МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувача кафедри інтелектуальних інформаційних систем, д-р.техн.наук, проф.
_____ Ю. П. Кондратенко
«____»_____2024 року

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПОДІЛУ ГУМАНІТАРНОЇ ДОПОМОГИ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21810123

Виконав студент 6-го курсу, групи 601
_____ А.О. Стратонов
«19» лютого 2024 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«19» лютого 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р.техн.наук, проф.
_____ Ю. П. Кондратенко
«___» _____20__ р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу

Стратонову Артему Олександровичу

1. Тема магістерської кваліфікаційної роботи «Інтелектуальна система розподілу гуманітарної допомоги».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «19» жовтня 2023 р. № 201__

2. Строк подання студентом роботи 21 лютого 2024 р.

3. Вхідні (початкові) дані до роботи: загальні відомості про підходи до розподілу гуманітарних ресурсів, набір даних стосовно осіб, які потребують гуманітарної допомоги.

Очікуваний результат роботи: система розподілу гуманітарної допомоги з вбудованими алгоритмами інтелектуального аналізу розподілу ресурсів відповідно до виявлених потреб.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- дослідження теоретичних засад розроділу гуманітарної допомоги та аналіз мережевих ресурсів у цій сфері;

- обґрунтування вибору технологій та інструментальних засобів розробки інтелектуальної системи;
- розробка та здійснення програмної реалізації системи розподілу гуманітарної допомоги з використанням алгоритмів кластерного аналізу.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Оцінка ризиків пов'язаних з роботою на ПК та заходи їх зниження.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	докт.біол.н.,професор Л. І. Григор'єва	
Методична частина	канд.пед.н., доцент Н.М. Болюбаш	

Керівник роботи __канд. пед. наук, доц. Болюбаш Н. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Стратонов А.О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_____» _____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН

виконання магістерської кваліфікаційної роботи

Тема: «Інтелектуальна система розподілу гуманітарної допомоги»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Визначення керівника і теми КРМ. Подання заяви на затвердження теми КРМ	01.09.2023	19.10.2023	Виконано
2.	Отримання завдання на виконання КРМ	20.10.2023	29.10.2023	Виконано
3.	Складання календарного плану	30.10.2023	5.11.2023	Виконано
4.	Огляд літератури за темою дослідження. Аналіз існуючих підходів до виявлення тенденцій у сфері розподілу гуманітарної допомоги, методів кластеризації	5.11.2023	25.11.2023	Виконано
5.	Проходження переддипломної практики, збір та аналіз матеріалів до КРМ	27.112023	24.12.2023	Виконано
6.	Аналіз предметної області та розробка технічного завдання	25.12.2023	27.12.2023	Виконано
7.	Проєктування та програмна реалізація інтелектуальної системи розподілу гуманітарної допомоги	28.12.2023	15.01.2024	Виконано
8.	Робота над розділами фахової частини КРМ	16.01.2024	24.12.2024	Виконано
9.	Розробка методичної частини КРМ та спеціальної частини з охорони праці	25.01.2024	01.02.2024	Виконано
10.	Обговорення отриманих результатів з керівником та попередній захист КРМ	29.01.2024	3.02.2024	Виконано
11.	Корегування роботи за результатами попереднього захисту	4.02.2024	6.02.2024	Виконано
12.	Остаточне оформлення пояснювальної записки та слайдів доповіді до зихисту	7.02.2024	9.02.2024	Виконано
13.	Подання рецензенту та рецензування КРМ	9.02.2024	12.02.2024	Виконано
14.	Подання КРМ, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2024	16.02.2024	Виконано
15.	Захист КРМ перед ЕК	27.02.2024	27.02.2024	Виконано

Розробив студент _____ Стратонов А.О. _____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи __канд.пед.н., доц. Болюбаш Н.М. _____ _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра
студента групи 601 ЧНУ ім. Петра Могили

Стратонова Артема Олександровича

на тему: «**ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПОДІЛУ ГУМАНІТАРНОЇ ДОПОМОГИ**»

Магістерська кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації системи розподілу гуманітарної допомоги із використанням методів кластерного аналізу даних. Що є актуальним в умовах ведення воєнних дій на території країни, оскільки полегшує взаємодію волонтерів із особами, які потребують допомоги, та забезпечує оптимальний розподіл ресурсів.

Об'єкт дослідження – процес розподілу ресурсів у сфері гуманітарної допомоги.

Предмет дослідження – програмні засоби та методи розроділу ресурсів гуманітарної допомоги на основі інтелектуального аналізу даних.

Мета дослідження – підвищення ефективності розподілу гуманітарної допомоги шляхом розробки вебзастосунку із використанням кластерного аналізу даних.

Магістерська кваліфікаційна робота складається з фахової, методичної і спеціальної частини з охорони праці. Пояснювальна записка фахової частини кваліфікаційної роботи складається зі вступу, трьох розділів, висновків та додатків. У першому розділі розкрито теоретичні засади розподілу гуманітарної допомоги та здійснити аналіз мережевих ресурсів у цій сфері. У другому розділі обґрунтовано вибір технологій та інструментальних засобів розробки системи розподілу гуманітарної допомоги. У третьому розділі описано розробку та програмну реалізацію системи розподілу гуманітарної допомоги з виявленням тенденцій розподілу на основі методів кластеризації та класифікації даних.

Магістерська кваліфікаційна робота містить ___ сторінку (без додатків), ___ рисунків, ___ таблиці, ___ джерел, ___ додатки.

Ключові слова: гуманітарна допомога, кластерний аналіз даних, алгоритм k-means, алгоритм c-means, ієрархічний алгоритм.

ABSTRACT

to the master's qualification work
by the student of the group 601 of Petro Mohyla Black Sea National University

StratonovA ArtemA OleksandrovychA

on the subject: «**AN INTELLECTUAL SYSTEM OF HUMANITARIAN AID DISTRIBUTION**»

The master's thesis is devoted to the development and implementation of the software implementation of the distribution system of humanitarian aid using methods of cluster data analysis. Which is relevant in the conditions of conducting military operations on the territory of the country, as it facilitates the interaction of volunteers with persons in need of help and ensures the optimal distribution of resources.

Object of research – the process of resource allocation in the field of humanitarian aid.

Subject of research – software tools and methods for the distribution of humanitarian aid resources based on Data Mining.

The purpose of the study is to increasing the efficiency of distribution of humanitarian aid by developing a web application using cluster data analysis.

The master's qualification work consists of a professional, methodical and special part on labor protection. The explanatory note of the qualification work consists of an introduction, three sections, conclusions and appendices. In the first chapter, the theoretical foundations of the distribution of humanitarian aid are disclosed and an analysis of network resources in this area is carried out. In the second chapter, the choice of technologies and tools for the development of the humanitarian aid distribution system is substantiated. The third chapter describes the development and software implementation of the humanitarian aid distribution system with the identification of distribution trends based on methods of cluster analysis and data classification.

Thesis contains ___ page (without appendices), ___ figures, ___ tables, ___ sources, ___ appendics.

Key words: humanitarian aid, cluster data analysis, k-means algorithm, c-means algorithm, hierarchical algorithm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

ШІ – штучний інтелект

CSS – Cascading Style Sheets

HDX – Humanitarian Data Exchange

HTML – Hyper Text Markup Language

DHNetwork – Digital Humanitarian Network

ВСТУП

Актуальність. Бурхливі темпи інформатизації сучасного суспільства супроводжуються впровадженням цифрових технологій в усі сфери оточуючої дійсності та у сферу обліку і розподілу гуманітарної допомоги зокрема. Обсяги гуманітарної допомоги різко зросли в умовах ведення воєнних дій на території країни й охоплюють велику кількість волонтерів, громадських вітчизняних та міжнародних організацій і благодійних фондів. Різко зросла потреба оперативного надходження допомоги тим, хто її потребує. Вирішення цієї проблеми обумовлює необхідність створення інформаційних систем для покращення прогнозування попиту та керування ланцюгами постачання і розподілу гуманітарної допомоги.

Гуманітарна допомога надається під час надзвичайних ситуацій і криз та спрямована на забезпечення порятунку життя людей і задоволення їх основних потреб у воді, їжі, проживанні, гігієні та медичній допомозі. На рівні держави на початок 2024 року в Україні розроблено веб-платформу Aidmonitor.org яка почала впроваджуватися для контроля за наданням та розподілом гуманітарної допомоги міжнародними донорськими організаціями. Однак є проблема у налагодженні узгодженої взаємодії волонтерів із особами, які проживають безпосередньо у прифронтових територіях та оптимальному механізмі розподілу допомоги. Що обумовлює необхідність у розробці системи аналізу даних стосовно осіб, які потребують гуманітарної допомоги та забезпечує необхідний до виявлених тенденцій розподіл наявних ресурсів.

Мета дослідження – підвищення ефективності розподілу гуманітарної допомоги шляхом розробки вебзастосунку із використанням кластерного аналізу даних.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

− дослідити теоретичні засади розподілу гуманітарної допомоги та здійснити аналіз мережевих ресурсів у цій сфері;

– обґрунтувати вибір технологій та інструментальних засобів розробки інтелектуальної системи;

– розробити та здійснити програмну реалізацію системи розподілу гуманітарної допомоги з використанням алгоритмів кластерного аналізу.

Об’єктом дослідження є процес розподілу ресурсів у сфері гуманітарної допомоги.

Предметом дослідження є програмні засоби та методи розроділу ресурсів гуманітарної допомоги на основі інтелектуального аналізу даних.

Методологічною основою дослідження є загальнонаукові аналітичні методи, методи кластерного аналізу даних, які дозволили вивчити предмет та об’єкт дослідження, дослідити розвиток теоретичних засад, напрямів та шляхів підвищення ефективності розподілу ресурсів гуманітарної допомоги шляхом розробки системи для виявлення тенденцій розподілу із використанням кластерного аналізу даних.

Наукова новизна одержаних результатів дослідження полягає у тому, що автором: запропоновано та обґрунтовано напрями вдосконалення розподілу гуманітарної допомоги серед осіб, які її потребують; одержали подальший розвиток підходи до підвищення ефективності у виявлені тенденцій розподілу ресурсів; узагальнено теоретичні засади виявлення тенденцій розподілу на основі методів кластерного аналізу та класифікації даних.

Результати дослідження обговорювалися на Всеукраїнської Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія» (29 січня – 2 лютого 2023 року) та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що розроблену систему можна застосувати для налагодження узгодженої взаємодії волонтерів із особами, які проживають безпосередньо у прифронтових територіях та оптимального механізму розподілу допомоги.

Структура магістерської роботи. Відповідно до мети, завдань і предмета дослідження магістерська робота містить основну, методичну та спеціальну частини. Основна частина магістерської роботи складається із вступу, чотирьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг магістерської роботи – __ сторінок, із них тексту основної частини – __ сторінок, методичної частини – __ сторінок, спеціальної – __ сторінок. Кількість використаних джерел – __.

1 ТЕОРЕТИЧНІ ЗАСАДИ РОЗПОДІЛУ ГУМАНІТАРНОЇ ДОПОМОГИ

1.1 Традиційні методи розподілу ресурсів

Гуманітарна допомога надається під час надзвичайних ситуацій і криз та спрямована на забезпечення порятунку життя людей і задоволення їх основних потреб у воді, їжі, проживанні, гігієні та медичній допомозі. Здійснимо аналіз традиційних методів розподілу ресурсів.

Розподіл ресурсів – це процес визначення та розподілу доступних ресурсів між різними суб'єктами, групами або областями для задоволення потреб чи досягнення певних цілей [1]. Ресурси можуть включати матеріальні блага (такі як їжа, вода, енергія, сировина), фінансові ресурси, робочу силу, технічні засоби, територію та інші складові, які використовуються для різноманітних діяльностей. Процес розподілу ресурсів може відбуватися за різними принципами, методами та механізмами, залежно від економічної системи, політичного устрою чи соціокультурного контексту. Наприклад, в ринковій економіці розподіл може відбуватися через взаємодію попиту та пропозиції на ринку, у плановій економіці - шляхом централізованого планування та регулювання з боку держави.

Мета розподілу ресурсів – забезпечити ефективне використання ресурсів та задоволення потреб суспільства, уникнути надмірної концентрації чи нерівності у розподілі, а також забезпечити сталий розвиток та екологічну безпеку.

Традиційні методи розподілення ресурсів в різних галузях можуть включати різні стратегії та підходи. Ось кілька загальних традиційних методів, які можуть використовуватися у різних областях:

– рівномірний розподіл: ресурси розподіляються рівномірно між всіма суб'єктами або областями без врахування індивідуальних потреб чи пріоритетів. Простий, але не завжди ефективний, особливо в умовах різних рівнів потреб;

– заснований на потребах підхід: розподіл ресурсів враховує індивідуальні потреби чи вимоги суб'єктів або областей. Спрямований на задоволення конкретних потреб, але може вимагати докладного аналізу;

– економічний підхід: ресурси розподіляються відповідно до економічних чи фінансових критеріїв, таких як бюджетні обмеження, рентабельність, вартість проектів тощо. Орієнтований на ефективне використання ресурсів, але може не враховувати соціальні аспекти;

– черговий метод: ресурси розподіляються в порядку чергування, де перші отримують пріоритет перед іншими. Простий та прозорий, але може бути несправедливим щодо особливих потреб;

– стратегічне планування: ресурси розподіляються відповідно до стратегічного планування, яке враховує довгострокові цілі та пріоритети. Спрямований на досягнення стратегічних цілей та розвиток;

– політичний метод: ресурси розподіляються на основі політичних рішень, владних структур та прийнятих законів. Залежить від політичних процесів та прийнятих рішень.

Традиційні методи можуть варіюватися в залежності від конкретної сфери застосування. У різних галузях існують свої власні підходи до розподілу ресурсів, залежно від потреб та особливостей цих галузей.

