливості ознак для прогнозування залежної змінної, оскільки враховує нелінійні взаємозв'язки та взаємодії між ознаками;

- стабільність та узгодженість оцінок важливості ознак, оскільки вони базуються на ансамблі дерев рішень.

Оскільки модель використовує T-кількість дерев рішень, то прогнози з усіх дерев у лісі усереднюються для отримання кінцевого прогнозу в формулі (2.1).

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T}\hat{y}_t, \tag{2.1}$$

де $\hat{y}$ — кінцевий прогноз,

T— кількість дерев,

$\hat{y_t}$ — прогноз від t-го дерева.

Після навчання моделі та отримання прогнозів за формулою (2.1), функція допасованості повертає значення середньоквадратичної похибки, яка вказує наскільки добре модель відповідає даним. Розрахунок значення середньоквадратичної похибки наведено у формулі (2.2).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.2}$$

де n – кількість спостережень в наборі,

y_i – фактичне значення спостереження,

$\hat{y}_i$ – прогнозоване значення для і-го спостереження.

Математична модель алгоритму WOA імітує соціальну поведінку та методи полювання горбатих китів у океанах.

Під час фази дослідження (пошуку здобичі) пошуковий агент (горбатий кит) випадково шукає найкраще рішення (здобич) на основі позиції кожного агента. Позиція пошукового агента оновлюється за допомогою випадково обраного пошукового агента, а не найкращого. Після цього, якщо |A| > 1, як визначено у формулі (2.4), то змусити пошукового агента віддалитися від опорного кита [16].

Математична модель цієї фази представлена у формулі (2.3).

$$\overrightarrow{X_{(t+1)}} = \overrightarrow{X_{\text{rand}}} - \vec{A} \cdot \left|\vec{C} \cdot \overrightarrow{X_{\text{rand}}} - \vec{X}\right|, \tag{2.3}$$

де $\overrightarrow{X_{(t+1)}}$ – вектор випадкового положення, вибраний з поточної сукупності,

t – поточна ітерація,

$\overrightarrow{X_{\text{rand}}}$ – вектор позиції здобичі,

$\vec{A}$ і $\vec{C}$ – вектори, які розраховуються за формулою (2.4) та формулою (2.5) відповідно,

$\vec{X}$ – вектор позиції кита.

$$\vec{A} = 2\vec{a}\,\overrightarrow{r_1} - \vec{a}, \tag{2.4}$$

де $\vec{a}$ – вектор, що лінійно зменшуються від 2 до 0 протягом ітерацій,

$\overrightarrow{r_1}$ – випадковий вектор.

$$\vec{C} = 2\,\overrightarrow{r_2}, \tag{2.5}$$

де $\overrightarrow{r_2}$ – випадковий вектор.

Горбаті кити оточують видобуток під час полювання. Потім вони розглядають поточне найкраще рішення-кандидат як найкраще рішення і близьке до оптимального. Модель поведінки оточення, яка використовується для оновлення позиції інших китів у бік найкращого пошукового агента, представлена у формулі (2.6) та у формулі (2.7) [16].

$$D = \left|\vec{C} \cdot \overrightarrow{X'}(t) - \overrightarrow{X}(t)\right|, \tag{2.6}$$

де t – поточна ітерація,

$\vec{C}$ – вектор коефіцієнтів,

$\overrightarrow{X'}$ – позиція найкращого рішення,

$\overrightarrow{X}$- вектор положення рішення.

$$\overrightarrow{X_{(t+1)}} = \overrightarrow{X'_{(t)}} - \vec{A} \cdot \vec{D}, \tag{2.7}$$

де t – поточна ітерація,

$\overrightarrow{X'}$ – позиція найкращого рішення,

$\vec{A}$– вектор коефіцієнтів.

Під час фази експлуатації (атака здобичі) за допомогою методу «бульбашкової сітки» горбаті кити оточують здобич під час полювання, вони

вважають поточне найкраще кандидатське рішення за найкраще та наближене до оптимального. Ця стратегія бульбашкової мережі поєднує в собі два підходи [16].

Стискальний механізм оточення. Значення в $\vec{A}$ цьому механізмі є випадковим значенням в інтервалі [-a, a], а значення зменшується $\vec{a}$ від 2 до 0 протягом ітерацій, як у формулі (2.4). Встановлення випадкових значень для $\vec{A}$ [-1, 1] нова позиція пошукового агента може бути визначена де завгодно між вихідною позицією агента та позицією поточного найкращого агента [16].

Механізм положення спірального оновлення. Цей метод починається з розрахунку відстані між китом (X, Y) і здобиччю (X', Y'). Спіральне рівняння, що лежить в основі спіралеподібного руху горбатих китів, щоб визначити положення між китом і здобиччю, представлено у формулі (2.8) [16].

$$\vec{X}(t+1) = \overrightarrow{D''} * e^{\mathrm{bl}} * \cos(2\pi \mathrm{l}) + \overrightarrow{X'}, \tag{2.8}$$

де $\overrightarrow{D''} = \left|\overrightarrow{X'}(t) - \vec{X}(t)\right|$ – відстань між китом і здобиччю (найкраще рішення, отримане на даний момент),

l – випадкове число між [-1, 1],

b – константа, що визначає форму логарифмічної спіралі.

Оскільки горбаті кити одночасно плавають навколо здобичі в межах кола, що звужується, і по спіралеподібній дорозі. Щоб змоделювати цю одночасну поведінку, ми припускаємо, що існує ймовірність 50% вибору між механізмом скорочувального оточування або спіральною моделлю для оновлення положення китів під час оптимізації. Математична модель виглядає так як представлено у формулі (2.7).

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X'}(t) - \vec{A}\,\vec{D} & p < 0{,}5 \\ \overrightarrow{D''} * e^{bl} * cos(2\pi l) + \overrightarrow{X'} & p < 0{,}5 \end{cases}, \quad (2.9)$$

де p – випадкове число в [0,1].

2.3. Опис використаної архітектури та шаблонів проєктування

Дане ПЗ містить ряд ключових ознак, що створюють певну комбінацію з декількох архітектурних рішень:

- клієнт-серверна архітектура;
- багатошарова архітектура;
- сервіс-орієнтована архітектура.

Клієнт-серверна архітектура (Client-Server Architecture) – це архітектурний шаблон, який розділяє систему на два основні компоненти: клієнт і сервер. Ці компоненти взаємодіють через мережу, де клієнт надсилає запити до сервера, а сервер обробляє ці запити та повертає відповіді [17].

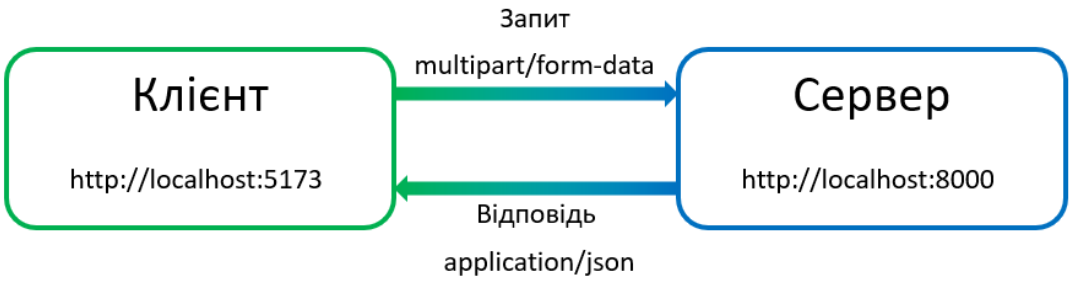


Рис. 2.1. Реалізована клієнт-серверна архітектура в ПЗ

ПЗ має веб-інтерфейс, через який користувач взаємодіє з системою. Це вказує на клієнт-серверну архітектуру, де клієнтська частина (веб-інтерфейс) відокремлена від серверної частини (бекенд). Клієнтська частина відповідає за

представлення даних, тоді як серверна – за логіку обробки даних. Клієнтом ПЗ є веббраузер, а сервером – локальний вебсервер.

Багатошарова архітектура (Layered Architecture), також відома як багаторівнева архітектура (N-tier Architecture), – це архітектурний шаблон, який організовує систему в кілька логічних шарів, кожен з яких виконує певну роль та має чітко визначені обов'язки. Ці шари розташовані вертикально, і кожен шар взаємодіє тільки з сусідніми шарами [18].

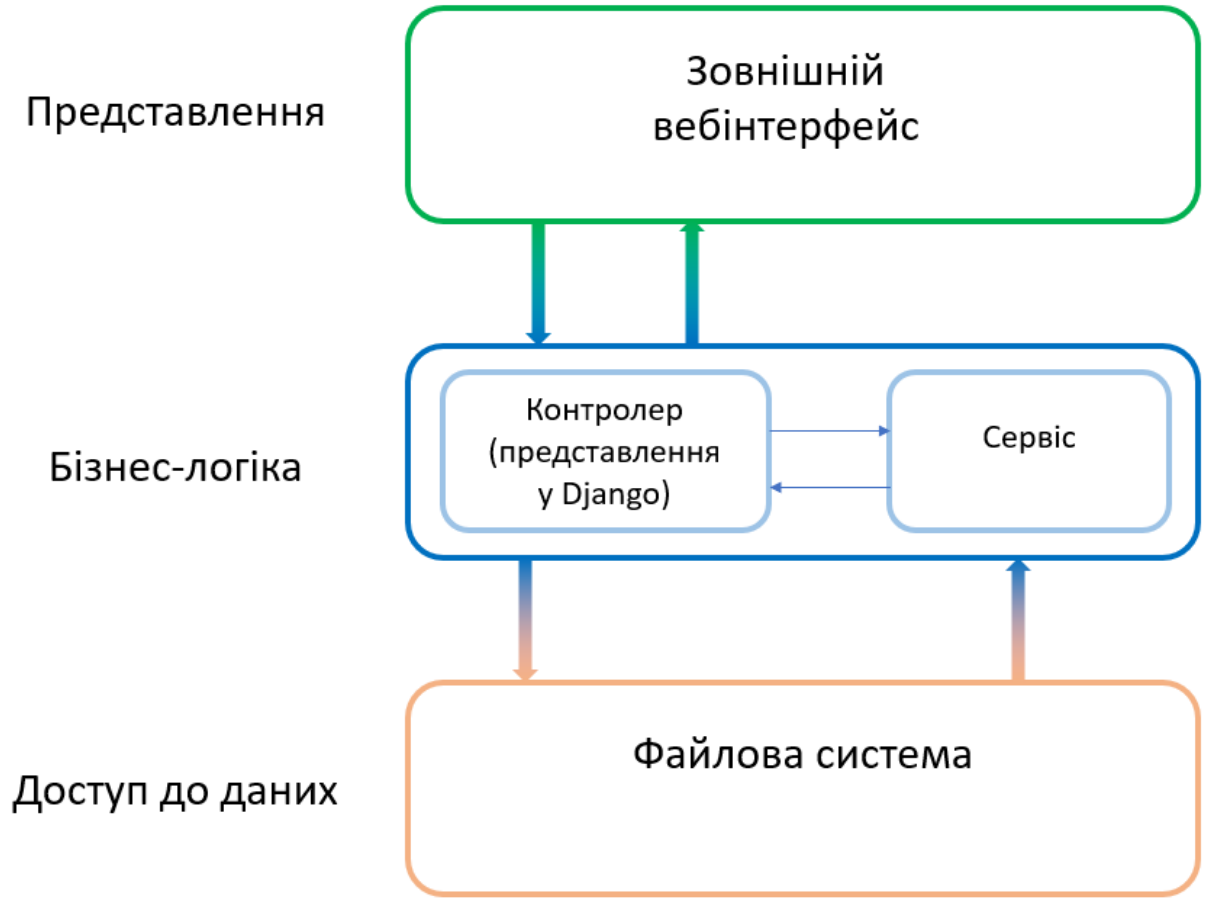


Рис. 2.2. Реалізована багатошарова архітектура в ПЗ

ПЗ може мати кілька шарів, шар презентації (веб-інтерфейс), шар бізнес-логіки (контролер та сервіс) та шар доступу до даних (взаємодія з файлами CSV).

Сервіс-орієнтована архітектура (Service-Oriented Architecture) – це архітектурний стиль, який організовує систему як набір слабко зв'язаних,

незалежних сервісів, які взаємодіють один з одним для виконання бізнес-функцій. Основна ідея SOA полягає в тому, щоб розбити складну систему на менші, більш керовані сервіси, які можуть бути розроблені, розгорнуті та масштабовані незалежно [19].

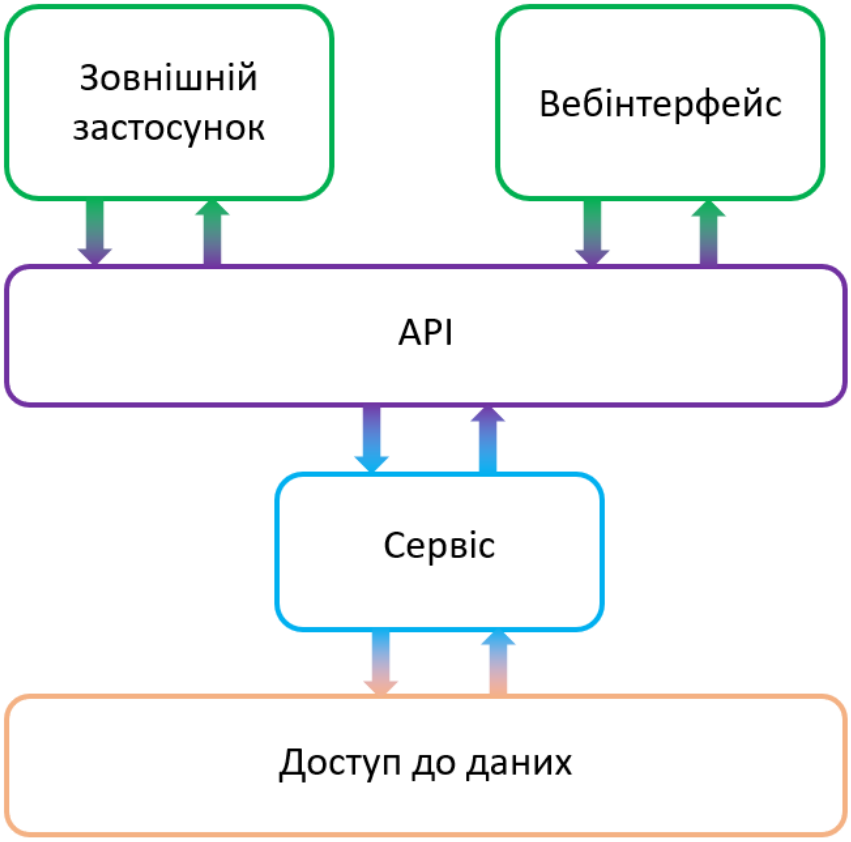


Рис. 2.3. Реалізована сервіс-орієнтовна архітектура в ПЗ

ПЗ надає функціонал через API, що дозволяє інтегрувати його в інші програми. Це відповідає принципам сервіс-орієнтованої архітектури, де функціональність надається через сервіси.