Традиційні підходи до розподілу ресурсів часто передбачають ручні процеси та багато в чому покладаються на людське судження. Ці методи часто забирають багато часу, схильні до помилок і неефективні. Вони борються зі швидкими змінами та невизначеністю у виробничих операціях. Крім того, традиційні методи розподілу ресурсів часто не враховують складні взаємозв’язки та залежності між різними видами діяльності. Їм бракує здатності аналізувати великі обсяги даних і приймати миттєві рішення. Це може призвести до неоптимального розподілу ресурсів, що призводить до неефективності та збільшення витрат. Іншою проблемою традиційних методів розподілу ресурсів є нездатність адаптуватися до змін попиту, пропозиції та зміни умов виробництва. Їм часто не вистачає гнучкості

та оперативності, необхідних в сучасному світі, що швидко змінюється. Це може погіршити здатність реагувати на попит споживачів та тенденції. Крім того, традиційним методам розподілу ресурсів часто не вистачає прозорості та видимості. Через це особам, які приймають рішення, може бути важко зрозуміти, як використовуються ресурси, визначити вузькі місця та оцінити вплив рішень щодо розподілу ресурсів на результати виробництва.

Зважаючи на ці виклики, зростає потреба в більш досконалих і складних методах розподілу ресурсів. Штучний інтелект для розподілу виробничих ресурсів пропонує багатообіцяюче рішення. Використовуючи технологію ШІ, виробники можуть автоматизувати розподіл ресурсів, покращити процес прийняття рішень, підвищити ефективність і стимулювати виробництво. За умови ефективного застосування розподіл ресурсів може значно покращити виробничі операції. Це може допомогти зменшити кількість відходів, мінімізувати час простою, збалансувати робоче навантаження та покращити якість продукції. Однак традиційних методів розподілу ресурсів часто недостатньо для вирішення складності та динаміки сучасного виробничого середовища. Це де в гру вступає штучний інтелект та ІС для розподілу ресурсів для виробництва або споживачів.

Ситуація кардинально змінюється при переході до складних, динамічних, нечітких та неформалізованих ситуаціях, коли доводиться надавати допомогу населенню прифронтових районів, які знаходяться поруч з веденням бойових дій. В цьому випадку централізоване планування вже перестає бути адекватним із низки причин: старіння інформації за час проходження організаційної вертикалі, неминуче накопичення помилок при передачі нечіткою та неформалізованої інформації.

Механізм розподілу ресурсів «пропорційно до запитів» полягає в тому, що обмежені ресурси R діляться між *i* групами осіб пропорційно до їх запитів $r_i^{зап}$. При цьому кожна і-та група отримує ресурси обсягом:

$$r_i = \frac{r_i^{зап}}{\sum_i r_i^{зап}} \cdot R, \quad (1.1)$$

де сумування ведеться по усім групам, серед яких ресурс розподіляється [2].

В умовах обмеженої кількості гуманітарних ресурсів необхідно виявлення груп осіб, які будуть відповідати різним ступеням потреб у гуманітарній допомозі. Виявлення таких груп – кластерів, дозволяє застосування алгоритмів кластерного аналізу даних.

Потреба впровадження інтелектуальної системи розподілу гуманітарної допомоги виходить з того, що в умовах воєнного стану в нашій країні ми можемо не мати достатньо часового ресурсу на прийняття швидких та справедливих рішень щодо розподілу гумінітарної допомоги. Це може призвести до корумпованості, залежно від доброчесності людини, відповідальної за розподіл ресурсів та аналіз поданих заяв на отримання допомоги. Для виключення цієї можливості в цей скрутний час може прийти на допомогу інтелектуальна система, яка буде брати на себе питання розподілення ресурсів в залежності від соціального становища, потреб людей, тощо. Це може не тільки пришвидшити процес та якість розподілу, але й зняти з відповідальної за це людини моральний тиск за прийняття подібних рішень.

Наразі подібні системи використовуються в таких газулях: транспорт та логістика, енергетика, сільське господарство, медицина, промисловість, фінанси. Ці області використання вказують на потенційні переваги використання інтелектуальних систем у розподілі ресурсів. Переваги включають оптимізацію використання ресурсів, зменшення витрат, підвищення ефективності та поліпшення управлінських рішень. Тому є потреба у створенні інтелектуальної системи розподілу гуманітарної допомоги, яка займається розподілом наявних ресурсів із врахуванням потреб осіб, які її потребують.

1.2 Основні підходи у застосуванні кластерного аналізу даних при розподілі ресурсів

Щоб зрозуміти сучасні реалії поставленої задачі – аналіз даних стосовно осіб, які потребують гуманітарної допомоги, та забезпечення необхідного до виявлених тенденцій розподілу наявних ресурсів необхідно розглянути доступні ресурси, які допоможуть в майбутньому визначитись з технологіями та методами для її вирішення.

Кластерний аналіз даних дозволяє розділити набір даних з характеристиками об’єктів, які досліджуються, на споріднені групи – кластери, кожен із яких буде містити об’єкти зі схожими характеристиками. При розподілі гуманітарної допомоги характеристиками є ті атрибути, які дозволяють визначити потреби осіб у порядку їх першочерговості. Є одинокі особи, особи з інвалідністю, люди похилого віку, багатодітні сім’ї. на лінії ведення бойових дій є особи, які не мають змоги отримувати належні їм грошові виплати. У відповідності до цього може бути встановлений рейтинг виявлених кластерів, який визначатиме механізм та порядок розподілу ресурсів, коли їх наявність є обмеженою [3].

Здійснений огляд літературних джерел дозволив установити, що кластерний аналіз дозволяє сегментувати клієнтів при розподілі ресурсів, у вебаналітиці при формуванні профілів клієнтів, при плануванні логістичних процесів з географічною прив’язкою. Наведені приклади свідчать про те, що використання кластерного аналізу даних дозволить створити точну систему розподілу гуманітарної допомоги.

Розглянемо більш детальніше базові поняття та основні алгоритми кластерного аналізу даних: ієрархічну кластеризацію та алгоритми k-means і c-means.

Кластерний аналіз не накладає умов на тип об’єктів, що розглядаються, тому дозволяє досліджувати різноманіття вихідних даних довільної природи і на відміну

від класифікації перелік груп не є чітко заданим. Основними етапами алгоритму кластеризації є такі:

- підготовка даних;
- вибір способу розрахунку міри близькості між об’єктами;
- вибір алгоритму: зумовлюється вихідними даними;
- виконання алгоритму;
- представлення результатів та інтерпретація.

Перший етап полягає у підготовці даних для кластерного аналізу. У більшості випадків дані описують у вигляді таблиць, де стовпець є одним з атрибутів, а рядок об'єктом даних. На другому етапі відбувається вибір мір близькості, з допомогою якої визначається подібність об’єктів. Серед мір близькості виділяють міри подібності та міри несхожості, які у випадку числових метричних даних називають відстанями. У випадку, коли ми маємо справу з категоріальними даним, необхідно здійснити перетворення даних до числової шкали. А дані, представлені у різних числових діапазонах, необхідно нормалізувати – перетворити до одного числового діапазону, зазвичай – проміжку від 0 до 1 [4].

Найчастіше для визначення відстані між об’єктами, атрибути яких представлені у числових шкалах, використовують відстані, наведені у таблиці 1.1.

Для даних, представлених категоріальними атрибутами, у якості міри несхожості об’єктів x_i і x_j, використовують відстань Хеммінга, яка дорівнює кількості атрибутів, значення яких для об’єктів відрізняються і є метрикою.

Для даних, представлених категоріальними атрибутами, у якості міри несхожості, використовують також відсоток незгоди:

$$d(x_i, x_j) = \frac{l}{m}, \tag{1.2}$$

де m – загальна кількість категоріальних ознак, l – кількість не співпадаючих ознак об’єктів x_i і x_j, для яких $x_{ik} \neq x_{jk}$, $k \in \{1,.2,..,m\}$.

Таблиця 1.1 – Міри відстаней між об’єктами для метричних шкал

Метрика	**Формула**
Відстань Евкліда	$d_E(x_i, x_j) = \sqrt{\sum_{t=1}^{m}(x_{it} - x_{jt})^2}$
Квадрат відстані Евкліда	$d_E^2(x_i, x_j) = \sum_{t=1}^{m}(x_{it} - x_{jt})^2$
Манхетенська відстань (відстань міських кварталів)	$d_H(x_i, x_j) = \sum_{t=1}^{m}\left\|x_{it} - x_{jt}\right\|$
Відстань Чебишева	$d_\infty(x_i, x_j) = \max_{1\le t\le m}\left\|x_{it} - x_{jt}\right\|$

На наступному етапі вибирається алгоритм, з якого групуються об'єкти. Вибір алгоритму є складним завданням, оскільки отриманий результат багато в чому залежить від алгоритму. Найчастіше доводиться використовувати кілька алгоритмів, тобто їх комбінувати для більш точного результату.

На четвертому етапі відбувається реалізація обраного алгоритму (або кількох алгоритмів). Результатом цього етапу є групування об’єктів за кластерами.

П'ятий етап передбачає представлення отриманого групування у зручному для інтерпретації вигляді. Подання результатів кластеризації покликане допомогти найточніше інтерпретувати результати виконання алгоритму.

В даний час існує досить багато алгоритмів ластерного аналізу. Прийнято всі алгоритми поділяти на дві категорії: ієрархічні та неієрархічні.

При здійсненні ієрархічного кластерного аналізу будують систему вкладених розбиттів набору даних на кластери й отримують на виході дерево кластерів, коренем якого є весь набір даних, а листи є об’єктами набору даних.

Серед алгоритмів ієрархічної кластеризації виділяють агломеративні алгоритми – побудова кластерів здійснюється знизу вгору, та дивізимні алгоритми – побудова кластерів здійснюється зверху вниз. Результат кластеризації графічно представляють у вигляді дендрограми (рис. 1.1).

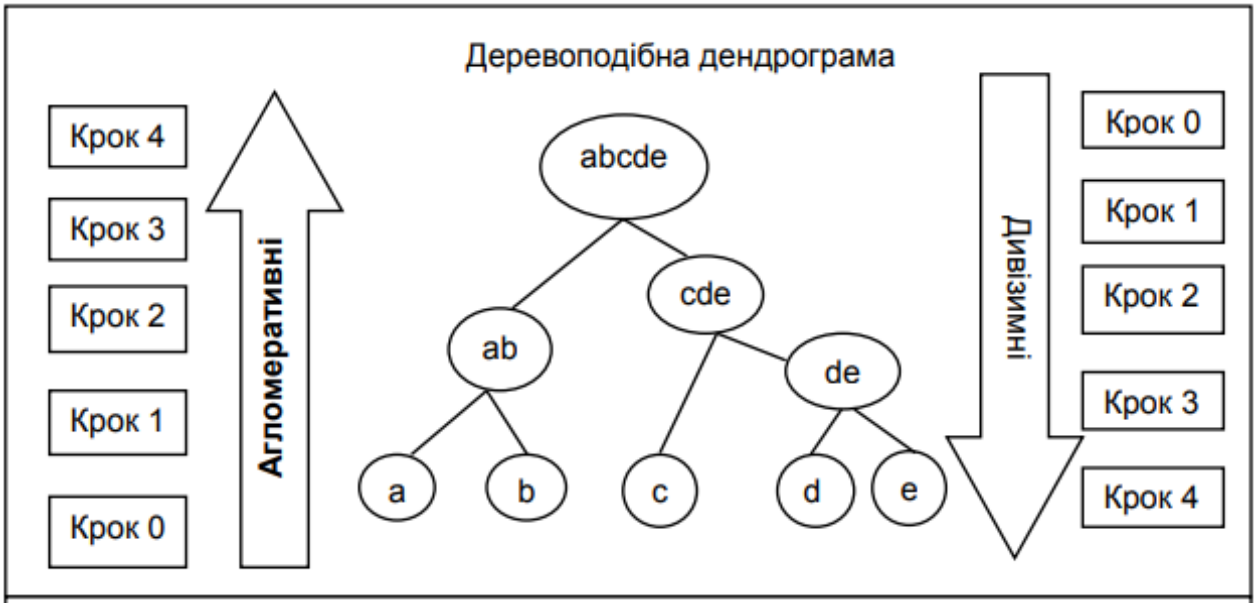

Рисунок 1.1 – Дендрограма алгоритмів ієрархічної кластеризації

Ієрархічний алгоритм є ітераційним, на кожному етапі об’єднують кластери, які є найбличими один до одного. Оптимальною буде така кількість кластерів, яка дорівнює різниці кількості об’єктів та номреру етапу ітерації, на якому відбувся перший різкий скачок відстані у об’єднаних кластерів у порівняні з попередніми відстанями.

На етапах ітерації ієрархічного кластерного аналізу застосовують різні методи зв’язку, які є критеріями під час об’єднання чи поділу. Найчастіше це метод найближчого сусіда – при об’єднанні кластерів за відстань між кластерами приймають відстань між найближчими об’єктами цих кластерів. Або метод дальнього сусіда – при об’єднанні кластерів за відстань між кластерами приймають відстань між дайдальшими об’єктами цих кластерів.

Неієрархічні алгоритми будують одне розбиття об’єктів набору даних на кластери що не перетинаються. Серед них виділяють чіткі та нечіткі алгоритми. Чіткі алгоритми кожно об’єкту ставлять у відповідність номер кластері, якому він належить, а нечіткі – ймовірність, з якою об’єкт належить до кожного кластеру [5].

Серед неієрархічних алгоритмів найбільш поширеними є чіткий алгоритм k-means та нечіткий алгоритм c-means. Ці алгоритми також є ітераційними, а

оптимальність розбиття на клатери визначається шляхом оптимізації цільової функції, яка є внутрішньокластерною квадратичною похибкою розбиття і для алгоритму k-means має вигляд:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} d^2(x_{ij}, c_j), \tag{1.3}$$

де c_j – j –й кластер, k – кількість кластерів,

x_{ij} – i -й об’єкт j –го кластера, n_j – кількість об’єктів у j –му кластері,

$d(x_{ij}, c_j)$ – відстань між i -м об’єктом j –го кластера та його центром ваги – центроїдом.

Центр ваги кластеру розраховують як точку у просторі ознак з координатами, що є середнім арифметичним усіх значень кожної ознаки об’єктів кластера.