Також варто відзначити, що серверна частина, що реалізовується за допомогою фреймфорку Django, використовує шаблон проєктування MVT.

MVT — це шаблон проєктування програмного забезпечення для розробки вебзастосунків, що складається з трьох частин:

- model (модель) – те, що працює як інтерфейс доступу до даних, відповідає за збереження даних;

- view (представлення) – те, що відповідає за обробку запитів користувача та повернення відповідей, отримує дані від моделей та передає їх у шаблони для відображення;

- template (шаблон) – те, що відповідає за презентацію даних користувачеві, визначає структуру та розмітку веб-сторінок з використанням мови шаблонів Django [20].

2.4. Опис використаних технологій та мов програмування

Дане програмне забезпечення використовує різні мови програмування для реалізації функціоналу як для серверної частини, так і для клієнтської.

У таблиці 2.1 наведено перелік мов програмування, бібліотек та фремворків, що використовується для розробки програмного забезпечення.

Таблиця 2.1

Перелік мов програмування, бібліотек та фреймворків для розробки програмного забезпечення для відбору інформативних ознак з результатів агромоніторингу на основі алгоритму Whale Optimisation

	Назва	Тип	Версія/Стандарт	Застосування
1.	Python	Мова	3.12.2	Для написання серверної програми та сервісу використовуючи засоби МН
2.	Django	Фреймворк	5.0.6	Для розробки ПЗ для вебсервера

Продовж. табл. 2.1

3.	Scikit-learn	Бібліотека	1.5.0	Для попередньої обробки набору даних, тренування моделі МН
4.	Pandas	Бібліотека	2.2.2	Для виокремлення даних від файлу CSV та зручної роботи з набором даних
5.	NumPy	Бібліотека	1.26.4	Для роботи з матрицями та векторами
6.	MealPy	Бібліотека	3.0.1	Для реалізації алгоритму WOA
7.	HTML	Мова	HTML5	Для розмітки вебінтерфейсу
8.	CSS	Таблиця стилів	CSS3	Для завдання стилів для вебінтерфейсу
9.	JavaScript	Мова	ECMAScript2015	Для завдання поведінки компонентів у вебінтерфейсі
10.	React	Бібліотека	18.2.0	Для створення компонентів, завдання їм більшої функціональності в ефективніший спосіб

Закінч. табл. 2.1

11.	React DOM	Бібліотека	18.2.0	Для реалізації бібліотеки React у вебзастосунках
12.	React Router DOM	Бібліотека	6.23.1	Для створення маршрутизації різних кінцевих точок на стороні клієнта, надання зручного функціоналу в роботі зі стороннім вебсервером

Мова Python є однієї з найбільш популярних мов програмування серед розробників. Згідно рейтингу мов, що проводив вебпортал DUO у липні 2023, то Python посідає третє місце поступившись JavaScript та Java [21].

Python – інтерпретована мультипарадигмова мова програмування високого рівня із суворою динамічною типізацією. Ця мова підтримує як об’єктно-орієнтовне програмування, так і функціональне.

Python – це універсальна мова, яка використовується в багатьох галузях завдяки своїй читабельності, простоті та великій кількості бібліотек.

Вона може застосовуватись в наступних галузях:

- веброзробка;
- машинне та глибинне навчання;
- аналіз даних;
- розробка мобільних застосунків;
- розробка ігор;
- авторматизованих системах.

Третя версія мови Python містить значні оновлення, які покращили читабельність та логіку мови. Нижче наведені наступні оновлення:

- введення нового синтаксису print як функції, а не як оператора;

- підтримка анотацій типів, що дозволяє розробникам вказувати очікувані типи даних для змінних та повернення функцій;

- рядки в Python 3 завжди є в кодуванні Unicode, а байти та текст тепер чітко розділені типами bytes та str;

- використання автоматичного збирання сміття для керування пам’яттю, що зменшує ризик витоку пам’яті;

- додавання нових модулів та функцій, які розширюють можливості мови [22].

Далі розглянемо фреймворк Django, що є вдалим інструментом для розробки вебсерверного ПЗ, за допомогою якого можна розробити досить великі проєкти, і піклуватись лише про свою бізнес-логіку програми, що зменшує кількість рядків шаблонного коду під час веброзробки.

Структура базового веб-сервера на Django організована навколо проєкту та його застосунків (apps).

Дане ПЗ складається з двох застосунків зі загального woa_api та кастомного, що містить функціонал навколо сервісів, api_services. Загальний застосунок містить конфігураційний файл settings.py, файл, що визначає схеми URL-адрес всього проєкту urls.py, файли asgi.py та wsgi.py, які використовуються під час розгорнення ПЗ у хмарі. Кастомний застосунок містить файли для моделей, представлень, локальної схеми URL, тестування. На додачу цей застосунок містить файл services.py, що містить код сервісів, які використовує даний застосунок.

Як було описано раніше, фреймворк Django використовує MVT архітектурний шаблон, де ключове місце мають представлення (views). Саме представлення містять бізнес-логіку, що виконується за певною кінцевою точкою. Представлення у фреймворці Django можуть бути двох типів як у вигляді функції (Function-Base View), так і у вигляді класу (Class-Base View). Для розробки серверної частини програми застосовується класовий тип, оскільки він

більш читабельним при роботі з багатьма HTTP-методами такими як GET, POST, PUT, DELETE тощо.

Для роботи з набором даних в сервісі використовується бібліотека Pandas. Завдяки неї можна зчитати дані з файлу CSV та виконувати різні маніпуляції з даними використовуючи метод read_csv та клас DataFrame.

Бібліотека Pandas – це потужний інструмент на Python для аналізу та маніпуляції даними, який надає структури даних, такі як Series та DataFrame. Вона дозволяє читати та записувати дані в різних форматах, включаючи CSV, Excel та JSON, та містить функції для очищення, фільтрації, групування та агрегування даних [23].

Однією з найбільш важливих бібліотек, що застосовується в машинному навчанні є Scikit-learn.

Бібліотека Scikit-learn — це відкрита бібліотека для машинного навчання на Python, яка надає прості та ефективні інструменти для аналізу передбачувальних даних. Вона побудована на NumPy, SciPy та matplotlib і включає широкий спектр алгоритмів для класифікації, регресії, кластеризації та зниження розмірності. Scikit-learn легко доступна та може бути використана в різних контекстах, підтримуючи модель вибору та порівняння параметрів і моделей, а також передобробку даних [24].

Для роботи з векторами та матрицями використовується бібліотека NumPy. Ця бібліотека пропонує потужні структури даних та функції для обчислень у сферах лінійної алгебри, перетворення Фур’є та матриць. Вона є основою для багатьох інших наукових та аналітичних бібліотек Python, забезпечуючи ефективність та швидкість обробки числових даних [25].

Важливе місце для реалізації сервісу, що виконує відбір інформативних ознак, використовуючи алгоритм WOA, – бібліотека MealPy.

Бібліотека MealPy містить широкий спектр метаевристичних алгоритмів (алгоритми, натхненні природою, оптимізація чорного ящика, глобальні пошукові оптимізатори, ітеративні алгоритми навчання, неперервна оптимізація, похідна вільна оптимізація, градієнтна вільна оптимізація, оптимізація

нульового порядку, стохастична пошукова оптимізація, випадкова пошукова оптимізація). Ці алгоритми належать до популяційних алгоритмів (PMA), які є найбільш популярними алгоритмами в області наближеної оптимізації [26].

Бібліотека MealPy має клас OriginalWOA, що реалізовує в собі логіку оригінальної версії алгоритму Whale Optimisation. Даний клас приймає три ключові параметри:

- кількість китів (розмір популяції);

-кількість ітерацій, за яку виконується алгоритм.

Щоб отримати результат роботи вибірки ознак, необхідно після створення екземпляра класу викликати в ньому метод solve, що приймає параметр problem. Параметр problem може бути як словником (вбудований тип даних в мові Python), так класом, який є нащадком вбудованого класу MealPy, Problem.

Головні властивості, які має містити проблема:

- цільова функція або функція допасованості;

- межі для змінних оптимізації;

- властивість minmax для визначення чи задача є на мінімізацію, чи максимізацію.

Отриманий об’єкт, що є результату виконання методу solve, містить необхідні дані про вибірку ознак (найкраще рішення, найкращу допасовність).

Після реалізації серверної частини, перейдемо до мов та бібліотек, що використовуються у вебінтерфейсі.

Розмітка вебсторінки та стилі задаються HTML та CSS відповідно. Мова JavaScript використовується для генерування вебкомпонентів, що на виході представленні як звичайні теги HTML.

Для того щоб спростити розробку вебкомпонентів на сторінці, застосовуються бібліотеки React та React DOM.

Бібліотека React — це декларативна, ефективна та гнучка JavaScript-бібліотека для побудови інтерфейсів користувача, яка дозволяє розробникам створювати великі вебзастосунки, що можуть змінювати дані без перезавантаження сторінки. Головні переваги React порівняно зі звичайним

JavaScript включають використання віртуального DOM для оптимізації оновлень, можливість створення повторно використовуваних компонентів, а також JSX, який спрощує читання та написання коду [26].

React DOM — це бібліотека, яка дозволяє React взаємодіяти з DOM, надаючи методи для відображення компонентів у веббраузерах та працює як міст між віртуальним DOM React та реальним DOM, що дозволяє розробникам писати декларативний код, який автоматично управляється бібліотекою [27].

Також розробка вебінтерфейсу включає бібліотеку React Router DOM. Ця бібліотека застосовується для визначення шляхів (routes) та компонентів, які будуть відображатися на певних URL-адресах, забезпечуючи динамічну зміну контенту без перезавантаження сторінки.

Хоч ПЗ містить одну сторінку, воно може масштабуватись в майбутньому і мати інші сторінки (наприклад, сторінка авторизації або історії використання API). Але це не основна причина, бо головним є те, що бібліотека React Router DOM пропонує зручний функціонал (хуки), які реалізовують відстеження стану завантаження сторінки або відправлення даних, використання інших шляхів які можуть реалізовувати лише певні дії (actions), які мають логіку для виконання запиту сервера та прийняття відповіді і надсилання даних відповіді до необхідного компоненту. Один з ключових хуків, що використовується вебінтерфейсі, виконує дію за певною кінцевою точкою на клієнтській частині та виконує надсилання даних до API, – це useFetcher. Цей хук створює об’єкт, що містить ряд властивостей як submit для відправлення даних за певним action, state для визначення стану відправлення даних до сервера, тобто при значенні state як submitting, вказується на те, що дані відправлені до сервера, але відповіді ще не отримано, data для отримання даних відповіді від дії тощо.

Отже розглянуті мови, бібліотеки та фреймворк надають можливість розробити програмне забезпечення, що відповідає перед поставленими вимогами.

2.5. Опис структури програми та алгоритмів її функціонування

Центральне місце у розробці програмного забезпечення займає алгоритм Whale Optimisation, за допомогою якого можна відібрати необхідні ознаки для подальшого аналізу набору даних із результатів агромонтіторингу.

На рис. 2.4. зображено блок-схему алгоритму Whale Optimisation.