Для виконання алгоритму k-means необхідно задати кількість кластерів, на які буде розбито сукупність об’єктів набору даних. на початковому етапі рандомно задаються центри кластерів, потім на кожному з етапів визначається відстань об’єктів до центрів та здійснюється перегрупування об’єктів до тих кластерів, центри яких є найближчими. Робота алгоритму припиняється, коли буде досягнено мінімум цільової функції.

Нечіткий алгоритм кластеризації c-means є вдосконаленням алгоритму k-means, він визначає ймовірність віднесення об’єкту до кожного з кластерів, формуючи матрицю нечіткого розбиття. Цільова функція алгоритму c-means має вигляд:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n_j} u_{ji}^{w} d^2(x_{ij}, c_j), \tag{1.4}$$

де u_{ji} – елемент матриці нечіткого розбиття U, W – коефіцієнт нечіткості.

Усі інші компоненти формули 1.4 аналогічні компонентам цільової функції, заданої формулою 1.3.

Розглянемо підходи, які застосовують для оцінки якості кластеризації. Після створення кластерного рішення зазвичай виникає питання, наскільки воно стійке та статистично значуще. Тут існує емпіричне правило – стійке групування повинне зберігатися при зміні методів кластеризації: наприклад, якщо результати ієрархічного кластерного аналізу мають частку збігів понад 70% із угрупуванням за методом k-середніх, припущення про стійкість приймається.

У теоретичному плані проблема перевірки адекватності кластеризації не вирішена, принаймні, без використання іншого виду аналізу чи апріорного знання належності об'єктів до відповідних класів.

Якщо детально проаналізувати рекомендовані методи перевірки адекватності кластеризації, то можна зазначити методи, які мають недоліки:

− тести на значущість розбиття даних на кластери (багатомірний дисперсійний аналіз) завжди дають значний результат, є складними у реалізації;

− методика повторних (випадкових) вибірок – не доводить обґрунтованості рішення;

− тести значущості для ознак, не використаних під час кластеризації, придатні лише за наявності повторних вимірів;

− методи Монте-Карло мають велику розрахункову складність та доступні лише досвідченим математикам.

Із використуваних на практиці критеріїв оцінки якості результатів кластеризації можна виділити кілька підходів до валідації кластерів:

− зовнішня валідація, яка полягає у порівнянні підсумків кластерного аналізу із заздалегідь відомим результатом (тобто мітки кластерів відомі апріорі);

− відносна валідація, яка оцінює структуру кластерів, змінюючи різні параметри одного і того ж алгоритму (наприклад, число груп k);

– внутрішня валідація, яка використовує внутрішню інформацію процесу об’єднання у кластери (якщо зовнішня інформація відсутня);

– оцінка стабільності об’єднання у кластери (або спеціальна версія внутрішньої валідації), яка використовує методи ресемплінгу.

Одна з проблем машинного навчання без вчителя полягає в тому, що методи кластеризації формуватимуть групи, навіть якщо аналізований набір даних є цілком випадковою структурою. Тому першим завданням валідації, яку рекомендується виконати перед початком кластерного аналізу, є оцінка загальної схильності даних до об’єднання в кластери (clustering tendency).

Статистика Хопкінса (англ. Hopkins) є одним із індикаторів тенденції до групування. Вона заснована на нульовій гіпотезі про те, що дані не мають схильності до групування. Для її розрахунку створюється сукупність наборів об’єктів із кількістю об’єктів, меншою або рівної вихідному набору даних, згенерованих випадковим чином на основі розподілу з тим самим стандартним відхиленням, що і оригінальний набір даних. Статистика Хопкінса H визначається за формулою:

$$H = \frac{\sum_{n} w_i}{\sum_{n} q_i + \sum_{n} w_i}, \qquad (1.5)$$

де w_i – відстань між об'єктами вихідного набору даних,

q_i – відстань між об'єктами вихідного набору даних та згенерованими об’єктами;

n – кількість об’єктів набору даних.

Якщо ця статистика первершує значення 0,5, це свідчить про те, що груповані об’єкти розподілені випадковим чином і не мають схильності до кластеризації. значення, менші за 0,25 показують, що з 90% ймовірністю у наборі даних є кластери [6].

При виявленні схильності до утворення кластерів у випадку реалізації алгоритму k-means необхідно визначити оптимальну кількість кластерів, оскільки слабким місцем цього алгоритму є необхідність на його початку задавати кількість кластерів. З цією метою доцільно застосувати метод силуета, основні етапи якого є наступними:

− здійснюється кластеризація методом k-means (або c-means,) *i* раз з різним числом кластерів;

− розраховується *x(i)* як середню величина близькості між усіма об’єктами всередині кластера;

− розраховується *y(i)* як мінімальна середню величину близькості між об’єктами різних кластерів.

− розраховується тестова статистика – коефіцієнт силуета *C(i)*:

$$C(i) = \frac{y(i) - x(i)}{\max(y(i), x(i))}. \quad (1.6)$$

Коефіцієнт силуета *C(i)* варіюється від -1 до +1. Значення близькі до -1, означають, що об'єкти загалом неправильно співвіднесені зі своїми кластерами, що означає, що слід змінити кількість кластерів. Значення близькі до 1 говорять про те, що вибрано правильну кількість кластерів. У виборі числа кластерів також допомагає візуалізація, зроблена на основі коефіцієнта силуету.

Ще дним методом визначення оптимального числа кластерів є метод ліктя (англ. Elbow method). Даний методу пропонує графічний метод визначення числа кластерів:

− кластеризація даних для різної кількості кластерів k;

− для кожного кластера від 1 до k розраховується внутрішньокластерна сума квадратів Wss;

− побудова графіка, де на осі абсцис відзначається число кластерів від 1 до k, а на осі ординат відповідне значення Wss;

− число по осі абсцис, де спостерігається сильний перегин на графіку, є числом кластерів, що рекомендується.

Логіка 4 кроку алгоритму пояснюється тим, що додаткове збільшення кластерів хоча б на один не призведе до сильного зменшення внутрішньокластерної відстані Wss. Хоча метод ліктя дуже простий при побудові та аналізі, до результатів такого підходу слід ставитися з обережністю, оскільки у даного методу немає формальних критеріїв вибору оптимальної кількості кластерів, крім, як візуально. Його бажано використовувати разом із коефіцієнтом силуету [7].

Таким чином, кластеризація є невід'ємною частиною обробки масивів інформації, помітно полегшуючи роботу з нею для визначеня розподілу осіб, які потребують гуманітарної допомоги, на групи кластерів, що містять споріднені по потребам урепи населення. Вибір різних алгоритмів кластеризації залежить від конкретного випадку і не може бути визначений однозначно. Для досягнення найкращого результату необхідно експериментувати з способів визначення мір близькості між об'єктами, а іноді навіть змінювати алгоритм. Жодного єдиного рішення не існує, тому кращим підходом є застосування різних алгоритмів та вибір такого, який дає кращий результат. Важливе значення при розроділі ресурсів має також інтерпретація отриманих результатів.

1.3 Мережеві ресурси у сфері надання гуманітарної допомоги

Мережеві ресурси у сфері надання гуманітарної допомоги є невід'ємною частиною в нашому житті, бо без них дуже важко під час кризових ситуацій, а саме природні катастрофи, конфлікти чи епідемії. На міжнародному рівні є цілий ряд ресурсів для гуманітарної допомоги. Зробимо їх огляд.

Humanitarian Data Exchange (HDX) – вебплатформа [8], створена Офісом координації гуманітарних справ Організації Об'єднаних Націй (ООН) (рис. 1.2).

Надає можливість ділитися та доступ до геопросторових даних, статистики, зображень та іншої інформації для поліпшення прийняття рішень в гуманітарних справах.

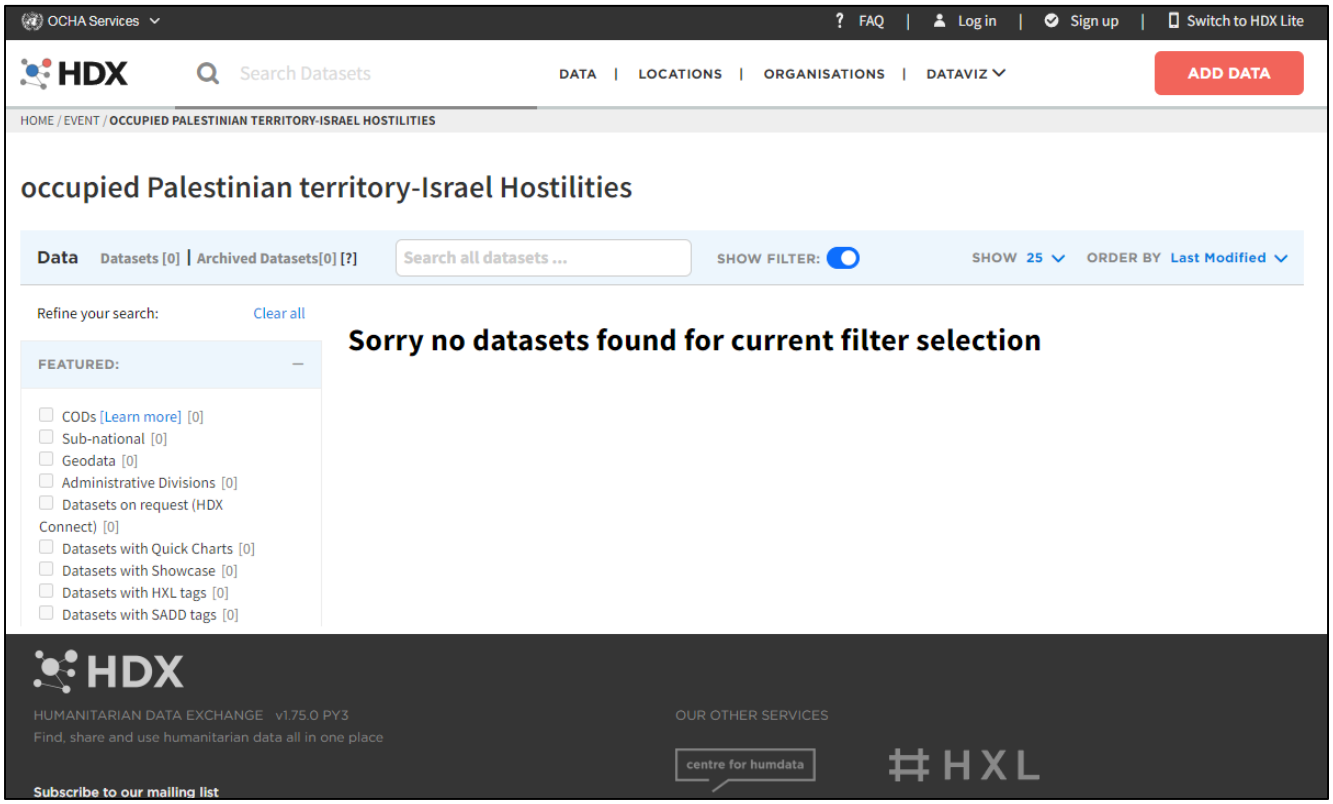

Рисунок 1.2 – Вебплатформа Humanitarian Data Exchange

Центр гуманітарних даних зосереджений на збільшенні використання та впливу даних у гуманітарному секторі. Ним керує Управління ООН з координації гуманітарних питань (OCHA) . Як частина Відділу управління інформацією OCHA, роль Центру полягає в тому, щоб дозволити організації реалізувати свої амбіції бути більш керованими даними та аналітичними, а також забезпечити відповідальний підхід до технології та управління даними.

Діяльність Центру організована навколо чотирьох робочих потоків: служби даних , наука про дані , відповідальність за дані , навчання та практика . Наша підтримка надається персоналу OCHA та партнерам, приділяючи пріоритетну увагу місцям, де проводяться гуманітарні операції.

Робота Центру стала більш актуальною та важливою після його створення в 2017 році. Дані тепер міцно включені в порядок денний Організації Об’єднаних Націй із створенням у 2020 році Стратегії даних Генерального секретаря , яка прагне «розблокувати потенціал даних у Сім'я ООН для кращої підтримки людей і планети». Наш Загальний план зобов’язує організацію до Організації Об’єднаних Націй 2.0, яка надає пріоритети даним, цифровим технологіям, інноваціям, стратегічному передбаченню та поведінковій науці.

Подібним чином попит на кращі дані та аналіз зріс у гуманітарному секторі, оскільки він бореться з безпрецедентними потребами та недостатніми коштами. Гуманітарія також стикається з розрахою, оскільки заклики до локалізації та деколонізації вимагають радикального переосмислення сектора. У цьому контексті дані є потужнішими, ніж будь-коли, і ставки ніколи не були такими високими: у 2023 році понад 330 мільйонів людей потребуватимуть гуманітарної допомоги.

Разом із нашими партнерами за останні кілька років дана компанія досягла значного прогресу в усіх напрямках роботи. Наведемо деякі приклади.

Більше даних, які передаються та використовуються: залучення з HDX зросло з трохи менше ніж 20 000 унікальних користувачів на місяць у липні 2017 року до понад 130 000 на місяць до середини 2023 року. З близько 300 організацій, які обмінюються даними на HDX, 50 відсотків є глобальними, 14 відсотків регіональними та 36 відсотків національними або місцевими організаціями.

Усунення прогалин у даних: у 2019 році було запроваджено мережу даних HDX для відстеження наявності основних даних у пріоритетних гуманітарних операціях. Повнота даних у різних країнах і категоріях зросла з 54 відсотків у 2019 році до 75 відсотків до середини 2023 року.

Керівництво схвалено та прийнято: Оперативне керівництво IASC щодо відповідальності за дані в гуманітарних діях було схвалено в лютому 2021 року та оновлено в квітні 2023 року після консультації під керівництвом Центру з сотнями зацікавлених сторін. Рекомендації ОСНА щодо відповідальності за дані були

погоджені в жовтні 2021 року, а до кінця 2022 року OCHA та його партнери ухвалили заходи щодо відповідальності за дані в 19 контекстах реагування.