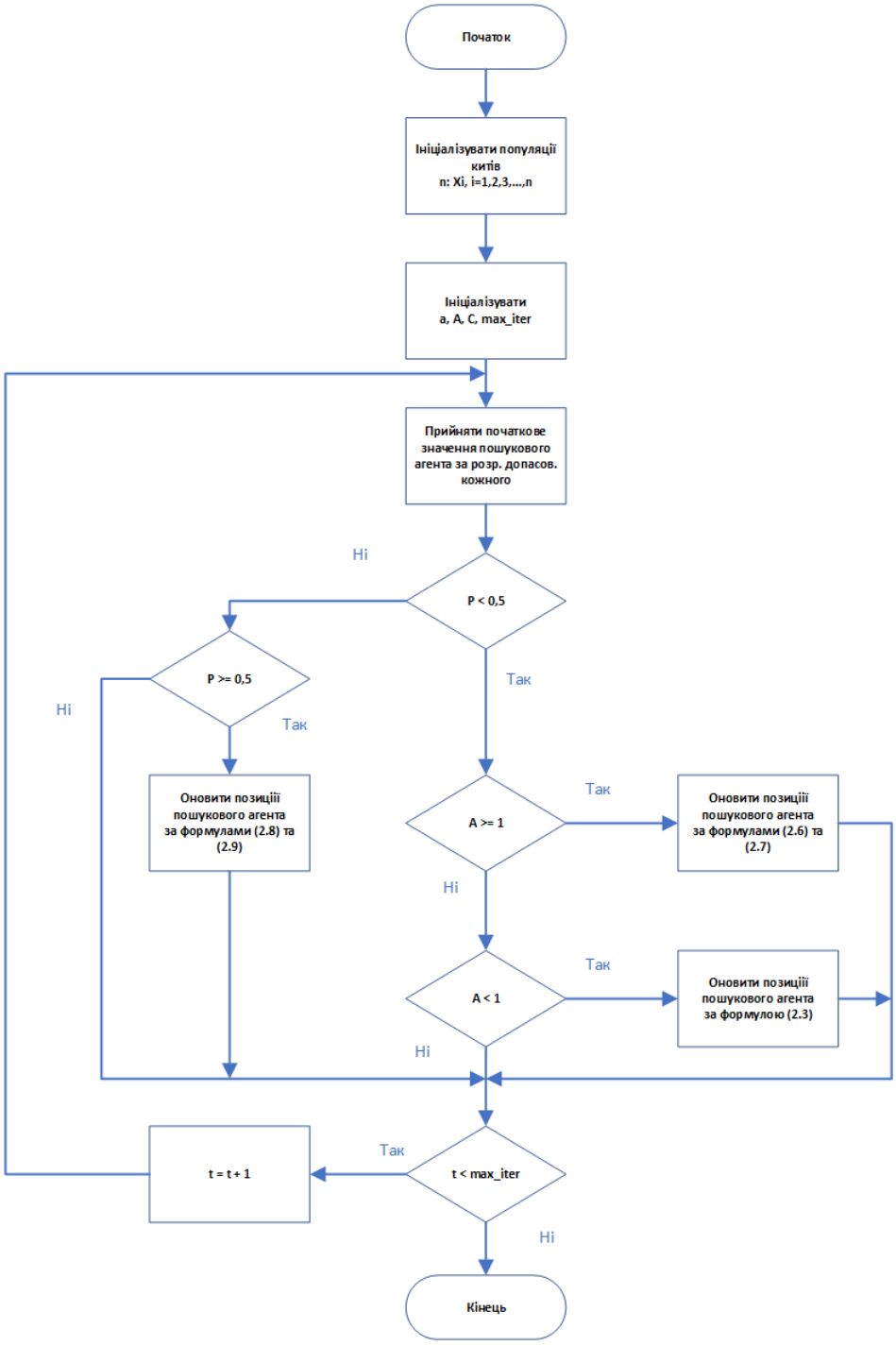


Рис. 2.4. Блок-схема алгоритму Whale Optimisation

Розглянемо принцип роботи алгоритму. Спочатку необхідно ініціалізувати популяцію n китів, а також значення. Потім оцінюємо значення рішення допасовасті пошукового агента. Далі встановлюємо кількість ітерацій рівним 1. Після цього використовуємо рівняння, що за формулою (2.3) для пошуку здобичі. Потім обводимо його за допомогою рівняння, що за формулою (2.6), знайшовши здобич. Після цього оновлюємо позицію пошукових агентів для атаки на здобич, використовуючи стратегію бульбашкової сітки в рівнянні, що за формулою (2.9). Потім оновлюємо значення a, A, C новою позицією пошукового агента. Після цього перевіряємо обмеження рівності та нерівності A p та для нової позиції кожного пошукового агента, а потім збільшуємо номер ітерації.

Нарешті, якщо досягнемо максимальної кількості ітерацій max_iter, то зупиняємось і зберігаємо значення допасованості як найкраще рішення, в іншому випадку будемо повторювати процес, поки не знайдемо рішення.

Програмне забезпечення може бути використано як звичайним користувачем, що використовує вебінтерфейс для отримання необхідних ознак, так і розробником, що використовує його як додатковий сервіс для розробки власних застосунків. Тому розглянемо UML діаграму використання для даного ПЗ.

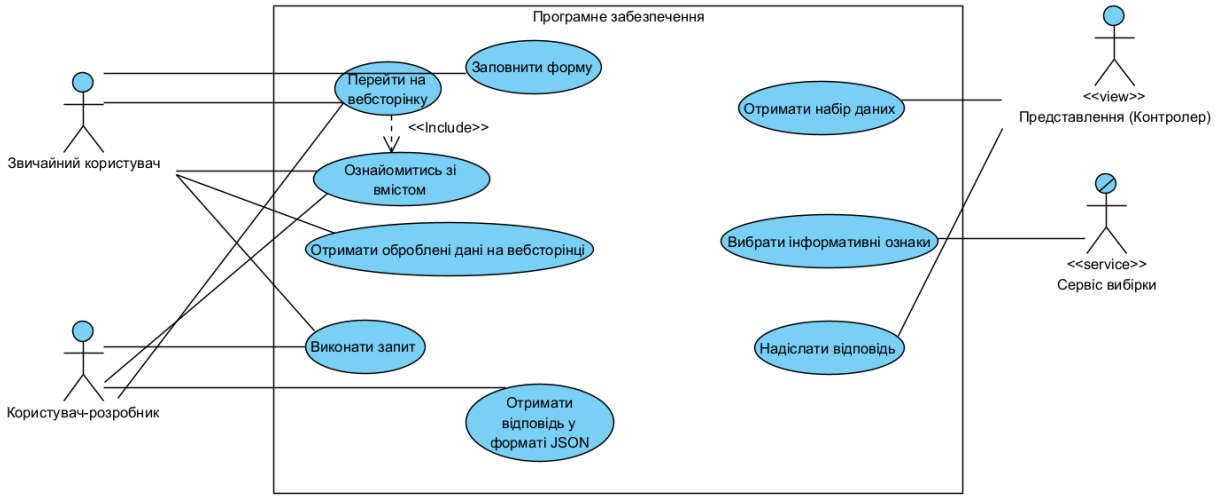


Рис. 2.5. UML діаграма використання (use case)

Пояснення до діаграми. Оскільки передбачаємо, що у нас є два типи користувача: звичайний, що може використовувати вебінтерфейс, і розробник, який хоче інтегрувати в своє програмне забезпечення відповідне API. Перед тим як кожен має почати отримувати дані від API у свій власний спосіб, для початку вони мають перейти на вебсторінку, ознайомитись зі вмістом (для розробника дослідити, які дані приймаються та відправляються у відповідь). Далі користувач заповнює форму, чекає на відповідь та отримує на тій же вебсторінці результат роботи API, в той час як розробник використовує API через власний програмний продукт або тестує через спеціалізовані ПЗ такі як Postman та отримує результат у вигляді JSON документу. Також можна виокремити ще два актори, це Представлення, який отримує запит та надсилає відповідь, та Сервіс, який виконує вибірку ознак.

Більш деталізовану роботу програми як зі сторони звичайного користувача, так і користувача-розробника, розглянемо в UML діаграмах діяльності.

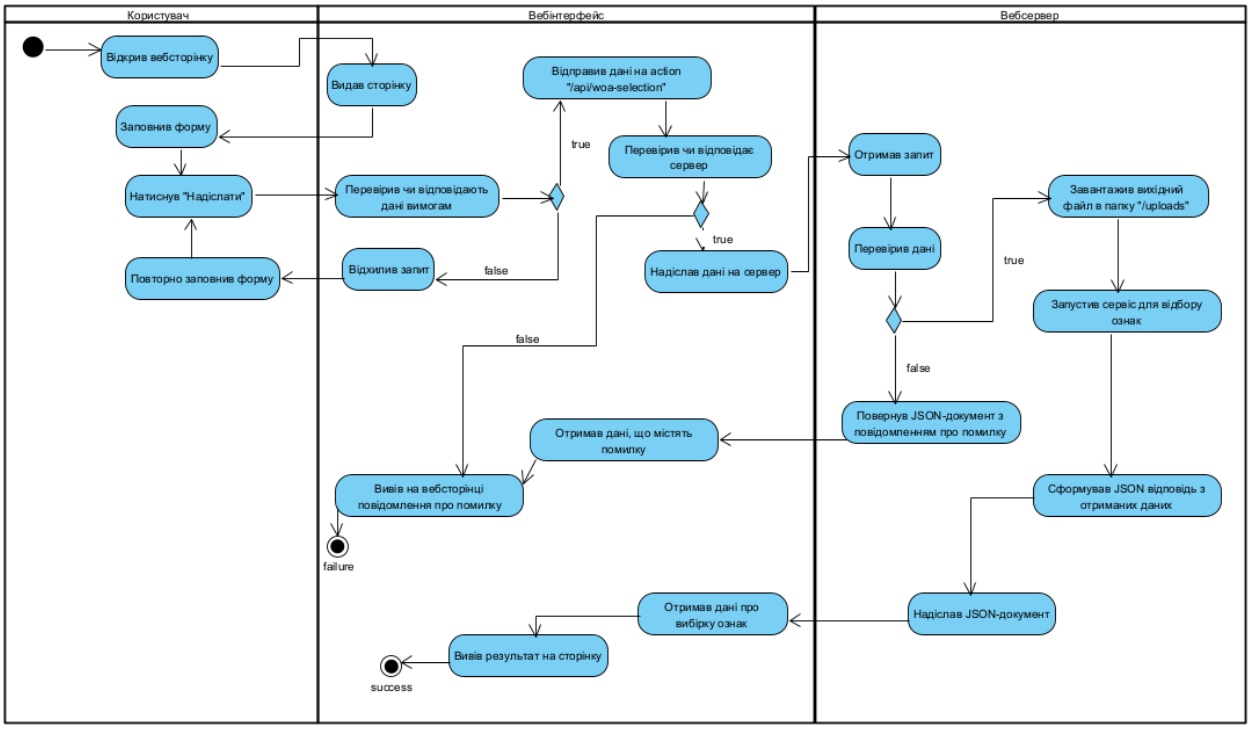


Рис 2.6. UML діаграма діяльності ПЗ зі сторони звичайного користувача

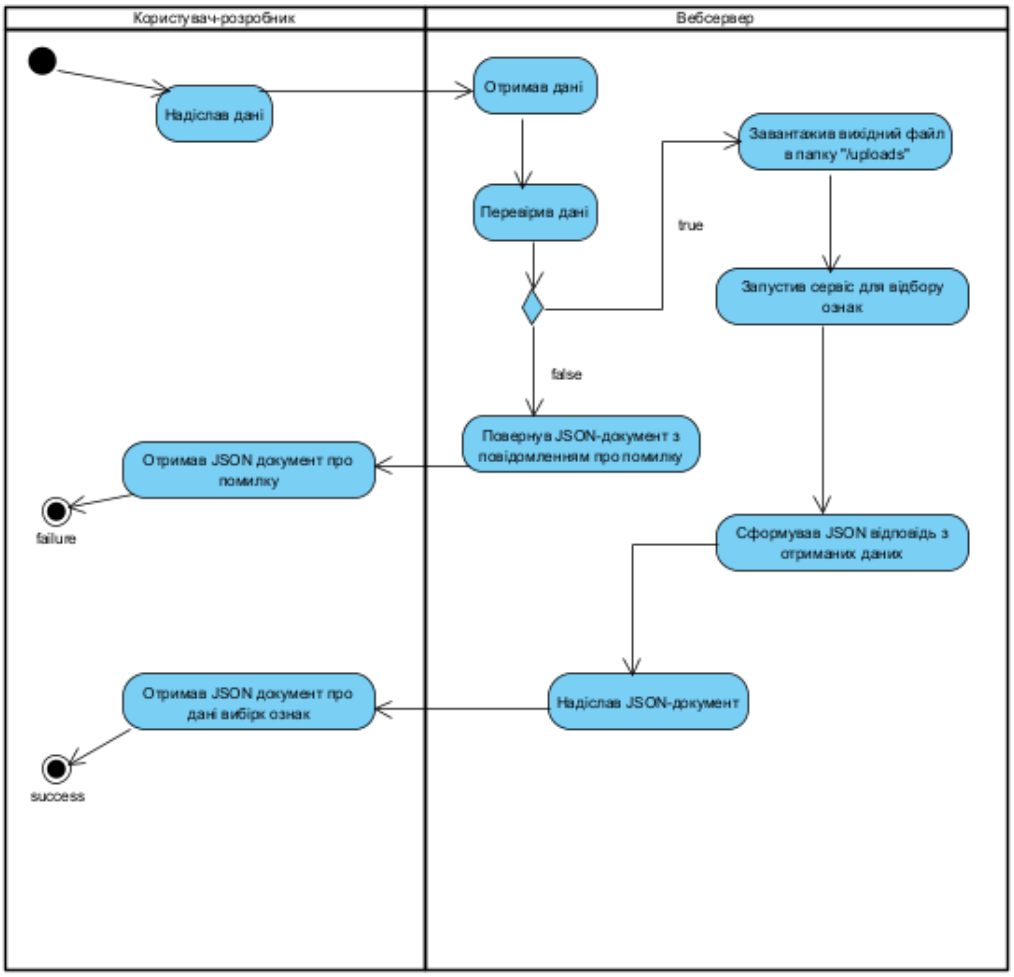


Рис. 2.7. UML діаграма діяльності ПЗ зі сторони користувача-розробника

Пояснення до вище наведених діаграм. У першій діаграмі представлено три суб’єкти, яким притаманні певні дії. Спочатку починає роботу Користувач, він заходить на вебсторінку, Вебінтерфейс має йому видати необхідну сторінку, на цій сторінці Користувач вводить форму та натискає «Надіслати». Як тільки натиснута кнопка, то Вебінтерфейс перевіряє чи задовольняють введені дані вимогам, якщо так, то свою роботу виконує action, якщо ні то відхиляється запит. Далі йде перевірка чи відповідає сервер, при відповіді відправляється запит, якщо ні, то виводиться помилка на сторінку. Вебсервер приймає дані та на своїй стороні також перевіряє дані, якщо вони не задовольняють вимогам, то повертається JSON документ з інформацією про помилку. Потім Вебсервер запускає сервіс, який видає необхідні дані, які потім надсилаються у JSON форматі до Вебінтерфейса. Той приймає дані та виводить їх на сторінку.

Друга діаграма є більш спрощеною, ніж попередня, оскільки надсилання та обробка отриманих даних від Вебсервера лежить на відповідальності Користувача-розробника, а це є поза межею роботи ПЗ.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення приймає дані, що містять інформацію про:

- назву дослідження (research_name);
- файл CSV з ознаками та залежною змінною (csv_file);
- кількість китів (whale_num);
- кількість ітерацій (iter_num).

У таблиці 2.2 представлено перелік вхідних даних, які приймає API.

Таблиця 2.2

Перелік вхідних даних для вибірки інформативних ознак

	Ключ	Тип даних	Примітки
1.	research_name	text	Має бути не пустим і довжина тексту не менше 5 символів
2.	csv_file	file	Має бути з розширенням csv та не бути null
3.	whale_num	number	Має бути не менше 5 і не більше 10000
4.	iter_number	number	Має бути не менше 10 і не бути більше 10000

Дані відправляються на сервер, використовуючи HTTP метод POST, мають тіло запиту, що закодовані як multipart/form-data, оскільки дані містять файловий тип.

Multipart/form-data є особливим типом кодування для передачі даних форми через HTTP. Він використовується, коли необхідно передати одночасно різні типи даних, таких як текст, файли та інші бінарні дані, що неможливо зробити за допомогою звичайного типу кодування application/x-www-form-urlencoded [28].

Кодування multipart/form-data дозволяє розділити дані на кілька частин (multi-part), де кожна частина може містити різні типи даних, включаючи текст, файли та інші бінарні дані. Кожна частина розділяється спеціальним роздільником (boundary), який визначається під час надсилання запиту [28].

Передача даних відбувається за допомогою HTTP методів, один з це POST.

У відповідь користувачі отримують JSON документ, що містить наступні дані, які показані на рис. 2.7., коли дані успішно оброблені, та на рис. 2.8, коли виникла певна помилка.

```
{
  "status": "success",
  "data": {
    "research_title": "...",
    "best_solution": [...],
    "best_fitness: 123.456,
    "selected_features_columns": [...],
    "X": [
      {
        "feature_0": 123,
        ...
      },
      ...
    ],
    "y": [
      {
        "result": 321,
      },
      ...
    ],
  }
}
```

Рис. 2.7. Формат відповіді у вигляді JSON, коли дані успішно оброблені

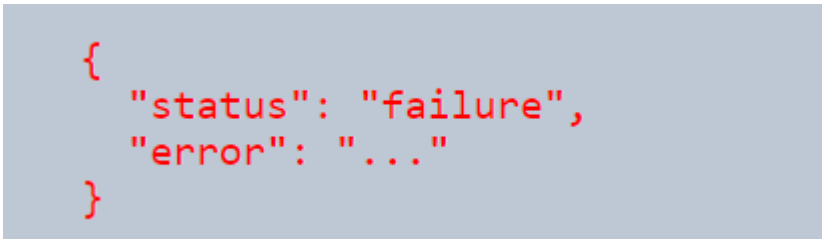


Рис. 2.8. Формат відповіді у вигляді JSON, коли виникла певна помилка

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Програмне забезпечення умовно ділиться на дві частини – це на клієнтську частину та серверну.

Клієнтська частина не вимагає значних технічних показників, тому може працювати на будь-якому пристрої, що має доступ до Інтернету. Але оскільки розроблене ПЗ є прототипом повноцінного продукту, що може бути розгорнуто на хмарі, то запустити програму можна на пристроях типу як ПК, ноутбук або моноблок.

Серверна частина передбачає наступні характеристики:

- розмір оперативної пам’яті не менше 8 Гб;
- кількість ядер процесора не менше 4;
- в якості накопичувача використовувати SSD, обсягом не менше 256 Гб;
- рік випуску процесорів не мають бути старше 2015 року.

Дане розроблювальне ПЗ використовує локальні порти для хостингу, як для клієнтської частини, так і для серверної.

2.7.2. Використані програмні засоби

Для того, щоб працювало ПЗ необхідно встановлені деякі програмні інструменти, що змогли б запустити програму. Серед Node та NPM.

Node.js (або просто Node) – це крос-платформова runtime-середовище для виконання JavaScript коду поза браузером.

NPM (Node Package Manager) – це менеджер пакетів для Node.js, допомагає встановлювати, видаляти та керувати пакетами (бібліотеками) в Node.js застосунках.

Ці пакети застосовуються для того, щоб запустити вебінтерфейс, який буде розміщено на локальному хості на порту 5173.

Для розробки вебінтерфейсу програмного забезпечення використовується інструмент для вебзастосунків Vite. Основними особливостями Vite є швидкість старту, швидка розробка з HMR і ефективна збірка застосунків для продуктивного середовища. Vite завантажує та пакує React компоненти майже миттєво завдяки своїй системі модульної збірки на основі ES модулів. Цей інструмент значно підвищує продутивність проєктів написані на React, а також використовує сучасні техніки мініфікації та оптимізації коду.

Для серверної програми необхідно встановити Python та PiP. Python – мова програмування, на якій написано фреймворк Django та бібліотеки, що пропонують працювати з моделями машинним навчанням та вибіркою ознак. PIP – це менеджер пакетів для мови програмування Python, що використовується для встановлення, оновлення та видалення пакетів.

Для розробки ПЗ використовується текстовий редактор Visual Studio Code 2022 (для розробки вебінтерфейсу) та PyCharm 2024.1.1 (для розробки серверної частини).

2.7.3. Виклик та завантаження програми

Для запуску програмного забезпечення необхідно виконати декілька кроків.

Спочатку запускається серверна частина:

- потрібно перейти до терміналу або командного рядку (Power Shell);

- виконати команду для переходу до папки woa-api (папка зі серверною частиною ПЗ);

- виконати команду python manage.py runserver.

Після цього буде запущено серверний застосунок на Django на порту 8000. Правильність роботи застосунку можна перевірити на за допомогою спеціальних програм, наприклад Postman.

Потім запускається клієнтська частина:

- - потрібно перейти до терміналу або командного рядку (Power Shell);

- виконати команду для переходу до папки service-web-page (папка зі клієнтською частиною ПЗ);

- виконати команду npm run dev.

Після цього вебінтерфейс на React на порту 5173.

У підсумку розроблювальне програмне забезпечення буде повністю запущено і може використовуватись повний функціонал.

2.7.4. Опис інтерфейсу користувача

Розроблювальне програмне забезпечення містить вебінтерфейс, за допомогою якого користувач може дізнатись як працює API сервісу вибірки ознак, також повноцінно використовувати вже готове клієнтське рішення, яке дає досить зрозумілу візуалізацію результатів та приємний досвід у користуванні API.

Інтерфейс складається зі сторінки, де розміщену весь необхідний зміст, а також на додачу сторінку помилки, яка може з’явитись, якщо ви перейшли на неіснуючий шлях.

Основна сторінка містить:

- навігація по сторінці;

- три розділи: «Про API», «Вибірка ознак», «Результат»;

- футер.

На рис. 2.9. показано вебсторінку з навігацією та розділом «Про API».

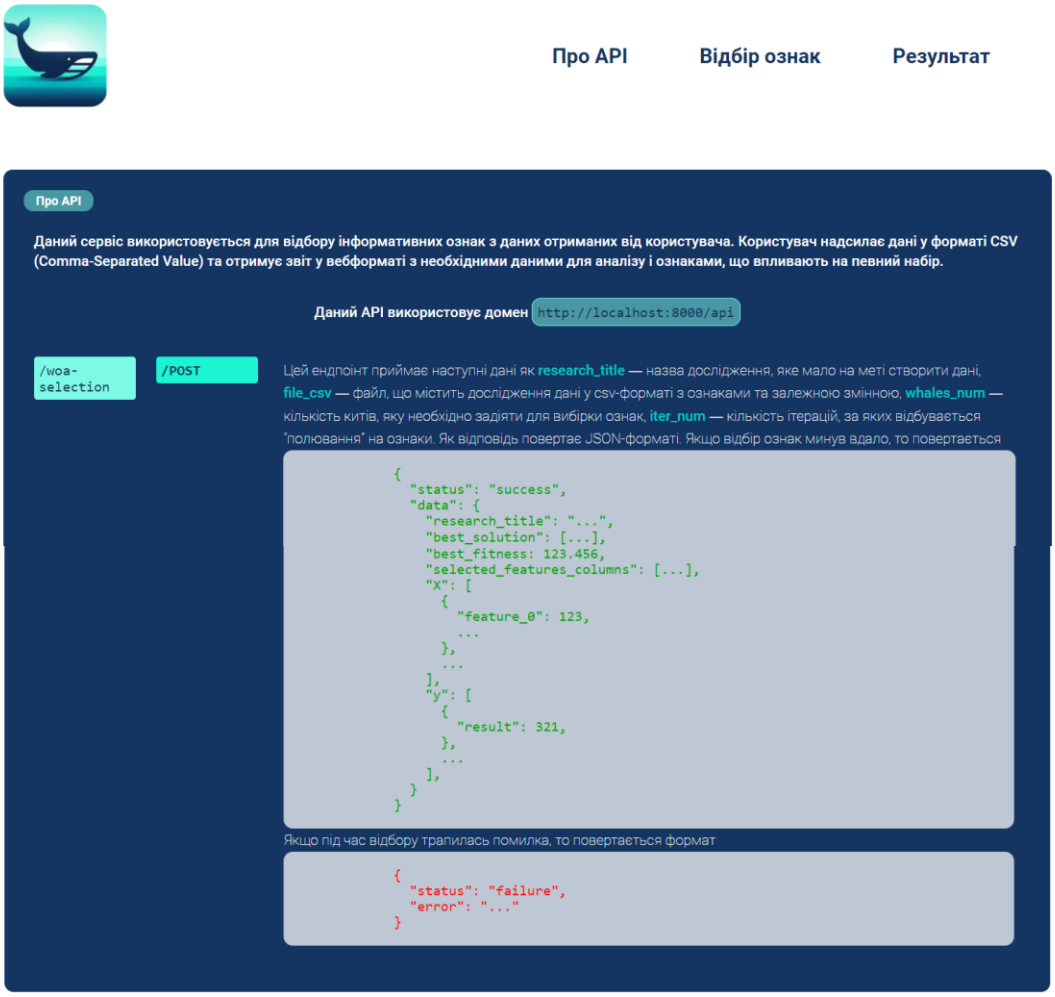


Рис. 2.9. Вебсторінка з навігацією та розділом «Про API»

Як показано на рис. 2.9., сторінка містить опис про що є дане програмне забезпечення, вказано домен, на якому хоститься сервер, кінцеву точку та HTTP метод, який потрібно використовувати для отримання результату. Також наведені фрагменти JSON документів, що повертаються до користувача, при успішному виконанні програми або при виникненні помилки.

У верхньому блоці сторінки розміщено також логотип, що робить сервіс більш впізнаваним та надає основні кольори, які необхідно застосовувати у сторінці.

На рис. 2.10. показана анімоване посилання на навігаційній панелі при наведенні курсору на нього.

Рис. 2.10. Анімоване посилання на навігаційній панелі

Далі перейдемо до розділу сторінки «Вибірка ознак», що показано на рис 2.11.

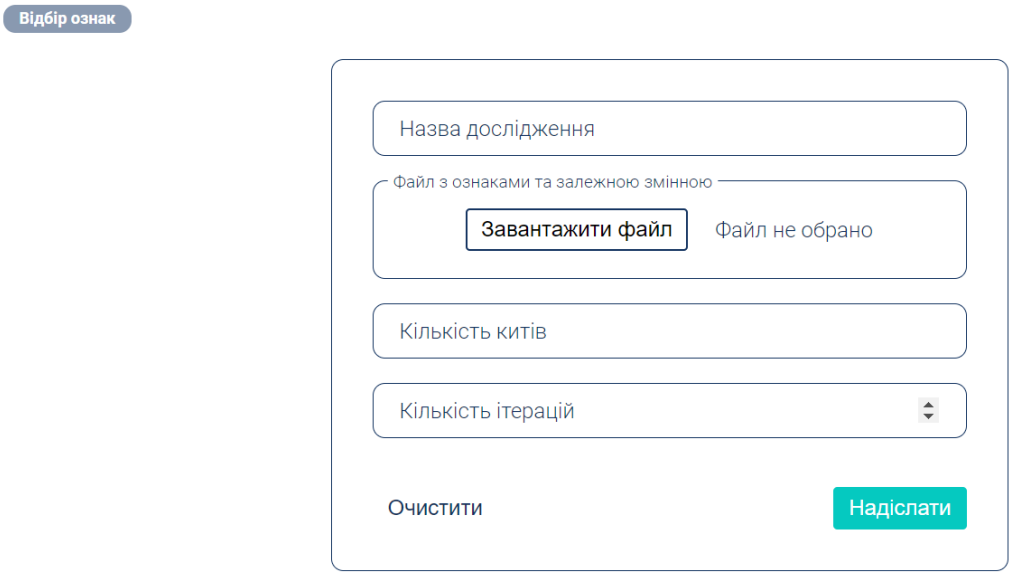


Рис 2.11. Розділ сторінки «Вибірка ознак»

Розділ «Вибірка ознак» представляє собою форму, яка приймає дані від користувача такі як:

- назва дослідження;
- файл з ознаками та залежною змінною;
- кількість китів;
- кількість ітерацій.

Також форма містить кнопки «Очистити» - для очищення даних у формі, кнопку «Надіслати» - для відправлення даних на обробку.

Розглянемо розділ «Результат» та футер сторінки, що показані на рис. 2.12.

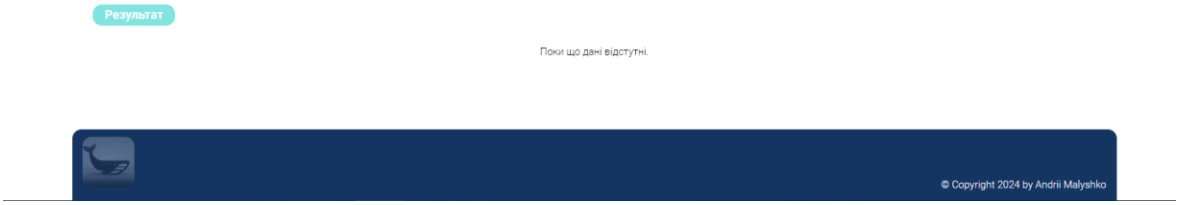


Рис. 2.12. Розділ «Результат» та футер сторінки

Як видно зі рис. 2.12. розділ «Результат» має лише повідомлення «Поки що даних немає». Це вказує, що жодного запиту з форми ще не було, тому необхідно заповнити форму та отримати візуалізований звіт.

На рис. 2.13. показана заповнена форма з даними.