Аналітичний механізм для попереджувальних дій: протягом початкового пілотного періоду з 2020 по 2022 рр. Центральний фонд реагування на надзвичайні ситуації ООН виділив 89 мільйонів доларів США на попереджувальні дії на основі механізмів запуску, розроблених Центром та офісами OCHA.

Прозорість моделі: дванадцять прогностичних моделей від академічних і гуманітарних партнерів пройшли через систему експертної оцінки Центру за підтримки групи технічних і етичних експертів. Результати були оприлюднені через модельні звіти.

Грамотність даних: за останні три роки навчальними ресурсами Центру користувалися більше 13 000 людей. Нещодавно ми запропонували семінари з прогнозування клімату та створили серію рекомендацій щодо клімату , щоб допомогти гуманітаріям отримувати доступ, аналізувати та інтерпретувати різні типи кліматичних даних.

Digital Humanitarian Network – Цифрова гуманітарна мережа — це консорціум, який дозволяє волонтерським і технічним спільнотам (V&TC) взаємодіяти з гуманітарними організаціями, які шукають їхні послуги (рис. 1.3).

Веб-сайт Digital Humanitarian Network (DHNetwork) був запущений 9 квітня 2012 року співзасновниками Андрієм Веріті. Наприкінці 2019 року DHN відзначив, що традиційні організації з надання допомоги стають більш спроможними використовувати сучасні інструменти, і оголосив, що вони більше не будуть активувати групи реагування на кризи. Метою DHNetwork є підтримка гуманітарних організацій у їхніх зусиллях з реагування на стихійні лиха по всьому світу.

Мережа складається з членських волонтерських і технічних спільнот (організацій, які керують мережами технічно підготовлених волонтерів по всьому світу, яких можна активувати для підтримки операцій з реагування на катастрофи та отримання інформації з обмеженим часом). Ці групи мають низку навичок від

картографування ГІС, краудсорсингу, аналізу та збору даних до управління волонтерами та проектування процесів. HNetwork об’єднує групи, які існували роками, під одну парасольку та забезпечує єдиний вихід для традиційних служб реагування на доступ до організацій.

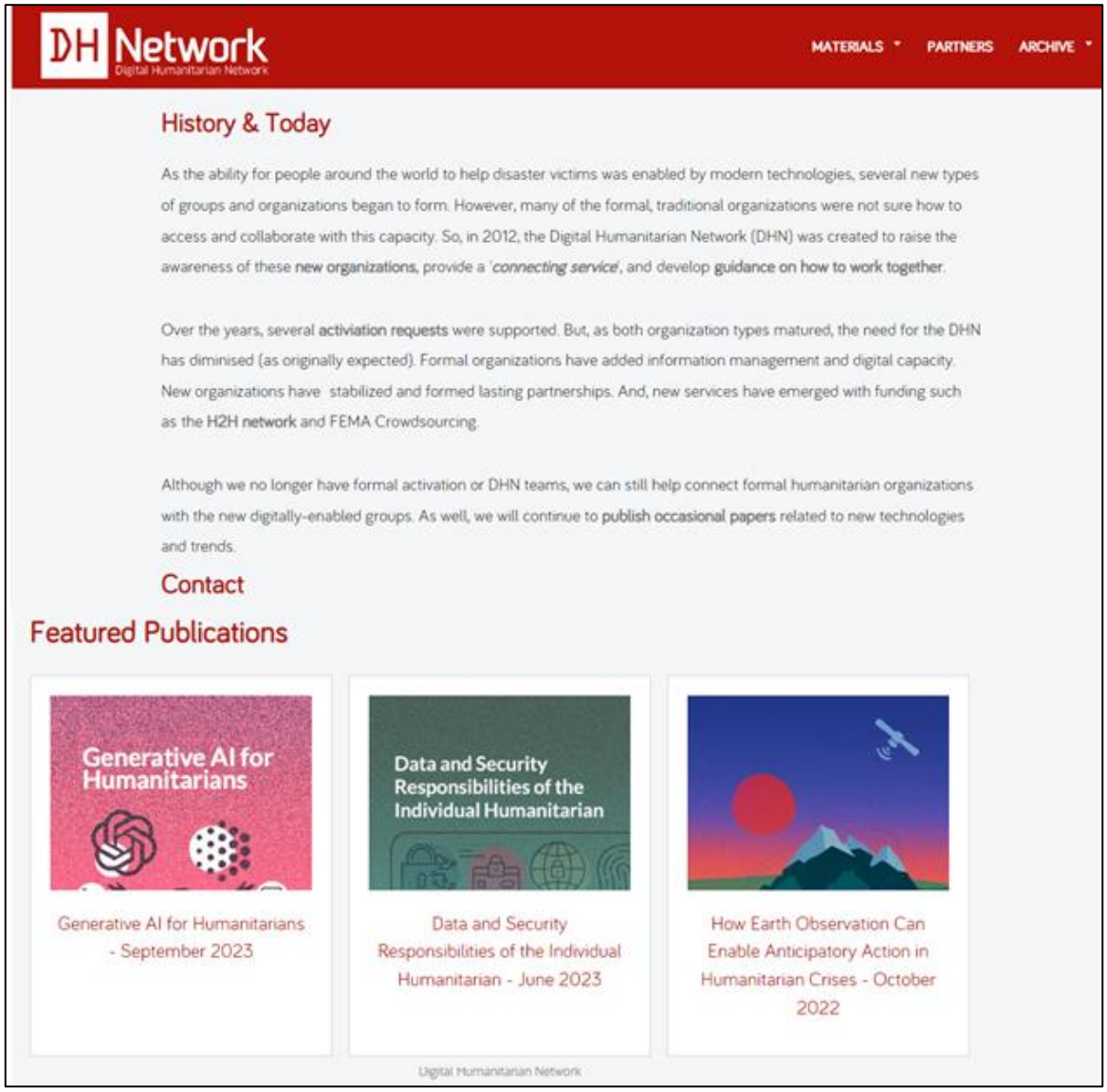

Рисунок 1.3 – Мережевий ресурс Digital Humanitarian Network

Мережа об’єднує численні волонтерські та технічні спільноти, тим самим підвищуючи їх видимість як між собою, так і серед традиційної гуманітарної спільноти, і визначила процес активації між VT&C та координаторами, щоб

традиційні організації могли подати один запит і покластися на DHNetwork для створення команда рішення з відповідними членами V&TC [9].

Ці ресурси допомагають збирати, обробляти та розповсюджувати інформацію у реальному часі, полегшуючи реагування на екстрені ситуації та координацію гуманітарних заходів.

На рівні держави на початок 2024 року в Україні розроблено веб-платформу Aidmonitor.org яка почала впроваджуватися для контроля за наданням та розподілом гуманітарної допомоги міжнародними донорськими організаціями (рис. 1.4).

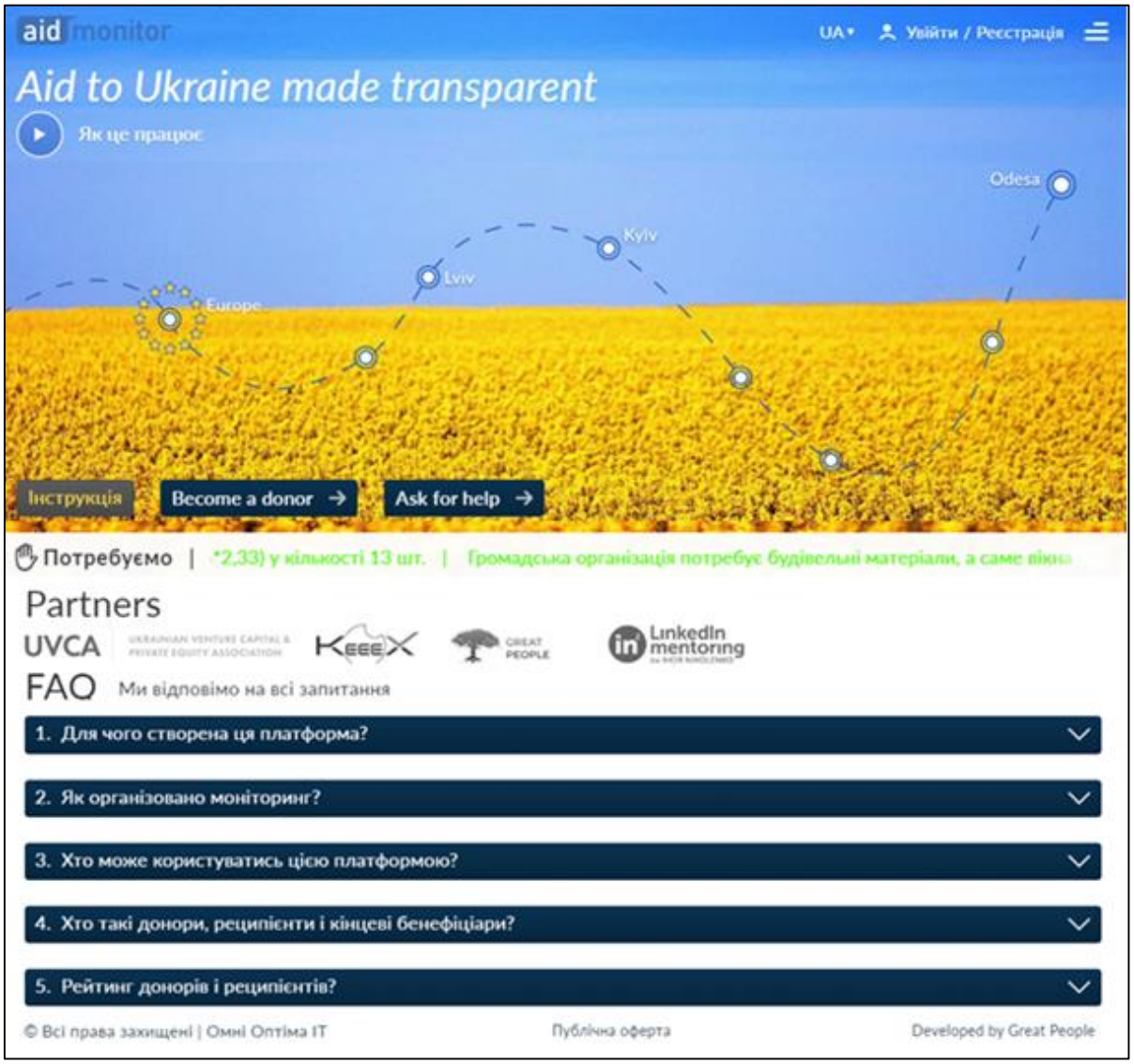

Рисунок 1.4 – Система Aidmonitor.org

Однак є проблема у налагодженні узгодженої взаємодії волонтерів із особами, які проживають безпосередньо у прифронтових територіях та оптимальному механізмі розподілу допомоги

1.4 Постановка задачі

Провівши аналіз підходів до розподілу гуманітрної допомоги зроблено висновок про необхідність розробки інтелектуальної системи розподілу гуманітарної допомоги з використанням методів кластерного аналізу даних.

Об’єктом дослідження є процес розподілу ресурсів у сфері гуманітарної допомоги.

Предметом дослідження є програмні засоби та методи розроділу ресурсів гуманітарної допомоги на основі інтелектуального аналізу даних.

Мета дослідження – підвищення ефективності розподілу гуманітарної допомоги шляхом розробки вебзастосунку із використанням кластерного аналізу даних.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

− дослідити теоретичні засади розподілу гуманітарної допомоги та здійснити аналіз мережевих ресурсів у цій сфері;

− обґрунтувати вибір технологій та інструментальних засобів розробки інтелектуальної системи;

− розробити та здійснити програмну реалізацію системи розподілу гуманітарної допомоги з використанням алгоритмів кластерного аналізу.

Висновки до розділу 1

Бурхливі темпи інформатизації сучасного суспільства супроводжуються впровадженням цифрових технологій в усі сфери оточуючої дійсності та у сферу

обліку і розподілу гуманітарної допомоги зокрема. Обсяги гуманітарної допомоги різко зросли в умовах ведення воєнних дій на території країни й охоплюють велику кількість волонтерів, громадських вітчизняних та міжнародних організацій і благодійних фондів. Різко зросла потреба оперативного надходження допомоги тим, хто її потребує. Вирішення цієї проблеми обумовлює необхідність створення інформаційних систем для покращення прогнозування попиту та керування ланцюгами постачання і розподілу гуманітарної допомоги.

Необхідність кластеризації виникає з потреби групувати схожі об’єкти чи ситуації для ефективного визначення та реагування на події. Кластеризація дозволяє виділити групи об’єктів зі спільними характеристиками, спрощуючи аналіз та оптимізуючи розподіл ресурсів. Здійснений аналіз дозволив виділити основні алгоритми, які доцільно задіяні для кластеризації – алгоритм ієрархічної кластеризації та алгоритми k-means і c-means. Визначено основні етапи даних алгоритмів, підходи до визначення якості кластеризації та оптимальної кількості кластерів: статистика Хопкінса, коефіцієнт силуета та метод ліктя.

Використання кластеризацієї сприяє більш точній і швидкій ідентифікації важливих патернів та забезпечує оптимальний розподіл гуманітарної допомоги в ситуаціях, коли час є критичним фактором. Створення подібної системи спрощує та економить час під час винесення рішення про розподіл гуманіторної допомоги, що дасть приріст продуктивності працівникам подібних організацій.

2 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1 Мова програмування та бібліотеки Python

Python – це високорівнева мова програмування, що відрізняється ефективністю, простотою та універсальністю використання. Він широко застосовується у розробці вебзастосунків та прикладного програмного забезпечення, а також у машинному навчанні та обробці великих даних. За рахунок простого та інтуїтивно зрозумілого синтаксису є однією з поширених мов програмування [10].

У Data Science використовують Python для включення очищення та розмітки даних, пошуку та обробки статистичної інформації, її візуалізацію у вигляді діаграм, графіків тощо. За допомогою бібліотеки Python ML класифікуються зображення, тексти, пошуковий трафік, здійснюється розпізнавання облич та мовлення, глибинне машинне навчання [11].