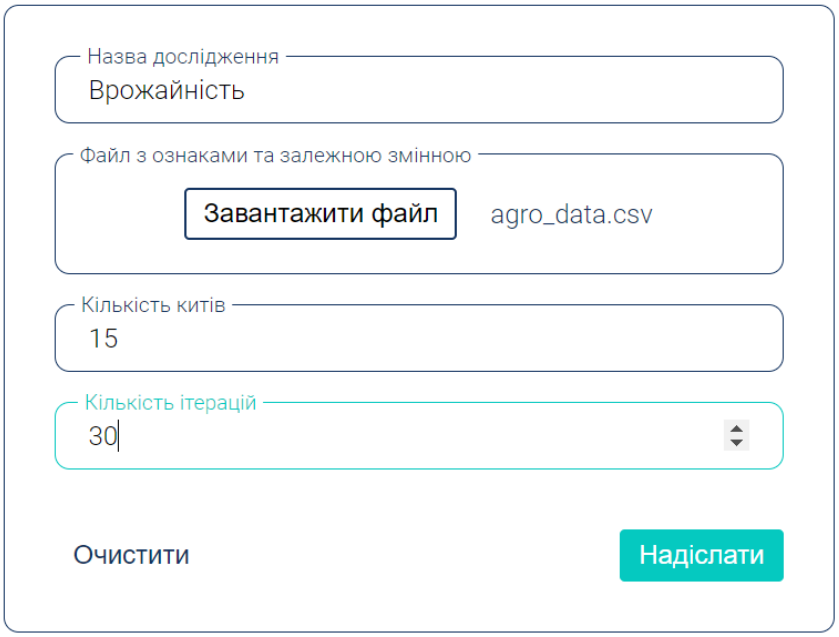


Рис. 2.13. Заповнена форма з даними

На рис. 2.14. показано отриманий результат у вигляді звіту.

Результат

Врожайність

Найкраще рішення

```
[
0.004160927774113254
-0.5063227993415536
-0.06117067641153118
0.5193048574333547
0.9139728525413522
-0.5788562101661061
0.1972150968074533
-0.8124411476098914
0.7847242345545475
-0.5044592373608268
]
```

Найкраща допасованість: `116.92082658940764`

Обрані ознаки: `feature_3` `feature_4` `feature_8`

Кількість необхідних ознак: `3`

	feature_3	feature_4	feature_8	yield
1.	0.6313247110151056	0.034114775730719	0.7469319707485604	53.51195596058224
2.	0.1313249388936997	0.5192372516100988	0.7965036802353896	72.88793060193605
3.	0.3536115558285676	0.3099920594160091	0.5210168484389118	44.74523883499297
4.	0.6702349528694351	0.0627400460275782	0.4912928313038615	47.9799955954524
5.	0.519342899586074	0.4983753622735169	0.49845346756894127	37.54925209001354
6.	0.5054362721655434	0.7915768826814156	0.1299210180923021	68.57251477927421
7.	0.4386103358623171	0.3401605101268135	0.2911655141755328	45.007831506177226
8.	0.519342899586074	0.4983753622735169	0.49845346756894127	48.89091141170847
9.	0.4315077035985906	0.6069981053126718	0.0796684569596631	38.69996963029164
10.	0.519342899586074	0.4983753622735169	0.49845346756894127	54.18212027817295
11.	0.0352157131454363	0.4984940222294294	0.2875755988786845	44.84173802826837
12.	0.3979325465858967	0.5962396334748841	0.0043058648395066	48.87162870057736
13.	0.519342899586074	0.4983753622735169	0.49845346756894127	58.69104581478519

Рис. 2.14. Отриманий результат у вигляді звіту

Для зручного використання таблицею зі зменшеною кількістю ознак, було застосована задання стилю на рядок, на який наводиться курсор (як показано на рис. 2.15.).

	feature_3	feature_4	feature_8	yield
1.	0.6313247110151056	0.034114775730719	0.7469319707485604	53.51195596058224
2.	0.1313249388936997	0.5192372516100988	0.7965036802353896	72.88793060193605
3.	0.3536115558285676	0.3099920594160091	0.5210168484389118	44.74523883499297
4.	0.6702349528694351	0.0627400460275782	0.4912928313038615	47.9799955954524
5.	0.519342899586074	0.4983753622735169	0.49845346756894127	37.54925209001354
6.	0.5054362721655434	0.7915768826814156	0.1299210180923021	68.57251477927421

Рис. 2.15. Таблиця з наведеним рядком

Може бути ситуація, коли наприклад сервер не відповідає, якщо це так, то може викидуватись помилка, яку потрібно перехопити, тому було використано різні повідомлення про помилку, на тій же сторінці, де і надсилався запит. На рис. 2.16. показано повідомлення про помилку при відсутності вдалого з'єднання до сервера.

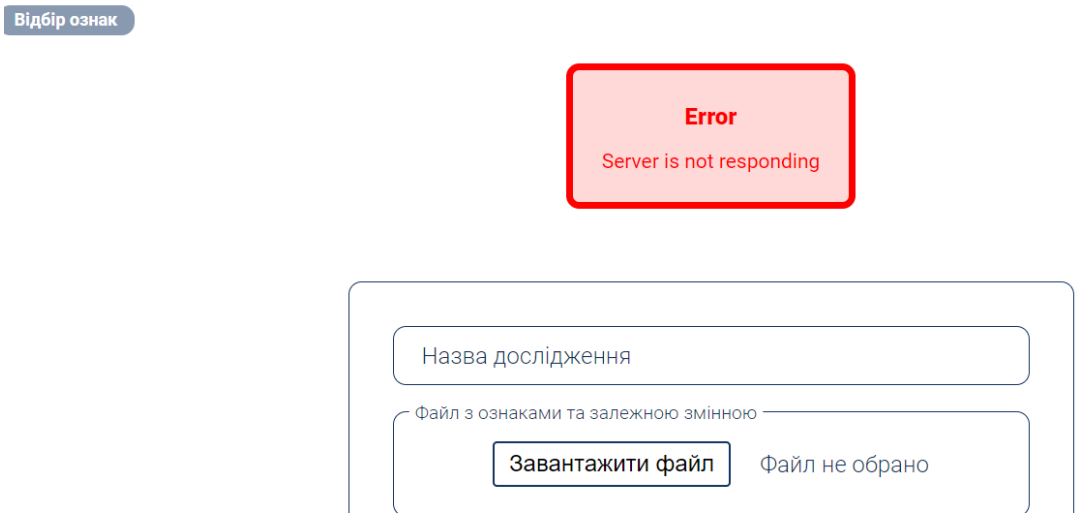


Рис. 2.16. Повідомлення про помилку при відсутності вдалого з'єднання до сервера

А якщо користувач випадково перейде до неіснуючого маршруту на вебінтерфейсі, то буде показана сторінка про помилку (як на рис. 2.17.).

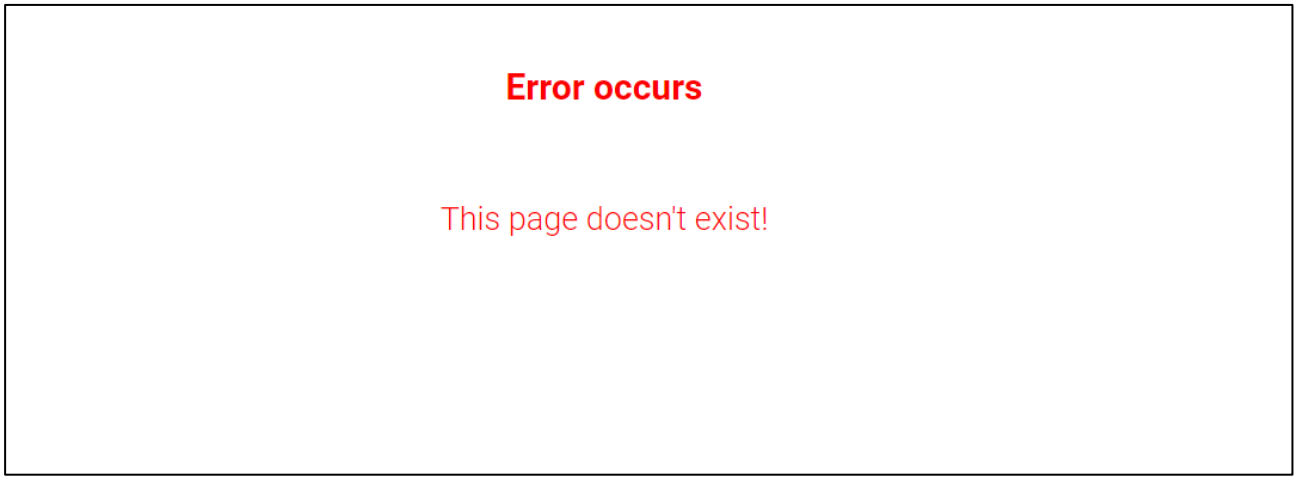


Рис. 2.17. Сторінка про помилку

Для покращення користувацького досвіду використовується ще один компонет – це завантажувач, який з’являється, коли надсилаються дані з форми на сервер, а відповіді ще немає. Цей компонент займає усю сторінку, що не надає користувачеві натискати декілька разів на кнопку надіслати, а також має анімовані фігурки, які пульсують з певним відставанням. Такий підхід покращує досвід користування вебінтерфейсом.

На рис. 2.18. зображено завантажувач на сторінці.

Рис. 2.18. Завантажувач на сторінці під час надсилання даних до сервера

Інтерфейс ПЗ також адаптовано під різну ширину дисплею, що робить більш зручним для користування не лише з ПК та ноутбука, а й мобільних пристроїв.

Завдяки медіа-запитам від CSS, а також сучасним стилям для задання макету сторінки (flexbox, grid) можна досягти повну адаптацію вебінтерфейса під різні розміри екранів.

На рис. 2.19. показано скриншот сторінки, де ширина екрану встановлена на 1605px. Завдяки цьому, якщо екран користувача досить величезний, і щоб сторінка не здавалась занадто маленькою або не розтягнулась на всю ширину екрану, він зможе зручно користуватись ПЗ.

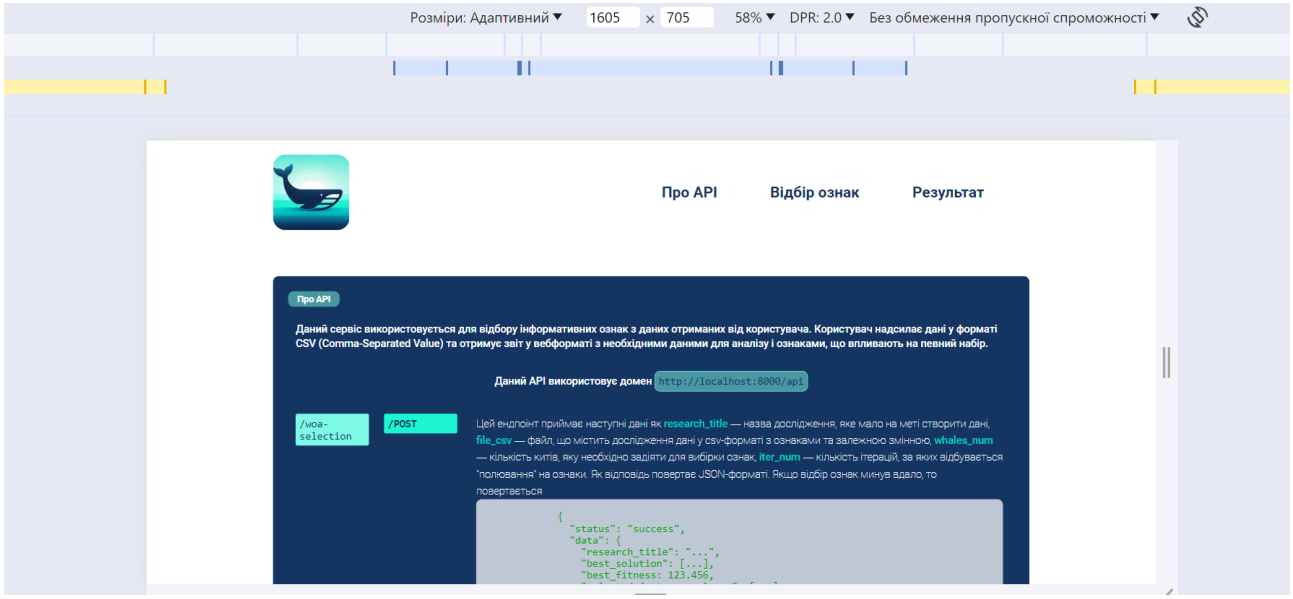


Рис. 2.19. Скриншот сторінки, де розміри екрану встановлені на 1605px

На рис. 2.20. показано скриншот сторінки, де ширина екрану встановлена на 388px.