Можливості Python використовуються тестувальниками та розробниками для пошуку та виправлення помилок, автоматичного складання, розробки прототипів програмного забезпечення, управління проектами тощо. Крім того, за допомогою середовищ модульного тестування Python здійснюється перевірка функцій [12]. Також цією мовою створюються тестові скрипти, що імітують різні сценарії використання програмного забезпечення. Розробники апаратних платформ (наприклад, IBM, Hewlett-Packard, Intel) також використовують Python для тестування своєї продукції.

Багато великих інтернет-компаній, таких як Google, Facebook, програмують на Python свої найвідоміші проекти, наприклад, Instagram, YouTube, Dropbox і т.д. Ця мова дозволяє вести веброзробку на стороні сервера, тому що його велика бібліотека включає безліч рішень для реалізації складних серверних функцій. За рахунок своєї простоти використання Python широко застосовується невеликими

командами та одиночними розробниками для створення сайтів, десктопних та мобільних веб-додатків.

У Python оператори коду виконуються послідовно за допомогою програми-інтерпретатора. Якщо під час виконання програми зустрічається помилка, воно відразу ж припиняється. Це дозволяє Python-розробнику швидко виявити та усунути недоліки, але водночас знижує продуктивність.

Динамічна типізація – це автоматичне зв'язування змінної та типу в момент, коли їй надається певне значення. Такий механізм прискорює написання програми в різних ситуаціях (наприклад, під час роботи зі змінними даними), але підвищує ймовірність помилки. Об'єктно-орієнтованість – написана на Python програма є сукупністю об'єктів, кожному з яких присвоєно певний клас і місце в ієрархії. Таким чином, простіше керувати процесом програмування, що особливо важливо при створенні складних проектів.

Python за своїм синтаксисом та граматикою близький до природних мов. Завдяки цьому програмісту з його допомогою легше описати різні структури даних та операції, що також прискорює та спрощує написання коду. Крім того, це робить ПЗ, написане на Python, менш залежним від платформи.

Програмісти та спільнота користувачів Python створили для цієї мови велику кількість біліотек, де містяться оптимізовані і багаторазово використовувані фрагменти коду для вирішення практично будь-яких завдань. Це прискорює роботу над проектом або аналізом даних. розглянемо їх більш детально.

Бібліотека NumPy (англ. Numerical Python) є однією з основних і найпотужніших бібліотек для наукових обчислень у мові програмування Python. Вона надає широкі можливості для роботи з масивами даних та векторизованими обчисленнями, що дозволяє виконувати швидкі та ефективні операції над числовими даними. Однією з ключових особливостей NumPy є об'єкт ndarray (N-dimensional array), який представляє собою многовимірний масив однакових типів даних [13]. Цей об'єкт забезпечує високу швидкість обробки даних та оптимізований доступ до елементів масивів, що робить NumPy ідеальним

інструментом для обробки великих об'ємів даних у наукових дослідженнях та інженерних застосуваннях. Бібліотека надає величезний набір функцій для математичних обчислень, лінійної алгебри, трансформацій Фур'є, випадкових чисел та інших операцій. Це дозволяє науковцям та інженерам виконувати складні математичні обчислення та статистичний аналіз даних з високою точністю та ефективністю [14].

Додатково, NumPy інтегрується з іншими популярними бібліотеками для наукових обчислень в Python, такими як SciPy, Matplotlib та Pandas, що розширює його можливості у сфері аналізу даних, візуалізації та моделювання. Загалом, NumPy відіграє ключову роль у розвитку та використанні наукових та інженерних додатків у Python, завдяки своїй потужності, ефективності та багатофункціональності.

Бібліотека Matplotlib є однією з найпоширеніших та потужних інструментів в мові програмування Python для візуалізації даних та створення графіків. Її використання в наукових дослідженнях, інженерних проектах та аналізі даних дозволяє зробити результати зрозумілішими, виразнішими та ефективнішими. Matplotlib надає широкий спектр можливостей для створення різноманітних типів графіків, включаючи лінійні графіки, діаграми розподілу, гістограми, кругові діаграми, контурні графіки та теплові карти. Кожен з цих типів графіків може бути налаштований та відформатований з використанням широкого спектру параметрів, що дозволяє створювати візуально привабливі та інформативні зображення. Однією з ключових переваг Matplotlib є його простота використання.

Бібліотека має зрозумілий та легкий у використанні інтерфейс, що дозволяє навіть початківцям швидко створювати графіки та діаграми. Крім того, Matplotlib інтегрується з іншими бібліотеками для наукових обчислень у Python, такими як NumPy та Pandas, що полегшує використання графіків у складних обчислювальних процесах. Незважаючи на свою простоту, Matplotlib також надає високий рівень кастомізації. Користувачі можуть налаштовувати різні аспекти графіків, включаючи колір, стиль ліній, розмір шрифтів та багато іншого [15]. Це дозволяє

створювати графіки, що відповідають конкретним вимогам та виглядають професійно. Узагальнюючи, Matplotlib відіграє важливу роль у візуалізації даних та результатів наукових досліджень у Python завдяки своїй потужності, гнучкості та простоті використання.

Бібліотека Pandas є ключовим інструментом у сфері аналізу даних та обробки табличних даних у мові програмування Python. Вона надає зручний та ефективний інтерфейс для роботи з даними, що дозволяє легко завантажувати, зберігати, обробляти, аналізувати та візуалізувати інформацію з таблиць та наборів даних. Однією з ключових особливостей Pandas є об'єкт DataFrame, який представляє собою двовимірну таблицю даних з рядками та стовпцями. Цей об'єкт дозволяє легко виконувати різноманітні операції з даними, такі як фільтрація, сортування, об'єднання, групування та агрегування. Крім того, Pandas надає широкий спектр методів для роботи з часовими рядами, що робить її важливим інструментом для аналізу часових послідовностей даних [16].

Ще однією важливою особливістю Pandas є підтримка різних форматів даних, таких як CSV, Excel, SQL, JSON, HDF5 та інші. Це дозволяє легко імпортувати та експортувати дані з різних джерел, що спрощує роботу з реальними наборами даних. Крім того, Pandas інтегрується з іншими потужними бібліотеками для аналізу даних та візуалізації у Python, такими як NumPy, Matplotlib, SciPy та інші. Це дозволяє створювати складні аналітичні звіти та візуалізації з використанням різноманітних інструментів. Загалом, Pandas інструментом для аналізу даних та виконання різноманітних завдань у сфері дослідження, аналітики та машинного навчання у Python [17].

Бібліотека Scikit-learn (Sklearn) є однією з найпопулярніших та потужних бібліотек для машинного навчання та аналізу даних у мові програмування Python. Вона надає широкий спектр інструментів для класифікації, регресії, кластеризації, вимірювання відстаней, виявлення аномалій, вибору ознак та багато іншого. Однією з ключових особливостей Scikit-learn є простота використання та консистентний API, що робить його ідеальним інструментом для як початківців,

так і досвідчених спеціалістів у сфері машинного навчання. Бібліотека містить реалізації різноманітних алгоритмів машинного навчання, таких як метод опорних векторів (SVM), випадковий ліс (Random Forest), наївний баєсівський класифікатор, градієнтний бустінг та багато інших [18].

Scikit-learn надає інструменти для підготовки та передобробки даних для аналізу та моделювання. Ще однією важливою особливістю Scikit-learn є його вбудована підтримка для оцінки моделей та підбору параметрів шляхом хресної перевірки (cross-validation). Це дозволяє ефективно оцінювати якість моделей та знаходити найкращі параметри для них, що є ключовим етапом у розробці машинно-навчальних моделей. Бібліотека Scikit-learn також інтегрується з іншими потужними інструментами для аналізу даних та візуалізації, такими як NumPy, Pandas та Matplotlib [19]. Це дозволяє легко інтегрувати різноманітні операції обробки даних та візуалізації з процесом розробки моделей машинного навчання.

2.2 Середовище розробки Jupyter Notebook

Jupyter Notebook – це середовище розробки, передачі та запуску коду, де можна побачити результат виконання коду та його окремих фрагментів, існує як вебсервіс, тобто є доступним через інтернет та дозволяє передавати код іншим розробникам (рис. 2.1).

Jupyter Notebook можна використовувати як своєрідне середовище розробки. Відмінність від традиційного середовища розробки в тому, що код можна розбити на окремі частини та виконувати їх у довільному порядку. У такому середовищі розробки можна, наприклад, написати функцію і перевірити її роботу, без запуску програми повністю. А ще можна змінити порядок виконання коду. Можна окремо завантажити файл у пам'ять, окремо перевірити його, окремо обробити вміст.

Найчастіше з Jupyter Notebook працюють програмісти на Python. Так склалося історично: проект виріс із IPython, хоча зараз платформа має підтримку і для інших мов. Jupyter Notebook підтримує мови Ruby, Perl, R, MATLAB, Julia та

інші. Відмінність Jupyter Notebook від традиційних середовищ розробки – у його інтерактивності. Програма дозволяє запускати окремі ділянки та блоки коду, виконувати їх у будь-якому порядку. А результати роботи відразу можна вивести у те саме вікно поруч із кодом. А ще в jupyter-ноутбуках є висновок результату одразу після фрагмента коду. Наприклад, можна прямо в середині коду побачити побудований графік, отримати попередні цифри чи іншу візуалізацію [20].

Рис. 2.1 – логотип Jupyter Notebook

Основні сфери використання середовища – big data та data science, машинне навчання, математична статистика та аналітика. У цих напрямках знадобилася здатність Jupyter Notebook виводити дані туди, де написаний код. В одному місці зібрані ділянки коду, результати їх виконання, таблиці, ділянки та графіки. Але спробувати Jupyter Notebook для своїх проектів можна і поза цією сферою. Якщо галузь має на увазі часту роботу з документами та графіками.

Можливості Jupyter Notebook:

– писати код у спеціальному середовищі з підсвічуванням синтаксису, виправленням помилок та іншими можливостями IDE;

– запускати різні ділянки коду довільної послідовності або написану програму цілком;

– завантажувати якісь дані, обробляти та перетворювати їх, не торкаючись при цьому інших ділянок програми;

– вставляти та виводити результати, включаючи візуалізацію, прямо посеред коду;

– ділитись кодом з іншими розробниками та давати їм загальний доступ до проекту;

– організовувати командну роботу, коли кожен програміст має своє завдання, пов’язане з іншими;

– писати текст, що супроводжує, і оформляючи його так, щоб він виглядав красиво і зрозуміло.

Jupyter Notebook дає ті ж можливості, що стандартна IDE, але при цьому він більш гнучкий і, як кажуть його творці, документоцентричний. Тобто все написане виглядає як документ та зібране в одному місці.

Інструмент Jupyter Notebook став необхідним для багатьох наукових досліджень та програмувальних проектів у сфері обробки даних та наукового обчислення, зокрема в сферах машинного навчання, статистики, біоінформатики та інших. Його популярність пояснюється зручністю та ефективністю використання завдяки можливості об'єднання коду, текстових коментарів та візуалізацій у єдиному документі. Даний інструмент дозволяє створювати інтерактивні середовища, що сприяють зрозумінню та аналізу даних, а також обміну знаннями та результатами досліджень [21].

Завдяки можливості використання різних мов програмування, але передусім Python, Jupyter Notebook став основним інструментом для вивчення та викладання програмування, а також для розв'язання різноманітних завдань у галузі науки та техніки. Відкритий формат файлу та можливість легкої конвертації у різні формати (наприклад, PDF або HTML) роблять його ідеальним інструментом для публікації результатів досліджень та спільної роботи над проектами.

У середовищі розробки Jupyter Notebook є можливість інтерактивної роботи з кодом, підтримка візуалізації даних, можливість додавання та форматування текстових блоків для пояснень та документації коду, а також зручний інтерфейс користувача. Це дозволяє ефективно організовувати робочий процес та полегшує

співпрацю між різними учасниками досліджень чи проектів. Відповідно, Jupyter Notebook став невід'ємною частиною сучасного наукового та програмувального середовища, яке значно полегшує вирішення складних завдань та сприяє продуктивному обміну ідеями та знаннями.

2.2 Середовище веброзробки PyCharm

Для створення backend частини веб-застосунків на Python, ви можете використовувати різні фреймворки та бібліотеки. Найбільш популярним з них є наступні (рис. 2.2):

− PyCharm: кроссплатформенне інтегроване середовище розробки мови програмування Python, розроблена компанією JetBrains на основі IntelliJ IDEA, надає користувачеві комплекс засобів для написання коду та візуальний відладчик;

− Django: високорівневий фреймворк для розробки веб-застосунків на Python, надає готові рішення для багатьох аспектів веб-розробки, включаючи адміністративний інтерфейс, ORM для взаємодії з базою даних, систему маршрутизації, автентифікацію та інше;

− Flask є мікрофреймворком для розробки веб-застосунків, він має менше вбудованих функцій порівняно з Django, але це надає більшу свободу і гнучкість у виборі компонентів для вашого проекту;

− FastAPI: сучасний, швидкий (використовуючи статичну типізацію Python) фреймворк для розробки API на Python, підтримує автоматичну генерацію документації та використовує статичну типізацію для підвищення продуктивності та безпеки;

− Tornado: асинхронний фреймворк для розробки веб-застосунків, особливо підходить для реалізації асинхронних серверів та додатків, які вимагають високої продуктивності;

− Pyramid: фреймворк з відкритим кодом для розробки веб-застосунків, прагне бути гнучким та легким для використання.

Рис. 2.2 – Фреймворки для написання вебзастосунків

IDE PyCharm надає великий набір інструментів: вбудований відладчик та інструмент запуску тестів, профільник Python, повнофункціональний вбудований термінал, інструменти для роботи з базами даних. IDE інтегрована з популярними системами контролю версій, містить вбудований SSH-термінал, підтримує можливості віддаленої розробки та віддалені інтерпретатори, а також інтеграцію з Docker та Vagrant [22].