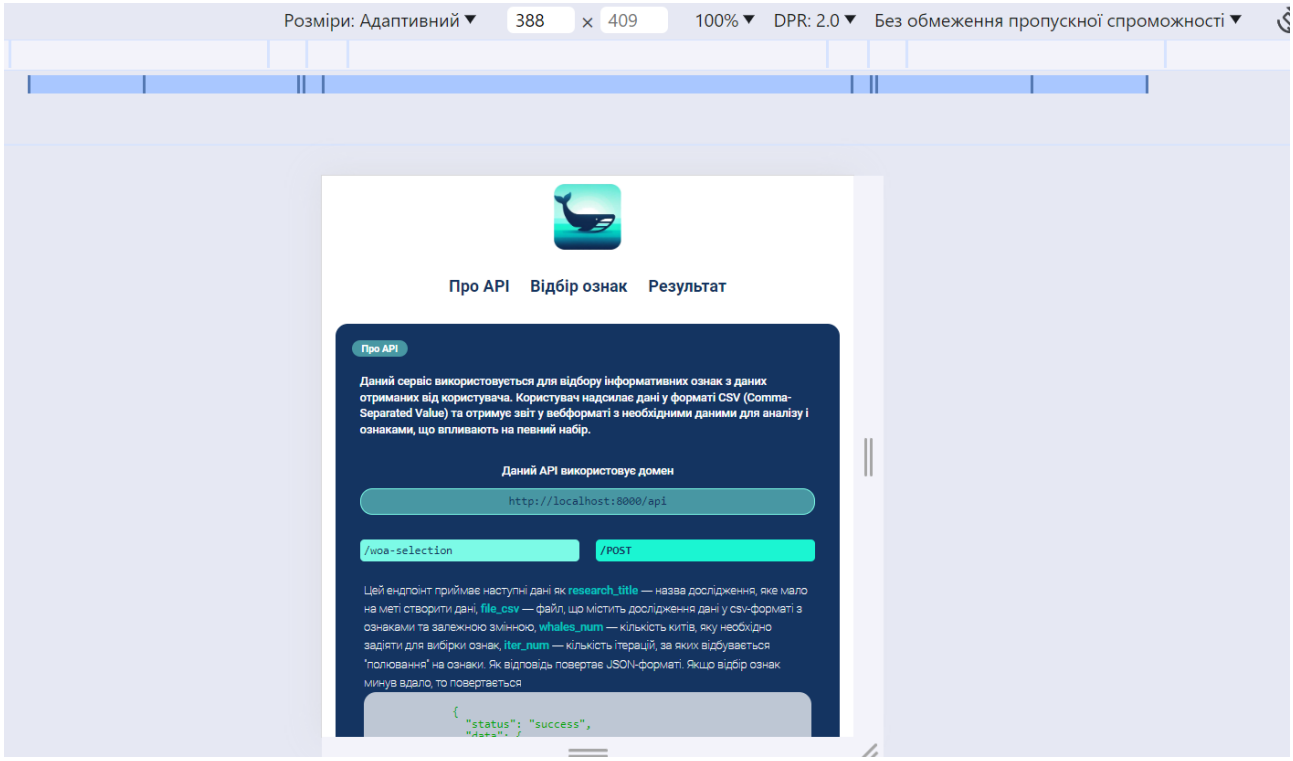


Рис. 2.20. Скриншот сторінки, де ширина екрану встановлена на 388px

Висновки до другого розділу

ПЗ приймає такі назви як назву дослідження, набір даних у вигляді файлу CSV, кількість китів, кількість ітерацій. ПЗ отримує дані, оброблює набір з ознаками та повертає JSON-документ про результат вибору ознак, що вміщує найкраще рішення, найкращу допасованість, назву стовпців, оброблений набір даних ознак та залежної змінної. Розроблювальне ПЗ можна інтегруватись в інші програми, доповнюючи їхній функціонал. В основі програми лежить сервіс, що виконує вибір ознак за алгоритмом WOA. Для забезпечення більш точної оцінки, стабільності та узгодженості оцінки важливості ознак використовується модель машинного навчання Random Forest Regressor. Математична модель алгоритму WOA імітує соціальну поведінку та методи полювання горбатих китів в океанах, що включає фазу дослідження, фазу експлуатації з двома підходами: стискальним механізмом оточення та механізмом положення спірального оновлення. В розроблювальному програмному забезпеченні можна розгледіти три архітектурні рішення такі як клієнт-серверна, багатошарова та сервіс-орієнтовна архітектури. Також шаблон проєктування MVT використовується в серверній частині, оскільки такий шаблон використовує фреймворк Django, на якому написано серверна частина ПЗ. Як на серверній частині, так і на вебінтерфейсі використовуються різні мови програмування, бібліотеки та фреймворки. При використаних програмних та технічних засобів розроблювалось програмне забезпечення, що дозволило реалізувати бажаний функціонал. Вебінтерфейс ПЗ містить навігацію по сторінки, три розділи та футер. В розділі «Про API» включає інформацію про API; в розділі «Відбір ознак» міститься форма, яка приймає дані від користувача; в розділі «Результат» виводиться звіт про оброблений набір даних. Програмне забезпечення може бути використано як звичайним користувачем, що використовує вебінтерфейс для отримання необхідних ознак, так і розробником, що використовує його як додатковий сервіс для розробки власних застосунків. Вебсторінка адаптована до різних розмірів екранів, що надає більше зручності для користувача.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

В таблиці 3.1 наведено початкові дані для розрахунку трудомісткості та вартості розробки програмного забезпечення.

Таблиця 3.1

Початкові дані для розрахунку трудомісткості та вартості розробки програмного забезпечення

	Параметр	Значення
1.	Передбачуване число операторів програми	988
2.	Коефіцієнт складності програми	1,4
3.	Коефіцієнт корекції програми в ході її розробки	0,08
4.	Годинна заробітна плата програміста	205,83 грн./год
5.	Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі	1,2
6.	Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності	0,8
7.	Вартість машино-години ЕОМ	1,97 грн./год

За даними вебпорталу DOU.UA на кінець грудня 2023 медіанна заробітна плата за місяць фахівців за категорією Junior Software Enginner складає 900 доларів США [29], що за поточним курсом НБУ у 40,25 грн./долар складає

36 225 грн. Звідси виходить, якщо стандартний робочий графік з 176 год/місяць, тоді заробітна плата за годину складає 205,83 грн./год.

Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі було прийнято 1,2.

Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності, при стажі до 2 років, складає 0,8.

Для розробки використовувався ноутбук, що споживає 45 Вт * год, тоді за тарифом, що діяв до кінці травня 2024 року, 2,64 грн./ кВт * год та за тарифом провайдера мобільного інтернету Lifecell «Вільний лайф» 325 грн./місяць або 1,85 грн/год виходить, що вартість машино-години ЕОМ складає 1,97 грн./год.

Трудомісткість програмного забезпечення для вибору інформативних ознак від результатів агромоніторингу на основі алгоритму Whale Optimisation розраховується за формулою (3.1).

$$t = t_o + t_{\text{и}} + t_{\text{а}} + t_{\text{п}} + t_{\text{отл}} + t_{\text{д}}, \text{ людино-годин}, \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм) міститься у формулі (3.2).

$$Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми,

p - коефіцієнт корекції програми в ході її розробки.

Отже, умовне число операторів складає 1480 оператор.

Витрати праці на вивчення опису задачі визначається з урахуванням уточнення опису і кваліфікації програміста наведені у формулі (3.3).

$$t_u = \frac{Q \cdot B}{85 \cdot k}, \text{ людино-годин}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k – коефіцієнт кваліфікації програміста.

Отже, витрати праці на вивчення опису задачі визначається з урахуванням уточнення опису і кваліфікації програміста складають 26,12 людино-годин.

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою (3.4).

$$t_a = \frac{Q}{25 \cdot k}, \text{ людино-годин} \quad (3.4)$$

де Q – умовне число операторів програми,

k – коефіцієнт кваліфікації програміста.

Отже, витрати праці на розробку алгоритму рішення задачі складають 74 людино-годин.

Витрати на складання програми по готовій блок-схемі визначаються за формулою (3.5).

$$t_n = \frac{Q}{20 \cdot k}\text{, людино-годин,} \qquad (3.5)$$

Отже, витрати на складання програми по готовій блок-схемі складають 92,5 людино-годин.

Витрати праці на налагодження програми на ЕОМ визначаються за умови автономного налагодження одного завдання (формула (3.6)) та за умови комплексного налагодження завдання (формула (3.7)).

$$t_{отл} = \frac{Q}{5 \cdot k}\text{, людино-годин,} \qquad (3.6)$$

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\text{, людино-годин,} \qquad (3.7)$$

Отже, витрати праці на налагодження програми на ЕОМ визначаються за умови автономного налагодження одного завдання складають 246 людино-годин, за умови комплексного налагодження завдання складають 370 людино-годин.

Витрати праці на підготовку документації визначаються за формулою (3.8).

$$t_д = t_{др} + t_{до}\text{, людино-годин,} \qquad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису, людино-годин,

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації, людино-годин.

Трудомісткість підготовки матеріалів і рукопису визначається за формулою (3.9).

$$t_{др} = \frac{Q}{15 \cdot k}\text{, людино-годин,} \quad (3.9)$$

Трудомісткість редагування, печатки й оформлення документації визначається за формулою (3.10).

$$t_{до} = 0{,}75 \cdot t_{др}\text{, людино-годин,} \quad (3.10)$$

Отже, трудомісткість підготовки матеріалів і рукопису складає 123,33 людино-годин, а трудомісткість редагування, печатки й оформлення документації складає 92,5 людино-годин. Звідси витрати праці на підготовку документації складають 215,83 людино-годин.

Згідно формули (3.1) трудомісткість програмного забезпечення для вибору інформативних ознак від результатів агромоніторингу на основі алгоритму Whale Optimisation складає 704,45 людино-годин.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$і витрат машинного часу $З_{МВ}$, необхідного на налагодження програми на ЕОМ. У формулі (3.11) наведено рівняння визначення витрат на створення ПЗ.

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{, грн.,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою (3.12).

$$З_{ЗП} = t \cdot С_{ПР}\text{, грн.,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

Оскільки медіанна годинна заробітна плата програміста складає 205,83 грн/год, то заробітна плата виконавців складає 144996,94 грн.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою (3.13).

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн.}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год,

$C_{мч}$ – вартість машино-години ЕОМ, грн./год.

Оскільки вартість машино-години ЕОМ складає 1,97 грн/год, то вартість машинного часу – 484,62 грн.

Отримані витрати на створення ПЗ складають 145481,56 грн.

Очікуваний період створення ПЗ наведено у формулі (3.14).

$$T = \frac{t}{B_k \cdot F_p}, \text{міс.}, \quad (3.14)$$

де B_k - число виконавців (приймаємо 1),

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Отримане значення очікуваного періоду створення ПЗ складає близько 4 місяців.

Висновки до третього розділу. Розроблювальний продукт має декілька показників, які характеризують його економічну значимість:

- передбачуване число операторів, яке складає 988;

- коефіцієнт складності програми, який складає 1,4;

- коефіцієнт корекції програми в ході її розробки, який складає 0,08;

- годинна заробітна плата програміста, яка складає 205,83 грн./год;

- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, який складає 1,2;

- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності, який складає 0,8;

- вартість машино-години ЕОМ, яка складає 1,97 грн./год.

Провівши ряд розрахунків було визначено наступні показники:

- трудомісткість ПЗ – 704,45 людино-годин;

- витрати на створення ПЗ – 145481,56 грн.;

- очікуваний період створення ПЗ – 4 місяців.

ВИСНОВКИ

Метою кваліфікаційної роботи була розробка програмного забезпечення для вибору інформативних ознак із результатів агромоніторингу на основі алгоритму Whale Optimisation, що здатне якісно оптимізувати з аналізу та моніторингу даних, що походять від різних систем, що фіксують зміни у навколишньому середовищі (найбільш популярні – IoT-системи).

Розроблене програмне забезпечення здатне отримувати дані про:

- назву дослідження;
- набір даних у вигляді файлу CSV;
- кількість китів;
- кількість ітерацій.

Ці дані передаються у вигляді multipart/form-data використовуючи HTTP-метод POST на вебсервер, який приймає ці дані, оброблює та видає як позитивний результат про успішне виконання вибірки та вкладені дані, так і негативний про виникнення помилки, у вигляді JSON-документу.

Програмне забезпечення представляє собою як вебзастосунок з однієї сторони, який містить вебінтерфейс та серверну частину, так і API, завдяки якому можна покращити власну програму, що може будувати різні моделі машинного навчання, що здатні в подальшому прогнозувати можливу врожайність на певний посівний період або визначити хворобу. Таким чином, ПЗ здатне використовуватись як звичайними користувачами, що мають на меті просто виокремити необхідні інформативні ознаки та визначити, що саме впливає на їхню цільову зміну, так і розробники, що хочуть покращити власний продукт, та заощадити обчислювальні ресурси на вибірку інформативних ознак.

Оскільки аграрна галузь є провідною в Україні і є постійний попит на програмні продукти, що могли покращити роботу агромоніторингу, то розробка програмного забезпечення була актуальною.

Дане програмне забезпечення має подальший розвиток, що може включати:

- додавання нових методів вибору інформативних ознак;

- створення CSV файли, що містять лише потрібні інформативні ознак;

- створення звіту у форматі .pdf;

- будування різних моделей машинного навчання;

- використання автентифікації та токенів для обмеження у певних ресурсах;

- використання документоподібних баз даних для зберігання даних досліджень, щоб вести історію;

- розширення вебінтерфейсу додаванням нових сторінок та головної сторінки для не авторизованих користувачів.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (704,45 людино-годин), підраховані витрати на створення програмного забезпечення (145481,56 грн.) і очікуваний період розробки (4 місяців).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). An Introduction to Statistical Learning. Springer. с. 204.

2. Створення програмних продуктів для сільського господарства | ELEKS. URL: https://eleks.com/ua/industries/software-development-agriculture/. Дата звернення 25.04.2024.

3. Поточний стан і перспективи розвитку агропромислового комплексу України. URL: http://ujae.org.ua/potochnyj-stan-i-perspektyvy-rozvytku-agropromyslovogo-kompleksu-ukrayiny/. Дата звернення 25.04.2024.

4. Палапа Н.В., Білотіл В.Ю., Гончар С.М., Бабікова К.О.. СІЛЬСЬКІ ТЕРИТОРІЇ УКРАЇНИ: СУЧАСНИЙ СТАН, ПРОБЛЕМИ, ШЛЯХИ РОЗВ'ЯЗАННЯ. Київ, 2023. С.57-65.

5. 4 тренди розвитку сільського господарства у 2024 році. URL: https://hub.kyivstar.ua/articles/4-trendi-agro-tech-yaki-budut-drajviti-silske-gospodarstvo-u-2024-roczi. Дата звернення 25.04.2024.

6. Шлях агро в 2024: агродрони, робототехніка, точне землеробство та діджиталізація". Підсумки Відкритої бесіди з DroneUA | Журнал "АгроЕліта. URL: https://agroelita.info/shliakh-ahro-v-2024-ahrodrony-robototekhnika-tochne-zemlerobstvo-ta-didzhytalizatsiia-pidsumky-vidkrytoi-besidy-z-droneua/. Дата звернення 25.04.2024.

7. Аутсорс-компанія ELEKS стала резидентом Дія.City | Міністерство цифрової трансформації. URL: https://thedigital.gov.ua/news/autsors-kompaniya-eleks-stala-rezidentom-diyacity. Дата звернення 25.04.2024.

8. Revolutionizing Agribusiness Through Digital Transformation. URL: https://miratechgroup.com/about/news/digital-transformation-agribusiness-miratech-kernel/. Дата звернення 25.04.2024.

9. Future agricultural systems and the role of digitalization for achieving sustainability goals. A review | Agronomy for Sustainable Development. URL:

https://link.springer.com/article/10.1007/s13593-022-00792-6. Дата звернення 25.04.2024.

10. Джерелюк К.І., Мазурець О.В. ДЕЯКІ АСПЕКТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ. м. Хмельницький. С.1-4. DOI: https://elar.khmnu.edu.ua/server/api/core/bitstreams/f0e13ee1-dbce-471d-a684-0e59fe31861c/content.

11. Merits and demerits of filter, wrapper and embedded feature selection... | Download Table. URL: https://www.researchgate.net/figure/Merits-and-demerits-of-filter-wrapper-and-embedded-feature-selection-methods_tbl1_313098800. Дата звернення 28.04.2024.