PyCharm надає широкі можливості налагодження коду на Python/Django та JavaScript: розставляє точки зупинки та задає умови їх спрацьовування прямо в редакторі, перевіряє контекстно-залежні локальні змінні та обумовлені користувачем watches, включаючи масиви та складні об'єкти, редагуйте значення на льоту. У режимі вбудованого налагодження значення змінних, параметрів функцій та інших об'єктів доступні у вікні редактора. Значення змінних можна переглянути прямо у вихідному коді поруч із їх використанням. Функція Step into My Code дозволяє в режимі налагодження здійснювати трасування лише за кодом проекту, не заглиблюючись у бібліотечні вихідні коди.

PyCharm вміє налагоджувати програми, які породжують кілька процесів Python, наприклад, програми Django, які не запускаються в режимі no-reload, або програми, які використовують інші веб-фреймворки, в яких реалізовано аналогічний підхід до автоматичного перезавантаження коду.

PyCharm підтримує два популярні профільники: yappi і cProfile. Робіть снепшоти та збирайте статистику роботи вашої програми прямо в PyCharm – результати відображаються у вигляді кольорового графа викликів функцій. Ви можете переглянути зведений звіт, також передбачена навігація з графа до коду.

При кожному виконанні скрипту/тесту або налагоджувача створюється спеціальна конфігурація запуску/налагодження, яку можна змінити та використовувати повторно. Як і налаштування проекту, конфігурацію Run/Debug можна використовувати спільно всією командою [23].

Підтримка Git, SVN, Mercurial, Perforce та інших систем контролю версій допомагає керувати локальними змінами та здійснювати складні операції з гілками. Всі одноманітні завдання (додавання та видалення файлів) виконуються автоматично. З PyCharm немає потреби залишати IDE під час розробки. Повнофункціональний термінал працює на платформах Windows, Linux і macOS.

2.3 Засоби розробки Backend та UI-інтерфейсу

Backend у термінах розробки вебзастосунків вказує на ту частину програми, яка відповідає за обробку запитів від клієнтської частини (frontend) та виконання логіки додатку на сервері. Бекенд складається з серверної частини архітектури вебзастосунку. Основні функції backend включають в себе:

− обробка запитів: отримує HTTP-запити від клієнтів і виконує відповідні дії. Це може включати обробку даних, виклик функцій, доступ до баз даних та інші операції;

− логіка додатку: містить код, який реалізує логіку додатку. Це може включати бізнес-логіку, обчислення, авторизацію, аутентифікацію та інші аспекти;

− взаємодія з базою даних: у веб-застосунках використовують бази даних для зберігання та управління даними. Backend забезпечує взаємодію з базою даних, виконуючи операції читання та запису;

− надсилання відповідей: відправляє відповіді клієнту у вигляді HTTP-відповідей, які можуть містити дані чи повідомлення про помилку;

– безпека: відповідає за забезпечення безпеки додатку, включаючи валідацію даних, контроль доступу та інші заходи безпеки.

Backend може бути написаний на різних мовах програмування, таких як Python, Java, Ruby, PHP, Node.js (JavaScript/TypeScript), інші. Вибір мови залежить від потреб проекту та власних вподобань розробників. Python є однією з популярних мов програмування для розробки backend частини веб-застосунків. Він володіє простим синтаксисом, широкою підтримкою та великою спільнотою розробників.

Фреймворк Flask для розробки backend веб-застосунків на мові програмування Python володіє великою популярністю завдяки своїй простоті та гнучкості. Його основним завданням є надання інфраструктури для створення ефективних та швидких веб-додатків. Flask визначається як мікрофреймворк, що вказує на його легкість та невеликий обсяг вбудованих компонентів. Фреймворк пропонує прозорий механізм маршрутизації, де URL-шляхи співставляються з функціями обробки запитів. Це дозволяє розробникам швидко реагувати на різні види HTTP-запитів. Flask також забезпечує інтеграцію з Jinja2 для шаблонізації HTML та інших форматів [24]. Це полегшує розділення логіки та представлення. Вбудований сервер Flask дозволяє вам швидко відлагоджувати та тестувати ваші додатки.

Однією з ключових особливостей Flask є його здатність до розширення. Розробники можуть вибирати різні розширення для роботи з базами даних, формами, автентифікацією та іншими завданнями. Іншою важливою характеристикою є відсутність жорсткої структури проекту, що дає розробникам свободу в організації свого коду та проектів. Узагальнюючи, Flask пропонує простий та ефективний спосіб розробки backend для веб-додатків, забезпечуючи при цьому гнучкість та широкі можливості налаштувань залежно від потреб.

Фронтенд (англ. frontend) – це термін, що використовується у веброзробці для позначення тієї частини програмного забезпечення, яка відповідає за інтерактивний та візуальний вивід інформації для користувача. Фронтенд описує інтерфейс

користувача (Інтерфейс користувача, англ. User Interface, UI), з яким користувач спілкується під час використання веб-застосунку чи веб-сайту. Ця частина програми відображається у браузері або іншому клієнтському програмному забезпеченні і представляє собою той інтерактивний шар, через який користувач може взаємодіяти з веб-сервісом. Метою UI є створення зручного, логічного та ефективного середовища для взаємодії користувача з програмою чи системою.

Фронтенд включає в себе ряд технологій та інструментів, таких як HTML (HyperText Markup Language), CSS (Cascading Style Sheets), і JavaScript [26]. HTML відповідає за структуру та семантику веб-сторінки, CSS визначає її вигляд і стилі, а JavaScript додає динамічну функціональність та інтерактивність. У веб-розробці, фронтенд тісно пов’язаний із бекендом, який відповідає за обробку бізнес-логіки та взаємодію із сервером. Разом вони створюють повноцінний вебзастосунок, де фронтенд забезпечує сприйнятливий та інтуїтивно зрозумілий інтерфейс, а бекенд виконує різноманітні операції з сервером. Фронтенд визначає користувацький досвід, його ефективність, зручність та естетичний вигляд веб-додатку. Важливою частиною розробки фронтенду є урахування принципів дизайну, доступності та реактивності, щоб забезпечити задоволення користувачів та ефективну взаємодію із вебсайтом чи застосунком.

Розмітка та стилізація інтерфейсу користувача в веб-застосунках вимагає використання HTML та CSS для ефективного подання та відображення контенту. HTML (HyperText Markup Language) визначає структуру сторінки, включаючи блоки, текст та мультимедіа, тоді як CSS (Cascading Style Sheets) відповідає за зовнішній вигляд та оформлення елементів. HTML використовується для розмітки структури сторінки, визначаючи заголовки, абзаци, списки та інші елементи. Елементи форм, такі як текстові поля та кнопки, також включаються в HTML для взаємодії з користувачем [27].

CSS використовується для визначення зовнішнього вигляду сторінки. Стилі включають у себе властивості, які визначають шрифти, розташування, колір, розміри та інші аспекти візуального оформлення. Каскадність (Cascading) в CSS

вказує на пріоритет визначених стилів, що дозволяє ефективно керувати оформленням елементів. Правильне використання HTML та CSS дозволяє створювати доступні, естетично приємні та користувацько-орієнтовані інтерфейси. Вони є важливими інструментами для веб-розробників, щоб забезпечити ефективну комунікацію із користувачами та покращити загальний дизайн та функціональність веб-застосунків.

JavaScript – це інтерпретована мова програмування, розроблена для інтерактивної взаємодії з веб-сторінками. У браузерах за замовчуванням вбудовано спеціальне програмне забезпечення, яке називається інтерпретатором JavaScript, це зроблено для того, щоб браузер міг виконувати написаний мовою JavaScript код. Як правило, JavaScript називають клієнтською мовою, підкреслюючи тим самим, що сценарій виконується на клієнтському комп'ютері у браузері, а не на веб-сервері. Можливості JavaScript:

− додавати різні ефекти анімації;

− реагувати на події - обробляти переміщення вказівника миші, натискання клавіш з клавіатури;

− здійснювати перевірку введення даних у поля форми до відправки на сервер, що знімає додаткове навантаження з сервера;

− створювати та зчитувати cookie, витягувати дані про комп'ютер відвідувача;

− визначати браузер;

− змінювати вміст HTML-елементів, додавати нові теги, змінювати стилі[28].

Цим, звичайно ж, список не обмежується, тому що крім перерахованого JavaScript дозволяє робити і багато іншого.

Бібліотека React – це бібліотека JavaScript, яка використовується для розробки інтерфейсів користувача веб-додатків (рис. 2.3). React визначається як декларативна, компонентна бібліотека, розроблена для покращення ефективності та модульності веб-розробки.

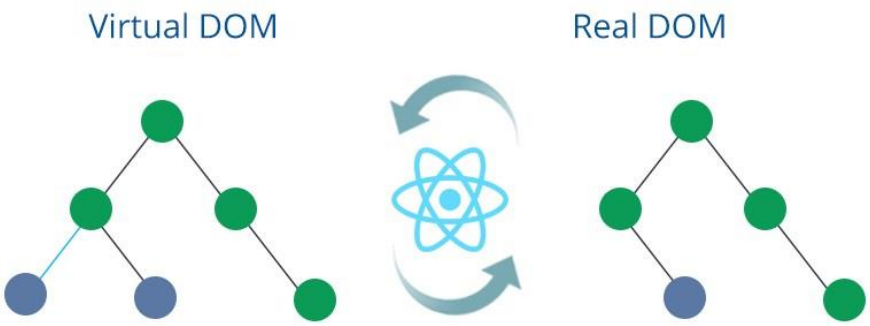

Рис. 2.3 – Приклад оновлення елементів за допомогою віртуального DOM

React використовує декларативний підхід до опису того, як виглядає інтерфейс користувача на різних етапах його стану. Це дозволяє розробникам фокусуватися на тому, що повинно відбуватися в інтерфейсі, а не яким чином досягти цього. React також побудований на концепції компонентів, що дозволяє виокремлювати частини інтерфейсу та перевикористовувати їх. Цей підхід сприяє модульності та підтримці чистого коду. Однією з ключових особливостей React є використання віртуального DOM (Document Object Model). Віртуальний DOM є ефективним механізмом для оптимізації оновлення елементів інтерфейсу (рис. 2.3). Замість безпосередньої модифікації реального DOM, React створює віртуальний DOM для відображення змін та ефективно оновлює реальний DOM лише тоді, коли це потрібно [29].

React використовує концепцію стану та властивостей (пропс) для управління даними та їх передачі між компонентами. Стан представляє собою внутрішній стан компоненту, який може змінюватися під час взаємодії з користувачем або подіями. Пропс використовується для передачі даних від одного компоненту до іншого. Це дозволяє створювати динамічні та інтерактивні інтерфейси. React визначається своєю ефективністю, декларативністю та компонентною архітектурою, що робить його популярним інструментом для розробки інтерфейсів користувача веб-додатків. Використання React дозволяє розробникам та інженерам покращувати продуктивність та обслуговувати складність сучасних веб-застосунків.

Висновки до розділу 2

При розробці системи для забезпечення вирішено застосувати мову програмування Python та середовищє розробки Jupyter Notebook. Бібліотеки Python Pandas, Matplotlib, Scikit-learn було використано для роботи з файлами даних, здійснення клстерного аналізу даних та оцінки його якості, візуалізації отриманих результатів. Для розробки функціоналу backend використовувався фреймворк Flask. JavaScript фреймворк React, HTML та CSS застосовувалися для розробки UI-інтерфейсу та взаємодії з Flask через API сервера.

3 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПОДІЛУ ГУМАНІТАРНОЇ ДОПОМОГИ

3.1 Основні етапи роботи системи

Основна частина функціональності системи полягає у аналізі набору даних із використанням алгоритмів кластерного аналізу з метою визначення найбільш уразливих груп людей, які потребують гуманітарної допомоги. У системі передбачено вибір для аналізу двох алгоритмів – алгоритму k-means та агломеративного алгоритму ієрархічної кластеризації. опишемо основні етапи розробленої системи.

На початку роботи передбачено введення початкових даних – кількості пакетів гуманітрної допомоги, які необхідно розподілити та завантаження набору даних у CSV-форматі з інформації про осіб, якіпотребують допомоги.

Після цього здійснюється вибір алгоритму: k-means чи алгоритму ієрархічної кластеризації.

Далі здійснюється попередня обробка та очистка набору даних, виявлення пропусків та дублікатів і їх обробка. Дублікати видаляються, пропущені значення замінюються на середні арифметичні значення ознаки.

На наступному етапі здійснюєтсья первинний статистичний аналіз даних, який включає описову статистику, перевірка на викиди та вивід теплової діаграми результатів кореляційного аналізу. У системі передбачена візуалізація результатів проведеного статистичного аналізу, яка включає побудову для кожної ознаки діаграм розмаху «ящик з вусами», гістограм та найбільш уразливих осіб по кожній з ознак набору даних.

Після зробленого первинного статистичного аналізу даних перед проведенням кластерного аналізу, який передбачається з використанням алгоритмів k-means та ієрархічного кластерного аналізу, для аналізу статистичної значущості наявності кластерів у наборі даних, було було здійснено розрахунок

статистки Хопкінса. Для приведення усіх ознак набору даних до одного числового діапазону від 0 до 1 передбачено їх нормалізацію. У якості метрики використовується відстань Евкліда.

У випадку реалізації алгоритму k-means для виявлення оптимальної кількості кластерів використовується метод ліктя та метод силуета. Для об’єднання кластерів на етапах ієрархічного алгоритму кластеризації використано метод середнього міжгрупового зв’язку.

Після здійснення кластеризації виводиться інформація про вміст кожного кластера, розрахунок середніх значень ознак кожного кластера та візуалізація результатів здійсненого аналізу. Для алгоритму ієрархічної кластеризації виводиться дентрограма. Для алгоритму k-means – графіки розподілу по кластерам у 2-х та 3-х мірному просторі ознак, що не є задовільним відображенням результатів у багатомірному просторі ознак, однак допомагає переглянути візуально результати розбиття набору даних на групи споріднених за ознаками осіб.

Далі проводиться інтерпретація результатів, визначення кластеру, який містить осіб, які найбільш потребують допомоги та визначення списку особі, які із цього кластера отримають допомогу.

Для забезпечення якості отриманого розподілу рекомендовано здійснити порівняння результатів розподілу, отриманого за обома алгоритмами. Співпадіння розподілу за обома алгоритмами на 80% і більше свідчить про якісний кластерний аналіз.