12. Whale Optimization Algorithm (WOA). URL: https://www.geeksforgeeks.org/whale-optimization-algorithm-woa/. Дата звернення 28.04.2024.

13. Sensors | Free Full-Text | The Whale Optimization Algorithm Approach for Deep Neural Networks. URL: https://www.mdpi.com/1424-8220/21/23/8003. Дата звернення 28.04.2024.

14. A Systematic Review of the Whale Optimization Algorithm: Theoretical Foundation, Improvements, and Hybridizations | Archives of Computational Methods in Engineering. URL: https://link.springer.com/article/10.1007/s11831-023-09928-7. Дата звернення 28.04.2024.

15. Eiben, A.E.; Smith, J.E. (2015). "What Is an Evolutionary Algorithm?". Introduction to Evolutionary Computing. Natural Computing Series. Berlin, Heidelberg: Springer. pp. 25–48.

16. Whale Optimization Algorithm | Baeldung on Computer Science. URL: https://www.baeldung.com/cs/whale-optimization-algorithm. Дата звернення 06.0.2024.

17. Клієнт-серверна архітектура. (qatestlab.com). URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/. Дата звернення: 06.06.2024.

18. Common web application architectures - .NET | Microsoft Learn. URL: https://learn.microsoft.com/en-gb/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures. Дата звернення: 06.06.2024.

19. SOA - individual services are independent of each other - What is SOA. URL: https://www.whatissoa.com/soa-individual-services-are-independent-of-each-other/. Дата звернення: 06.06.2024.

20. Django Project MVT Structure - GeeksforGeeks. URL: https://www.geeksforgeeks.org/django-project-mvt-structure/. Дата звернення: 06.06.2024.

21. Рейтинг мов програмування 2023. JavaScript/TypeScript завойовують світ, Python увійшов у топ-3, Salesforce Apex випередив 1C | DOU. URL: https://dou.ua/lenta/articles/language-rating-2023/. Дата звернення: 06.06.2024.

22. What’s New In Python 3.12 — Python 3.12.3 documentation. URL: https://docs.python.org/3/whatsnew/3.12.html. Дата звернення: 06.06.2024.

23. Package overview — pandas 2.2.2 documentation (pydata.org). URL: https://pandas.pydata.org/docs/getting_started/overview.html. Дата звернення: 06.06.2024.

24. scikit-learn: machine learning in Python — scikit-learn 1.5.0 documentation. URL: https://scikit-learn.org/stable/index.html. Дата звернення: 06.06.2024.

25. NumPy: the absolute basics for beginners — NumPy v1.26 Manual. URL: https://numpy.org/doc/stable/user/absolute_beginners.html. Дата звернення: 06.06.2024.

26. React vs Vanilla JavaScript: Pros, Cons, and the Perfect Fit for Your Project – Digitally Tailored. URL: https://digitallytailored.com/resources/react-vs-vanilla-javascript-pros-cons-and-the-perfect-fit-for-your-project/. Дата звернення: 06.06.2024.

27. React Reference Overview – React. URL: https://react.dev/reference/react. Дата звернення: 06.06.2024.

28. HTTP Multipart and Security Implications (f5.com). URL: https://community.f5.com/kb/technicalarticles/http-multipart-and-security-implications/316894. Дата звернення: 06.06.2024.

29. Статистика зарплат програмістів, тестувальників і PM в Україні | DOU. URL: https://jobs.dou.ua/salaries/. Дата звернення: 06.06.2024.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

./woa-api/woa_api/urls.py

```
from django.urls import path, include

# визначення схеми кінцевих точок для всього вебсерверу
urlpatterns = [
    path('api/', include('api_services.urls'))
]
```

./woa-api/api_services/urls.py

```
from django.urls import path

from .views import WoaSelectionView

# визначення схеми кінцевих точок для api/
urlpatterns = [
    path("woa-selection/", WoaSelectionView.as_view())
]
```

./woa-api/api_services/views.py

```
from django.views.generic.base import View
from django.http import JsonResponse
from django.utils.decorators import method_decorator
from django.views.decorators.csrf import csrf_exempt
from django.core.files.storage import FileSystemStorage
from .services import woa_selection_service

# декоратор для скасування csrf захисту
@method_decorator(csrf_exempt, name='dispatch')
# класове представлення для обробки запитів від /api/woa-selection
class WoaSelectionView(View):
    def post(self, request):
        # Перевірка на наявність файлу у запиті
        print(f"FILES={request.FILES}")

        if "csv_file" not in request.FILES:
            return JsonResponse({"status": "failure", "error": "Missed file."}, status=400)

        # Отримання значень з полів введення
```

```
    research_title = request.POST.get("research_title", "")
    whales_num = request.POST.get("whales_num", "")
    iter_num = request.POST.get("iter_num", "")

    # Перевірка поля research_title
    if not research_title or len(research_title) <= 5:
        return JsonResponse({"status": "failure",
                             "error": "No correct research_title."}, status=400)

    try:
        whales_num = int(whales_num)
        iter_num = int(iter_num)
        if whales_num < 5:
            return JsonResponse({"status": "failure",
                                 "error": "Whales number value must be equal or greater than 5."}, status=400)
        if iter_num < 10:
            return JsonResponse({"status": "failure",
                                 "error": "Iter number value must be equal or greater than 10."}, status=400)
    except ValueError:
        return JsonResponse({"status": "failure",
                             "error": "Not correct number values."}, status=400)
    try:
        # Збереження файлу
        file = request.FILES['csv_file']
        fs = FileSystemStorage()
        filename = fs.save(file.name, file)
        uploaded_file_url = fs.path(filename)

        response_data = woa_selection_service(uploaded_file_url, whales_num, iter_num)
    except Exception as e:
        return JsonResponse({"status": "failure",
                             "error": str(e)}, status=400)

    return JsonResponse({"status": "success", "data": {"research_title": research_title, **response_data}}, status=201)
```

./woa-api/api_services/services.py

```
import pandas as pd
import numpy as np
from mealpy import FloatVar
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from mealpy.swarm_based.WOA import OriginalWOA


def woa_selection_service(data_csv_url, whales_num=50, iters_num=30):
    # Завантаження даних
    data = pd.read_csv(data_csv_url)

    # Розділення даних на ознаки та залежну змінну
    X = data.iloc[:, :-1]
    y = data.iloc[:, -1]

    # Вибір числових стовпців
    numeric_columns = X.select_dtypes(include=[np.number]).columns

    # Ініціалізація DataFrame для імпутованих даних
    data_imputed = pd.DataFrame()

    # Заміна пропущених значень на середнє
    imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
    data_imputed[numeric_columns] = pd.DataFrame(imputer.fit_transform(data[numeric_columns]),
columns=numeric_columns)

    # Вибір стовпців з категоріями
    categorical_columns = X.select_dtypes(include=['object']).columns

    # Додавання категоріальних стовпців назад до датасету без змін
    data_imputed[categorical_columns] = X[categorical_columns]

    # Набір ознак для відповіді
    X_response = data_imputed

    # Кодування категоріальних даних
    categorical_data = data_imputed[categorical_columns]
    one_hot_encoder = OneHotEncoder()
    categorical_encoded = one_hot_encoder.fit_transform(categorical_data).toarray()

    # Видалення оригінальних категоріальних стовпців
    data_imputed.drop(categorical_columns, axis=1, inplace=True)
```

```
# Додавання закодованих категоріальних даних до датасету
data_imputed = np.hstack((data_imputed.values, categorical_encoded))

print(f"data_imputed={data_imputed}")

# Кодування залежну зміну

if y.dtype == 'object':
    label_encoder = LabelEncoder()
    y_encoded = label_encoder.fit_transform(y)
else:
    y_encoded = y.values

print(f"y_encoded={y_encoded}")

# Підготовка даних для вибору ознак за допомогою WOA
X_prepared = data_imputed
y_prepared = y_encoded

# Функція пристосованості
def objective_function(solution):
    # Обмеження рішення до 0 або 1
    binary_solution = np.where(solution > 0.5, 1, 0)

    # Перевірка чи є хоча б одна ознака вибрана
    if np.sum(binary_solution) == 0:
        # Повертаємо нескінченність, якщо не вибрано жодної ознаки
        return float('inf')

    # Розділення даних на навчальні та валідаційні набори
    X_train, X_val, y_train, y_val = train_test_split(
        X_prepared[:, binary_solution == 1], y_prepared, test_size=0.2, random_state=0
    )

    # Створення та навчання моделі Random Forest Regressor
    model = RandomForestRegressor(n_estimators=100, random_state=0)
    model.fit(X_train, y_train)

    # Оцінка моделі на валідаційних даних
    predictions = model.predict(X_val)
    mse = mean_squared_error(y_val, predictions)
```

```
        return mse

    # Ініціалізація WOA
    model = OriginalWOA(epoch=iters_num, pop_size=whales_num)

    # Створення словника з параметрами проблеми
    problem_dict = {
        "bounds": FloatVar(lb=(-1,) * X_prepared.shape[1], ub=(1,) * X_prepared.shape[1], name="agro_data"),
        "minmax": "min",
        "obj_func": objective_function
    }

    # Запуск оптимізації
    g_best = model.solve(problem_dict)

    best_solution = g_best.solution
    best_fitness = g_best.target.fitness

    # Визначення індексів вибраних ознак
    selected_features = np.where(best_solution > 0.5)[0]

    # Визначення стовпців вибраних ознак
    selected_features_columns = data.columns[selected_features]

    # Виведення результату
    response = {
        "best_solution": best_solution.tolist(),
        "best_fitness": best_fitness,
        "selected_features_columns": selected_features_columns.tolist(),
        "X": X_response[selected_features_columns].to_dict(orient='records'),
        "y": pd.DataFrame(y).to_dict(orient='records')
    }

    return response
```

./service-web-page/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/icon.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link

href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,
400;1,500;1,700;1,900&family=Signika:wght@300..700&display=swap"
      rel="stylesheet"
    />
    <title>WOA API Service</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

./service-web-page/src/main.jsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

./service-web-page/src/index.css

```
*,
*::before,
*::after {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

html {
  box-sizing: border-box;
  font-size: 62.5%;
}
```

```
body {
  font-family: "Roboto", sans-serif;
  font-weight: 300;
}

#root {
  margin: 0 auto;
  width: 80vw;
  min-height: 100vh;
  display: grid;

  grid-template-columns: repeat(12, 1fr);
  grid-template-rows: 15rem 1fr;
  gap: 3rem;

  --dark-color: #143461;
  --middle-color: #05c9c0;
  --light-color: #1bf5d2;
  --extra-light-color: #7cfae6;
}

@media screen and (max-width: 818px) {
  html {
    font-size: 50% !important;
  }
}

@media screen and (max-width: 650px) {
  #root {
    width: 90vw;
  }
}

@media screen and (max-width: 416px) {
  #root {
    width: 95vw;
  }
}

@media screen and (max-width: 388px) {
  html {
    font-size: 37.5% !important;
```

```
  }
}

@media screen and (min-width: 1605px) {
  #root {
    width: 75vw;
  }
}

@media screen and (min-width: 1540px) {
  html {
    font-size: 75% !important;
  }
}
```

./service-web-page/src/App.jsx

```
import {
  Navigate,
  RouterProvider,
  createBrowserRouter,
} from "react-router-dom";
import MainPage from "./pages/Main/Main";
import RootLayout from "./pages/Root";
import { WOAServiceAction } from "./pages/WOAService";
import ErrorPage from "./pages/Error/Error";

function App() {
  // визначення маршрутів вебінтерфейсу
  const router = createBrowserRouter([
    {
      path: "/",
      id: "root",
      element: <RootLayout />,
      errorElement: <ErrorPage />,
      children: [
        {
          index: true,
          element: <Navigate to="/main" />,
        },
        {
          path: "main",
          element: <MainPage />,
```

```
      },
    ],
  },
  // визначення шляху, що виконує action для відправки даних на сервер
  {
    path: "/api/woa-selection",
    element: <p>No content</p>,
    action: WOAServiceAction,
  },
]);

 return <RouterProvider router={router} />;
}

export default App;
```

./service-web-page/src/pages/Root.jsx

```
import { Outlet } from "react-router-dom";
import Header from "../components/Header/Header";
import MainWrapper from "../components/MainWrapper/MainWrapper";
import Footer from "../components/Footer/Footer";

export default function RootLayout() {
 return (
  <>
    <Header />
    <MainWrapper>
      <Outlet />
    </MainWrapper>
    <Footer />
  </>
 );
}
```

./service-web-page/src/pages/WOAService.jsx

```
import { json } from "react-router-dom";

export async function WOAServiceAction({ request }) {
 // отримання даних від форми
 let formData = await request.formData();

 try {
```

```
    // виконання POST запиту та отримання відповіді
    const response = await fetch("http://localhost:8000/api/woa-selection/", {
      method: "POST",
      body: formData,
    });

    // отримання даних від відповіді
    const data = await response.json();

    if (response.ok) {
      return json(data, { status: 201 });
    }

    return json(data, { status: 200 });
  } catch (error) {
    // якщо трапилась помилка - сервер не відповів, то повертаємо повідомлення про помилку на сторінку
    return json(
      { status: "failure", error: "Server is not responding" },
      { status: 200 }
    );
  }
}
```