3.2 Програмна реалізація системи розподілу

Процес розробки Pyrhon-застосунку починається з завантаження необхідних для розробки бібліотек: NumPy, Pandas, Matplotlib, Scikit-learn (рис. 3.1).

Бібліотека NumPy використовується для роботи з масивами даних та векторизованими обчисленнями у Python. Pandas використовується для обробки

табличних даних та виконання операцій з даними, таких як фільтрація, групування, сортування тощо. Scikit-learn (sklearn) використовується для реалізації алгоритмів машинного навчання, таких як масштабування ознак, кластеризація, оцінки якості моделей тощо. KModes (kmodes) використовується для кластеризації методом KModes, який призначений для категоріальних даних.

SciPy використовується для наукових обчислень, зокрема для ієрархічної кластеризації даних та відображення дендрограм. Matplotlib.pyplot (plt) використовується для візуалізації даних, включаючи графіки, діаграми, дендрограми тощо. Seaborn (sns) використовується для покращення візуалізації даних, зокрема для статистичних графіків та теплових карт.

Далі здійснюється імпорт набіру даних для подальшої роботи з ним (рис. 3.2).

```
# Importing required packages
import numpy as np
import pandas as pd

import sklearn
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors

from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans

from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree

# For Visualization
import matplotlib.pyplot as plt

import matplotlib.cm as cm
%matplotlib inline
from pylab import rcParams
from kmodes.kmodes import KModes
import seaborn as sns
```

Рис. 3.1 – Підключення Pyrhon-бібліотек

```
import os
os.listdir()
df = pd.read_csv('updated_ukraine_data.csv')
df.head()
```

Рис. 3.2 – Імпорт DataSet

Набір даних містить сім атрибутів, які характеризують уразливість осіб населення, що перебувають у небезпечній прифронтовій зоні, та їх потребу у отриманні гуманітарної допомоги.

– surname: прізвище замовника гуманітарної допомоги;
– danger: рівень небезпеки за місцем проживання (від 0 до 100);
– childs: кількість дітей у сім’ї;
– health: загальні витрати на охорону здоров'я у % від загального рівня;
– income: чистий дохід на особу в сім’ї;
– tax: відсоток податків в залежності від доходу людини;
– age: скільки років людині;
– satis: умовна оцінка якості життя.

Загальна інформація про набір даних відображена на рисунку 3.3.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   surname  300 non-null    object
 1   danger   300 non-null    float64
 2   childs   300 non-null    float64
 3   health   300 non-null    float64
 4   income   300 non-null    float64
 5   tax      300 non-null    float64
 6   age      300 non-null    float64
 7   satis    300 non-null    float64
dtypes: float64(7), object(1)
memory usage: 18.9+ KB
```

Рис. 3.3 – Загальна інформація по набір даних

На першому етапі аналізу проводиться очищення даних шляхом видалення викидів та заміна пропущених значень на середні арифметичні значення ознаки, яка містила пропуски. Після цього проводиться первинний статистичний аналіз даних, перевірка на викиди та вивід теплової діаграми результатів кореляційного аналізу (рис. 3.4).

Для кожної ознаки було передачено побудову діаграм розмаху «ящик з вусами». Така діаграма зручно візуально відображає медіану, середню, мінімальне, максимальне значення, перший та третій квартилі та викиди (рис. 3.5). Соціально-економічні фактори та фактори охорони здоров'я, які визначають потребу у допомозі: danger, health та total_fer є важливими для остаточного розподілу гуманітарної допомоги, то ж вони були виділені помаранчевим кольором.

```
df.describe()
df1[cols].describe(percentiles= [0.01,0.25,0.5,0.75,0.99])
new_df = df.drop(columns=['surname'], inplace=True)
plt.figure(figsize= (12,8))
sns.heatmap(df.corr(), annot = True, cmap = "YlGnBu")
```

Рис. 3.4 – Код для описової статистики

```
fig, axs = plt.subplots(3,3, figsize = (15,12))
plt1 = sns.boxplot(df['danger'], ax = axs[0,0], color = 'orange')
plt2 = sns.boxplot(df['childs'], ax = axs[0,1])
plt3 = sns.boxplot(df['health'], ax = axs[0,2], color = 'orange')
plt4 = sns.boxplot(df['income'], ax = axs[1,0])
plt5 = sns.boxplot(df['tax'], ax = axs[1,1])
plt6 = sns.boxplot(df['age'], ax = axs[1,2])
plt7 = sns.boxplot(df['satis'], ax = axs[2,1], color = 'orange')
plt.show()
```

Рис. 3.5 – Код для побудови діаграм розкиду

У разі наявності викидів було введено обмеження їх верхньою та нижньою межею. Однак це може призвести до зміщення центроїда кластера. Тому, враховуючи всі можливі випадки, було обмежено екстремальні значення викидів

до 0,01 та 0,99 відсотків. Таким чином ризик перекриття кластерів буде мінімальним. Для ознак tax, health, total_fer викиди зі значеннями на вищому рівні прирівнюються до верхньої межі (0,99 процентиля).

На етапі попередньої обробки даних також передбачено визначення осіб, які мають найбільш уразливі значення по кожній з ознак окремо. Код для цього наведено на рисунку 3.6.

Код для виведення гістограм ознак набору даних наведено на рисунку 3.7.

```
fig, axs = plt.subplots(3,3,figsize = (15,15)) # poor top 5 surname represented as `pt10`
# danger
pt10_danger = df[['surname','danger']].sort_values('danger', ascending = False).head(5)
plt1 = sns.barplot(x='surname', y='danger', data= pt10_danger, ax = axs[0,0])
plt1.set(xlabel = '', ylabel= 'Danger')
# childs
pt10_childs = df[['surname','childs']].sort_values('childs', ascending = True).head(5)
plt2 = sns.barplot(x='surname', y='childs', data= pt10_childs, ax = axs[2,1])
plt2.set(xlabel = '', ylabel= 'childs')
# Health
pt10_health = df[['surname','health']].sort_values('health', ascending = False).head(5)
plt3 = sns.barplot(x='surname', y='health', data= pt10_health, ax = axs[1,0])
plt3.set(xlabel = '', ylabel= 'Health')
# income
pt10_income = df[['surname','income']].sort_values('income', ascending = False).head(5)
plt5 = sns.barplot(x='surname', y='income', data= pt10_income, ax = axs[1,2])
plt5.set(xlabel = '', ylabel= 'Income')
# tax
pt10_tax = df[['surname','tax']].sort_values('tax', ascending = False).head(5)
plt6 = sns.barplot(x='surname', y='tax', data= pt10_tax, ax = axs[2,0])
plt6.set(xlabel = '', ylabel= 'Tax')
# satis
pt10_satis = df[['surname','satis']].sort_values('satis', ascending = False).head(5)
plt7 = sns.barplot(x='surname', y='satis', data= pt10_satis, ax = axs[0,1])
plt7.set(xlabel = '', ylabel= 'satis')
# age
pt10_age = df[['surname','age']].sort_values('age', ascending = True).head(5)
plt8 = sns.barplot(x='surname', y='age', data= pt10_age, ax = axs[0,2])
plt8.set(xlabel = '', ylabel= 'age')
for ax in fig.axes:
    plt.sca(ax)
    plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
```

Рис. 3.6 – Код для визначення найбьільш уразливих значень по кожній з ознак набору даних

```
plt.figure(figsize=(15, 15))
features = ['danger', 'childs', 'health', 'income','tax', 'age', 'satis']
for i in enumerate(features):
    ax = plt.subplot(3, 3, i[0]+1)
    sns.distplot(df[i[1]])
    plt.xticks(rotation=20)
```

Рис. 3.7 – Код побудови гістограм ознак набору даних

Після здійснення статистичного аналізу даних у випадку, якищо дані представлено у різних числових діапазонах, здійснюється нормалізація кожної змінної до діапазону від 0 до 1 з використанням MinMax-нормалізації (рис. 3.8).

```
scaler = MinMaxScaler()
# fit_transform
df_scaled = scaler.fit_transform(df)
df_scaled.shape
```

Рис. 3.8 – Код номалізації ознак набору даних

Після зробленого первинного статистичного аналізу даних перед проведенням кластерного аналізу, який передбачається з використанням алгоритмів k-means та ієрархічного кластерного аналізу, для аналізу статистичної значущості наявності кластерів у наборі даних, було було здійснено розрахунок статистки Хопкінса за формулою 1.5 (рис. 3.9).

Для реалізації алгоритму k-means було використано функцію KMeans(). Для виявлення оптимальної кількості кластерів було використано метод ліктя та метод силуета (рис. 3.10, рис. 3.11). У якості метрики використовується відстань Евкліда.

```python
from random import sample
from numpy.random import uniform
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2, return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

```python
# Create dataframe of sclaled features
df_scaled = pd.DataFrame(df_scaled, columns = df.columns)

# Evaluate Hopkins Statistics
print('Hopkins statistics is: ', round(hopkins(df_scaled),2))
```

Рис. 3.9 – Код для перевірки статистики Хопкінса

```python
# pltng elbow method plot
plt.figure(figsize=(10,6))
plt.plot(num_clusers,ssd, marker = 'o')
plt.title('Elbow Method', fontsize = 16)
plt.xlabel('Number of clusters',fontsize=12)
plt.ylabel('Sum of Squared distance',fontsize=12)
plt.vlines(x=3, ymax=ssd[-1], ymin=ssd[0], colors="r", linestyles="-")
plt.hlines(y=ssd[2], xmax=9, xmin=1, colors="r", linestyles="--")

plt.show()
```

Рис. 3.10 – Код для застосування методу ліктя

```
# silhouette analysis
num_clusters = list(range(2,11))
ss = []
for cluster in num_clusters:

    # intialise kmeans
    kmeans = KMeans(n_clusters= cluster, max_iter=50)
    kmeans.fit(df_scaled)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = round(silhouette_score(df_scaled, cluster_labels),4)
    ss.append(silhouette_avg)
    print("For n_clusters={0}, the silhouette score is {1}".format(cluster, silhouette_avg))


plt.plot(num_clusters,pd.DataFrame(ss)[0])
plt.title('Silhouette Score', fontsize = 16)
plt.show()
```

Рис. 3.11 – Код для реалізації методу силуета

Для кожного об’єкту набору даних передбачено визначення та виведення індентифікатора належності до певного кластера (рис. 3.12). Передбачено виведення середніх значень кожного кластеру для інтерпретації даних та візуалізація отриманих результатів (рис. 3.13).

```
#adding produced labels dataframe
df_country = df.copy()
df_country['KMean_clusterid']= pd.Series(kmeans.labels_)
df_country.head()
```

Рис. 3.12 – Код для визначення ідентифікаторів належності об’єктів до певного кластеру та їх виведення

Для інтепретації результатів передбачено виведення середніх значень ознак кожного кластера та їх візуалізація у 2–х мірному просторі ознак (рис. 3.14, рис. 3.15).

```
plt.figure(figsize=(18, 5))
plt.subplot(1, 3, 1)
sns.scatterplot(x='health', y='danger', hue='KMean_clusterid', data=df_surname, palette="bright", alpha=.4)
plt.subplot(1, 3, 2)
sns.scatterplot(x='satis', y='health', hue='KMean_clusterid',data=df_surname, palette="bright", alpha=.4)
plt.subplot(1, 3, 3)
sns.scatterplot(x='danger', y='satis', hue='KMean_clusterid', data=df_surname, palette="bright", alpha=.4)
plt.show()
#3d Scatter plot on various variables to visualize the clusters based on them
fig = plt.figure(figsize=(18, 5))
ax1 = fig.add_subplot(131, projection='3d')
ax1.scatter(df_surname['health'], df_surname['danger'], df_surname['KMean_clusterid'], c=df_surname['KMean_clusterid'], cmap='viridis', alpha=0.4)
ax1.set_xlabel('Health')
ax1.set_ylabel('Danger')
ax1.set_zlabel('Cluster ID')

ax2 = fig.add_subplot(132, projection='3d')
ax2.scatter(df_surname['satis'], df_surname['health'], df_surname['KMean_clusterid'], c=df_surname['KMean_clusterid'], cmap='viridis', alpha=0.4)
ax2.set_xlabel('Satisfaction')
ax2.set_ylabel('Health')
ax2.set_zlabel('Cluster ID')

ax3 = fig.add_subplot(133, projection='3d')
ax3.scatter(df_surname['satis'], df_surname['danger'], df_surname['KMean_clusterid'], c=df_surname['KMean_clusterid'], cmap='viridis', alpha=0.4)
ax3.set_xlabel('Satisfaction')
ax3.set_ylabel('Danger')
ax3.set_zlabel('Cluster ID')
plt.show()
```

Рис. 3.13 – Код для візуалізації результатів кластеризації за алгоритмом k-means

```
df_surname = df.copy()
df_surname['KMean_clusterid']= pd.Series(kmeans.labels_)
df_surname.head()
# Cheking the cluter means
df_surname.groupby(['KMean_clusterid']).mean().sort_values(['danger','health','satis'],ascending = [False ,False,True])
```

Рис. 3.14 – Код для виведення середніх значень ознак кожного кластера

```
plt.figure(figsize=(18,5))
plt.subplot(1,3,1)
sns.barplot(x = 'KMean_clusterid', y = 'health', data=df_surname, palette="bright")
plt.title('health')

plt.subplot(1,3,2)
sns.barplot(x = 'KMean_clusterid', y = 'danger', data=df_surname, palette="bright")
plt.title('danger')

plt.subplot(1,3,3)
sns.barplot(x = 'KMean_clusterid', y = 'satis', data=df_surname, palette="bright")
plt.title('satis')

plt.tight_layout()

plt.show()
```

Рис. 3.15 – Код для візуалізації середніх значень ознак кластера

У системі також передбачено здійснення кластерного аналізу з використанням алгоритму ієрархічної кластеризації та виведення його результатів (рис. 3.16).

```
plt.figure(figsize = (18,8))
mergings = linkage(df_scaled, method="complete", metric='euclidean')
dendrogram(mergings)
plt.show()
cluster_labels = cut_tree(mergings, n_clusters=4).reshape(-1, )

cluster_labels
```

Рис. 3.16 – Код для здійснення ієрархічної кластеризації

Для розподілу зазначенної кількості гуманітарної допомоги між людьми використовується змінна, котру можна буде задавати як параметр. У випадку обмеженої кількості пакетів допомоги розподіл відбувається для осіб, які потрапили до кластера з найбільш вразливими категоріями людей у порядку, який визначається близькістю осіб до цетру кластера (рис. 3.17). Мірою близькості обрану відстань Евкліда.

```
HELP_PARAMETER = 20
cluster_0_indices = np.where(kmeans.labels_ == 0)[0]
# Визначемо відстань від кажної точки кластера "0" до його центра
distances_to_center_0 = euclidean_distances(df_scaled.iloc[cluster_0_indices], [cluster_centers[0]])

# знайдемо індекси 50 найближчіх точок до центру кластера "0"
closest_indices_to_center_0 = cluster_0_indices[distances_to_center_0.ravel().argsort()[:HELP_PARAMETER]]
df_scaled = df_scaled.join(df['surname'])
# виведемо індекси 50 найближчіх точок до центру кластера "0"
print(f'Top {HELP_PARAMETER} peoples dire need of aid based on K cluster are:', df_scaled.iloc[closest_indices_to_center_0]['surname'].values)
```

Рис. 3.17 – Код для визначення пріоритету надання гуманітарної допомоги

3.3 Розробка веб-інтерфейсу

Процес розробки вебінтерфейсу було виконано у середовищі розробки PyCharm, складові проєкту зазначено на рисунку 3.18. Для розгортання веб-додатку на сервері був використаний мікрофреймворк Flask, за допомогою нього вебзастосунок було розділено на три сторінки HTML. Стилістичне оформлення сторінок реалізовано за допомогою CSS.