./service-web-page/src/pages/Main/Main.jsx

```
import { Form, useFetcher } from "react-router-dom";
import FileUploader from "../../components/FileUploader/FileUploader";
import Input from "../../components/Input/Input";
import mainClasses from "./Main.module.css";
import { useState } from "react";
import Loader from "../../components/Loader/Loader";
import ErrorMessage from "../../components/ErrorMessage/ErrorMessage";

export default function MainPage() {
  const [file, setFile] = useState(null);
  //  використання fetcher дозволяє виконувати action без перходу на сторінку, а також завантажува від нього відповідь
  const fetcher = useFetcher();

  const isSubmitting = fetcher.state === "submitting";

  // функція для очищення форми
  function handleReset(event) {
```

```
  event.preventDefault();

  for (const input of event.target) {
    if (input.value !== "") {
      input.value = "";
    }

    setFile(null);
  }
}

// функція для надсиалння даних

function handleSubmit(event) {
  event.preventDefault();

  // отримує дані форми від таргету на якій виконалась подія
  const formData = new FormData(event.target);

  // імперативно надсилаємо дані на action від WOAService
  fetcher.submit(formData, {
    action: "/api/woa-selection",
    method: "POST",
    encType: "multipart/form-data",
  });

  // очищаємо дані форми
  handleReset(event);
}

return (
  <>
    {/* якщо дані надсилаються, з'являється лоадер */}
    {isSubmitting && <Loader />}

    {/* розділ "Про API" */}
    <section id="about-api" className={mainClasses["about-api-section"]}>
      <h2
        className={`${mainClasses["secondary-heading"]} ${mainClasses["light-heading"]}`}
      >
        Про API
      </h2>
```

```
<p className={mainClasses["main-text"]}>
  Даний сервіс використовується для відбору інформативних ознак з даних
  отриманих від користувача. Користувач надсилає дані у форматі CSV
  (Comma-Separated Value) та отримує звіт у вебформаті з необхідними
  даними для аналізу і ознаками, що впливають на певний набір.
</p>
<p className={mainClasses["api-text"]}>
  <span>Даний API використовує домен</span>{" "}
  <code>http://localhost:8000/api</code>
</p>
<article className={mainClasses["api-item"]}>
  <code className={mainClasses["endpoint"]}>/woa-selection</code>
  <code className={mainClasses["http-method"]}>/POST</code>
  <p className={mainClasses["description"]}>
    Цей ендпоінт приймає наступні дані як{" "}
    <span className={mainClasses["text-highlighter"]}>
      research_title
    </span>{" "}
    &amp;#8213; назва дослідження, яке мало на меті створити дані,{" "}
    <span className={mainClasses["text-highlighter"]}>file_csv</span>{" "}
    &amp;#8213; файл, що містить дослідження дані у csv-форматі з ознаками
    та залежною змінною,{" "}
    <span className={mainClasses["text-highlighter"]}>whales_num</span>{" "}
    &amp;#8213; кількість китів, яку необхідно задіяти для вибірки ознак,{" "}
    <span className={mainClasses["text-highlighter"]}>iter_num</span>{" "}
    &amp;#8213; кількість ітерацій, за яких відбувається
    &amp;quot;полювання&amp;quot; на ознаки. Як відповідь повертає JSON-форматі.
    Якщо відбір ознак минув вдало, то повертається
    <pre className={mainClasses["success-json"]}>
      {`
      {
        "status": "success",
        "data": {
          "research_title": "...",
          "best_solution": [...],
          "best_fitness: 123.456,
          "selected_features_columns": [...],
          "X": [
            {
              "feature_0": 123,
              ...
            },
```

```
            ...
          ],
          "y": [
            {
              "result": 321,
            },
            ...
          ],
        }
      }
      `}
    </pre>
    Якщо під час відбору трапилась помилка, то повертається формат
    <pre className={mainClasses["failure-json"]}>
      {`
      {
        "status": "failure",
        "error": "..."
      }
      `}
    </pre>
  </p>
 </article>
</section>
{/* розділ "Відбір ознак" */}
<section
 id="feature-selection"
 className={mainClasses["feature-selection-section"]}
>
 <h2
  className={`${mainClasses["secondary-heading"]} ${mainClasses["dark-heading"]}`}
 >
  Відбір ознак
 </h2>

 {!isSubmitting && fetcher.data?.status === "failure" && (
  <ErrorMessage message={fetcher.data?.error} />
 )}
 <fetcher.Form
  className={mainClasses["feature-selection-form"]}
  onReset={handleReset}
  onSubmit={handleSubmit}
```

```
>
  <Input
    type="text"
    name="research_title"
    id="research_title"
    label="Назва дослідження"
  />

  <FileUploader
    type="file"
    name="csv_file"
    id="csv_file"
    accept=".csv"
    label="Файл з ознаками та залежною змінною"
    file={file}
    setFile={setFile}
  />

  <Input
    type="number"
    name="whales_num"
    id="whales_num"
    label="Кількість китів"
    min="5"
    max="10000"
  />

  <Input
    type="number"
    name="iter_num"
    id="iter_num"
    label="Кількість ітерацій"
    min="10"
    max="10000"
  />

  <div className={mainClasses["btn-container"]}>
    {/* якщо дані завантажуються, то кнопки блокуються */}
    <button type="reset" disabled={isSubmitting}>
      Очистити
    </button>
    <button type="submit" disabled={isSubmitting}>
```

```
        Надіслати
      </button>
    </div>
  </fetcher.Form>
</section>

{/* розділ "Результат" */}
<section id="result" className={mainClasses["result-section"]}>
  <h2
    className={`${mainClasses["secondary-heading"]} ${mainClasses["middle-color-heading"]}`}
  >
    Результат
  </h2>

  {/* якщо дані від сервера існують, то з'являється звіт */}
  {fetcher.data?.data && (
    <article className={mainClasses["report"]}>
      <h3 className={mainClasses["report-heading"]}>
        {fetcher.data?.data["research_title"]}
      </h3>
      <p className={mainClasses["report-param"]}>Найкраще рішення </p>
      <code className={mainClasses["report-value"]}>
        <span className={mainClasses["brackets"]}>[</span>
        <ul>
          {fetcher.data?.data["best_solution"].map((bs, i) => {
            return <li key={i}>{bs}</li>;
          })}
        </ul>
        <span className={mainClasses["brackets"]}>]</span>
      </code>
      <p className={mainClasses["report-param"]}>
        Найкраща допасованість
      </p>
      <code
        className={`${mainClasses["report-value"]} ${mainClasses["fitness"]}`}
      >
        {fetcher.data?.data["best_fitness"]}
      </code>
      <p className={mainClasses["report-param"]}>Обрані ознаки </p>
      <code className={mainClasses["report-value"]}>
        <ul className={mainClasses["selected-features"]}>
          {fetcher.data?.data["selected_features_columns"].map((f, i) => {
```

```
      return <li key={i}>{f}</li>;
    })}
  </ul>
</code>
<p className={mainClasses["report-param"]}>
  Кількість необхідних ознак
</p>
<code
  className={`${mainClasses["report-value"]} ${mainClasses["selected-features-num"]}`}
>
  {fetcher.data?.data["selected_features_columns"].length}
</code>
<div className={mainClasses["report-table-wrapper"]}>
  <table className={mainClasses["report-table"]}>
    <thead>
      <tr>
        <th></th>
        {Object.keys(fetcher.data?.data?.X[0]).map((key) => {
          return <th key={key}>{key}</th>;
        })}
        <th>{Object.keys(fetcher.data?.data?.y[0])[0]}</th>
      </tr>
    </thead>
    <tbody>
      {fetcher.data?.data.X.map((row, rowIndex) => (
        <tr key={rowIndex}>
          <td>{rowIndex + 1}.</td>
          {Object.keys(fetcher.data?.data?.X[0]).map(
            (feature, colIndex) => (
              <td key={colIndex}>{row[feature]}</td>
            )
          )}
          <td>
            {fetcher.data?.data?.y[rowIndex] &&
            Object.keys(fetcher.data?.data?.y[0])[0] in
              // eslint-disable-next-line no-unsafe-optional-chaining
              fetcher.data?.data?.y[rowIndex]
              ? fetcher.data?.data?.y[rowIndex][
                  Object.keys(fetcher.data?.data?.y[0])[0]
                ]
              : ""}
          </td>
```

```
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </article>
    )}

    {/* якщо даних від сервера немає, то з'являється повідомлення "Поки що дані відстутні." */}
    {fetcher.data?.status !== "success" && <p>Поки що дані відстутні.</p>}
  </section>
 </>
 );
}
```

./service-web-page/src/pages/Main/Main/module/css

```
/* ====================== */
/* =========ABOUT API SECTION==================== */
/* ========================== */

.about-api-section {
  display: flex;
  flex-direction: column;
  gap: 3rem;
  color: var(--dark-color);
  background-color: var(--dark-color);
  border-radius: 9px;
  font-size: 1.4rem;

  padding: 6rem 3rem 4rem 3rem;
  position: relative;
}

.api-item {
  display: flex;
  flex-wrap: wrap;
  gap: 2rem;
  align-items: first baseline;
}

.secondary-heading {
  color: white;
```

```
  position: absolute;

  left: 2rem;
  top: 2rem;
  padding: 0.3rem 1.2rem;
  border-radius: 9px;
  font-size: 1.2rem;

  font-weight: 700;
}

.light-heading {
  background-color: rgba(124, 250, 230, 0.5);
}

.middle-color-heading {
  background-color: rgba(5, 201, 192, 0.5);
}

.dark-heading {
  background-color: rgba(20, 52, 97, 0.5);
}

.main-text,
.api-text {
  font-weight: 500;
  line-height: 1.4;
  color: #fff;
}

.api-text {
  text-align: center;
}

.api-text > code {
  background-color: rgba(124, 250, 230, 0.5);
  color: var(--dark-color);
  padding: 0.5rem;
  border-radius: 9px;
  border: 1px solid var(--extra-light-color);
}
```

```
.endpoint {
  flex: 1;
  font-weight: 400;
  background-color: var(--extra-light-color);
  padding: 0.5rem;
  border-radius: 3px;
}

.http-method {
  flex: 1;
  font-weight: 600;
  background-color: var(--light-color);
  padding: 0.5rem;
  border-radius: 3px;
}

.description {
  flex: 8;
  color: white;

  padding: 0.5rem;
  border-radius: 3px;
  font-weight: 200;
  line-height: 1.6;
}

.description > pre {
  line-height: 1.1;
  border-radius: 9px;
  background-color: #ffffffb9;
}

.text-highlighter {
  color: var(--middle-color);
  font-weight: 500;
}

.success-json {
  color: rgb(2, 165, 2);
}

.failure-json {
```

```
  color: red;
}

@media screen and (max-width: 424px) {
  .api-text {
    display: flex;
    flex-direction: column;
    gap: 1.2rem;
  }
}

/* ============ */
/* =========FEATURE SELECTION SECTION========== */
/* ========================== */

.feature-selection-section {
  display: flex;

  align-items: center;
  justify-content: center;
  flex-direction: column;
  gap: 5rem;

  padding: 6rem 3rem 4rem 3rem;
  position: relative;
}

.feature-selection-form {
  display: flex;
  flex-direction: column;
  gap: 1.8rem;
  padding: 3rem;
  border: 1px solid var(--dark-color);
  border-radius: 9px;
  width: 50rem;
}

.btn-container {
  display: flex;
  justify-content: space-between;
  padding-top: 1.8rem;
}
```

```
.btn-container > button {
  display: inline-block;
  border: 1.5px solid transparent;
  border-radius: 3px;
  cursor: pointer;
  padding: 0.5rem 1rem;

  font-size: 1.6rem;
  transform-origin: bottom;

  transition: all 0.3s;
}

.btn-container > button[type="reset"] {
  background-color: transparent;
  color: var(--dark-color);
}

.btn-container > button[type="submit"] {
  background-color: var(--middle-color);
  color: #fff;
}

.btn-container > button[type="reset"]:hover {
  border-bottom-color: var(--dark-color);
}

.btn-container > button[type="submit"]:hover {
  background-color: var(--dark-color);
}

.btn-container > button[type="reset"]:active {
  color: #fff;
  background-color: black;
}

@media screen and (max-width: 416px) {
  .feature-selection-form {
    gap: 1.6rem;
    padding: 2rem;
```

```
    width: 40rem;
  }
}

/* =============== */
/* ================RESULT SECTION====================== */
/* ============================== */

.result-section {
  display: flex;

  align-items: center;
  justify-content: center;

  padding: 6rem 3rem 4rem 3rem;
  position: relative;
}

.report {
  display: grid;

  grid-template-columns: repeat(2, 1fr);
  justify-content: center;
  align-content: center;
  align-items: center;
  column-gap: 1rem;
  row-gap: 2rem;

  width: 80rem;
}

.report-heading {
  grid-area: 1 / span 2;
  text-align: center;

  color: var(--dark-color);
  font-size: 2.4rem;
  margin-bottom: 1.4rem;
}

.report-param {
  font-size: 1.6rem;
```

```
  font-weight: 400;
}

.report-value {
  font-size: 1.6rem;
}

.brackets {
  font-size: 2.4rem;
  font-weight: 700;
}

.brackets + ul {
  list-style: none;
  display: flex;
  flex-direction: column;
  color: var(--middle-color);
}

.fitness {
  display: inline;
  background: var(--extra-light-color);
  color: var(--dark-color);
  border-radius: 3px;
  padding: 0.5rem;
  width: fit-content;

  font-size: 1.8rem;
  font-weight: 700;
}

.selected-features {
  list-style: none;
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
}

.selected-features > li {
  background: var(--dark-color);
  color: #fff;
  padding: 1rem;
```

```
  width: fit-content;

  border-radius: 9px;
}

.selected-features > li:nth-child(3n + 2) {
  background: var(--middle-color);
}

.selected-features > li:nth-child(3n + 3) {
  background: var(--light-color);
}

.selected-features-num {
  display: inline-block;
  width: fit-content;

  padding: 1rem 2rem;
  border-radius: 9px;
  font-size: 1.8rem;
  font-weight: 700;
  color: #fff;

  background-color: rgb(2, 143, 44);
}

.report-table-wrapper {
  grid-area: 7 / 1 / span 1 / span 2;
  overflow-x: auto;
}

.report-table {
  border: 3px solid var(--dark-color);
  color: var(--dark-color);
  font-size: 1.6rem;
  min-width: 100%;

  border-collapse: collapse;
  margin-top: 2rem;
}
```

```
.report-table > thead {
  border: 1px solid var(--dark-color);
  color: #fff;
  background-color: var(--dark-color);
}

.report-table th,
.report-table td {
  padding: 0.5rem;
}

.report-table > tbody > tr:hover {
  color: #fff;
  background: var(--light-color);

  cursor: pointer;
}

.report-table > tbody > tr > td:last-child {
  color: #fff;
  background: var(--light-color);
}

.report-table > tbody > tr > td {
  border: 1px solid var(--light-color);
}

@media screen and (max-width: 424px) {
  .selected-features > li {
    font-size: 1.4rem;
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Малишко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Малишко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
WoaApiService.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Малишко.ppt	Презентація кваліфікаційної роботи