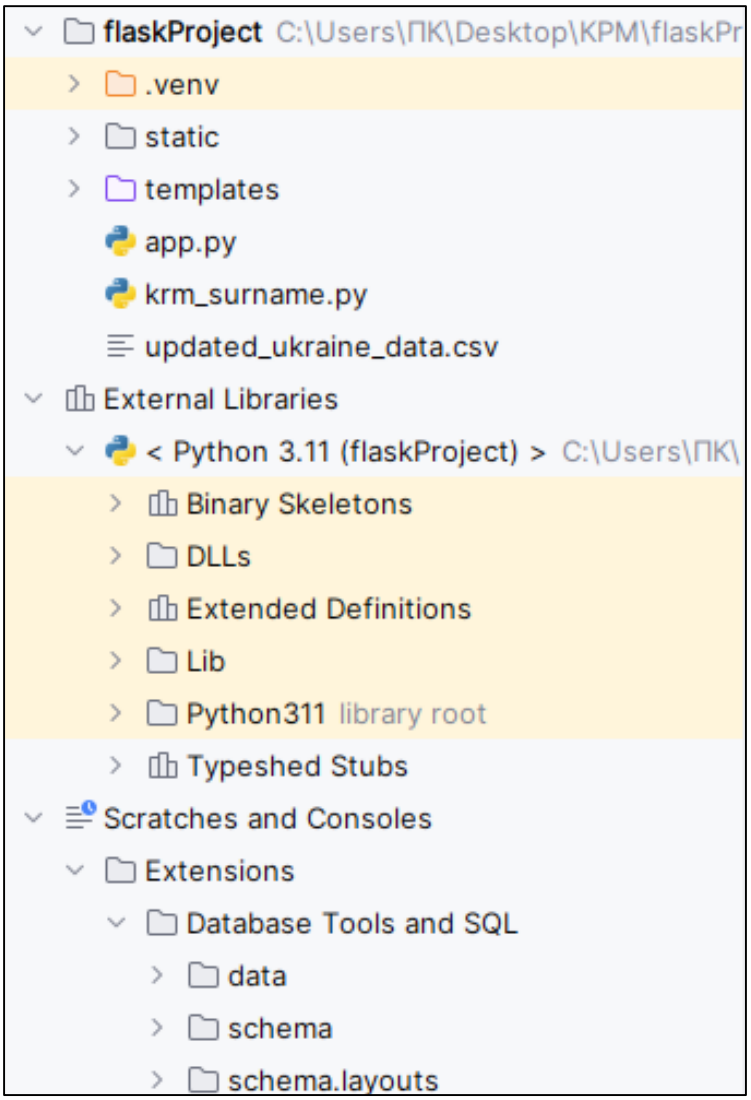

Рис. 3.18 – Компоненти проєкту

У файлі app.py проводиться імпорт необхідних модулів: Flask, render_template, request, jsonify. Також було виконано імпорт функції з файлу krm_surname.py для виконання аналізу даних. здійснено встановлення шляхів до статичних файлів, таких як зображення (.png) та шаблони HTML. Визначення маршрутів здійснювалося за допомогою декораторів @app.route ('/...'), які вказують, які дії повинні відбуватися при отриманні певного запиту на веб-сервері (рис 3.19):

– /: головна сторінка, яка відображає index.html та надає користувачу можливість обрати один із методів кластерного аналізу даних;

– /method1: сторінка для виконання інтелектуального аналізу за алгоритмом

к-середніх;

– /method2: сторінка для виконання інтелектуального аналізу за алгоритмом ієрархічного кластерного аналізу;

– /execute_code: маршрут для виконання коду після натискання кнопки на клієнтській стороні. Викликається функція execute_code для виконання аналізу даних з файлу krm_surname.py.

```python
from flask import Flask, render_template, request, jsonify
import importlib.util
from flask import url_for

app = Flask(__name__)

# Імпорт функції з файлу krm_surname.py
spec = importlib.util.spec_from_file_location( name: "krm_surname",  location: "krm_surname.py")
module = importlib.util.module_from_spec(spec)
spec.loader.exec_module(module)

@app.route('/')
def index():
    return render_template( template_name_or_list: 'index.html', background_image=url_for( endpoint: 'static'

@app.route('/method1')
def method1():
    return render_template( template_name_or_list: 'method1.html', plot_kmeans1=url_for( endpoint: 'static',

@app.route('/method2')
def method2():
    return render_template( template_name_or_list: 'method2.html', plot_hclust1=url_for( endpoint: 'static',

# Маршрут для виконання коду після натискання кнопки
@app.route( rule: '/execute_code', methods=['POST'])
def execute_code():
    # Виклик функції з файлу krm_surname.py
    module.execute_code()
    return 'Code executed successfully'

if __name__ == '__main__':
    app.run(debug=True)
```

Рис. 3.19 – Код імпорту модулів та визначення маршрутів по запитам серверу

Запуск вебзастосунку відбувається за допомогою методу app.run(), який запускає веб-сервер Flask.

3.4 Здійснення аналізу для визначення пріоритету у наданні гуманітарної допомоги

Для аналізу уразливості груп населення з метою визначення пріоритетів у розподілі гуманітарної допомоги було використано DataSet, що містив інформацію про 300 осіб та їх характеристики.

Для початку роботу на головній сторінці користувачу надана можливість обрати алгоритм аналізу, за которим буде розподілятися гуманітарна допомога: k-means чи алгоритим ієрархічної кластеризації (рис. 3.20). Для цього необхідно натиснути на відповідну кнопку головної сторінки.

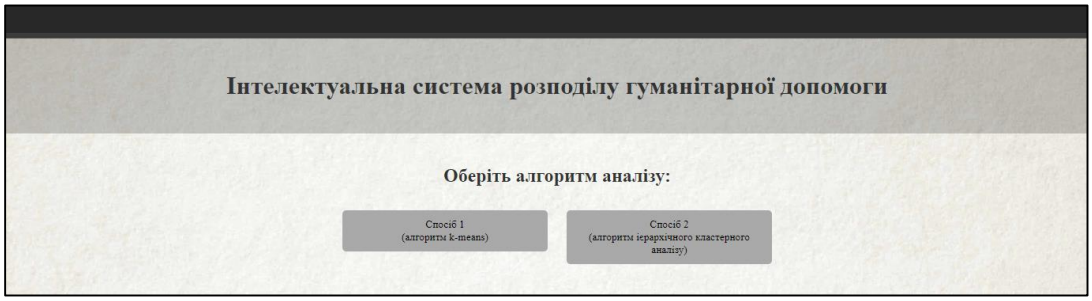

Рис. 3.20 – Вибір алгоритму для аналізу на головній сторінці

У випадку вибору одного із алгоритмів з’являється вінко, у якому користувач отримує можливість зазначити у відповідному полі, скільки пакетів гуманітарної допомоги необхідно розподілити між людьми. Для демонстрації роботи системи для прикладу введено 20 пакетів з гуманітарною допомогою.

Далі необхідно натиснути кнопку *Choos File* для завантаження файлу CSV-файлу з датасетом. Для початку аналізу введених даних з метою визначення пріоритету у потребі гуманітарної допомоги за обраним алгоритмом необхідно натиснути кнопку *Розподілити* (рис 3.21).

На рисунку 3.22 показано вибір файлу з набором даних у форматі CSV.

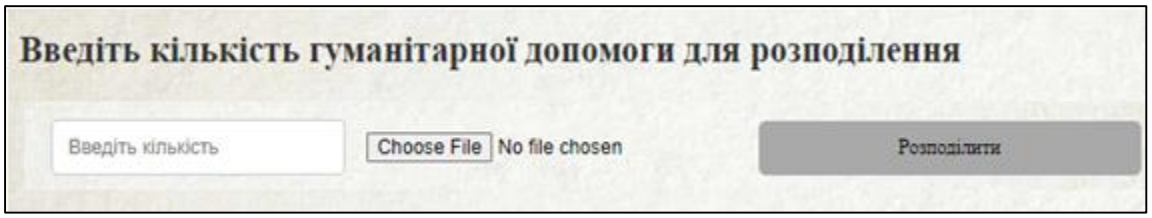

Рис. 3.21 – Введення даних для початку роботи системи

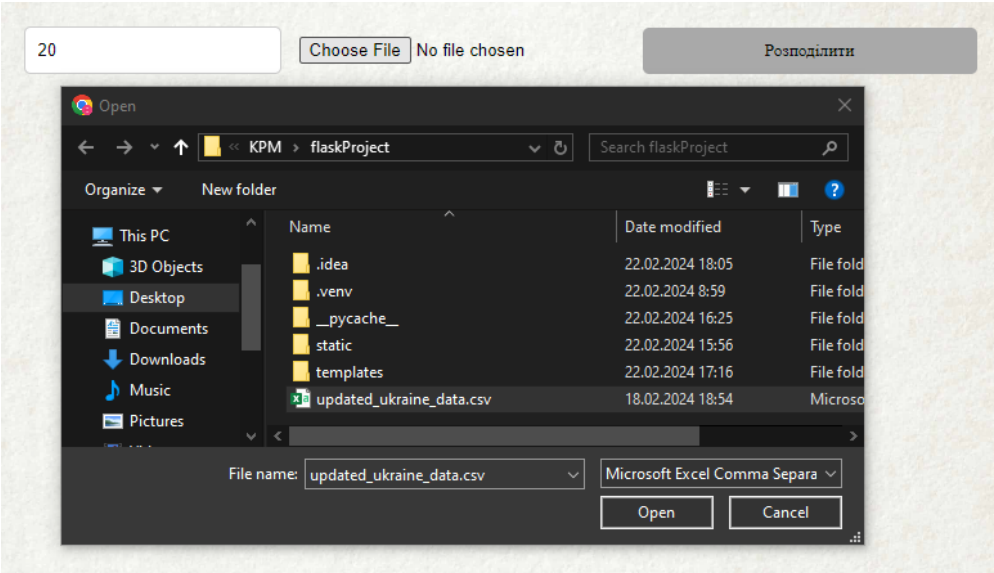

Рис. 3.22 – Вибір файлу з даними

Після завантаження набору даних у системі передбачено його виведення та перегляд (рис. 3.23).

	surname	danger	childs	health	income	tax	age	satis
0	Poliak	99.0	0.0	14.51	12326.0	10.00	56.0	3.8
1	Dmytrenko	58.0	2.0	2.50	16578.0	10.00	38.0	5.0
2	Kucher	17.0	0.0	1.45	21685.0	4.44	53.0	3.8
3	Korniychenko	81.0	1.0	6.82	13520.0	10.00	47.0	3.4
4	Kuzmenchuk	61.0	0.0	2.04	22096.0	4.22	75.0	3.0

Рис. 3.23 – Вхідний набір даних

Як бачимо, характеристики людей представлені у різних числових діапазонах, тому у подальшому було здійснено нормалізацію даних. Після цього виводиться інформація про відсутність у наборі даних дублікатів та пропусків (рис. 3.24). Описова статистика, здійснена на етапі первинного статистичного аналізу, представлена на рисунку 3.25.

```
В даному наборі даних ВІДСУТНІ дуплікати!
Перевірка на пропуски:
surname    False
danger     False
childs     False
health     False
income     False
tax        False
age        False
satis      False
dtype: bool
```

Рис. 3.24 – Результат перевірки на наявність пропусків та дублікатів

	danger	childs	health	income	tax	age	satis
count	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000	300.000000
mean	58.246667	0.796667	5.156900	20318.603333	8.171167	47.446667	4.173000
std	28.554574	1.116308	4.187814	9963.142393	3.879695	16.981216	1.633012
min	10.000000	0.000000	0.100000	7155.000000	3.020000	18.000000	1.600000
25%	35.750000	0.000000	1.567500	12303.250000	4.277500	32.000000	2.900000
50%	60.000000	0.000000	4.270000	17881.500000	10.000000	48.500000	4.000000
75%	85.250000	2.000000	7.660000	25035.750000	10.000000	62.000000	5.300000
max	100.000000	3.000000	14.900000	44211.000000	24.750000	75.000000	8.000000

Рис. 3.25 – Описова статистика набору даних

Гістограми кожної з ознак наведено на рисунку 3.26. Аналіз графіків свідчить про те, що більшість ознак розподілені не нормально. На рисунку 3.27 показано результат графічного зображення розкиду значень по кожній з ознак набору даних.

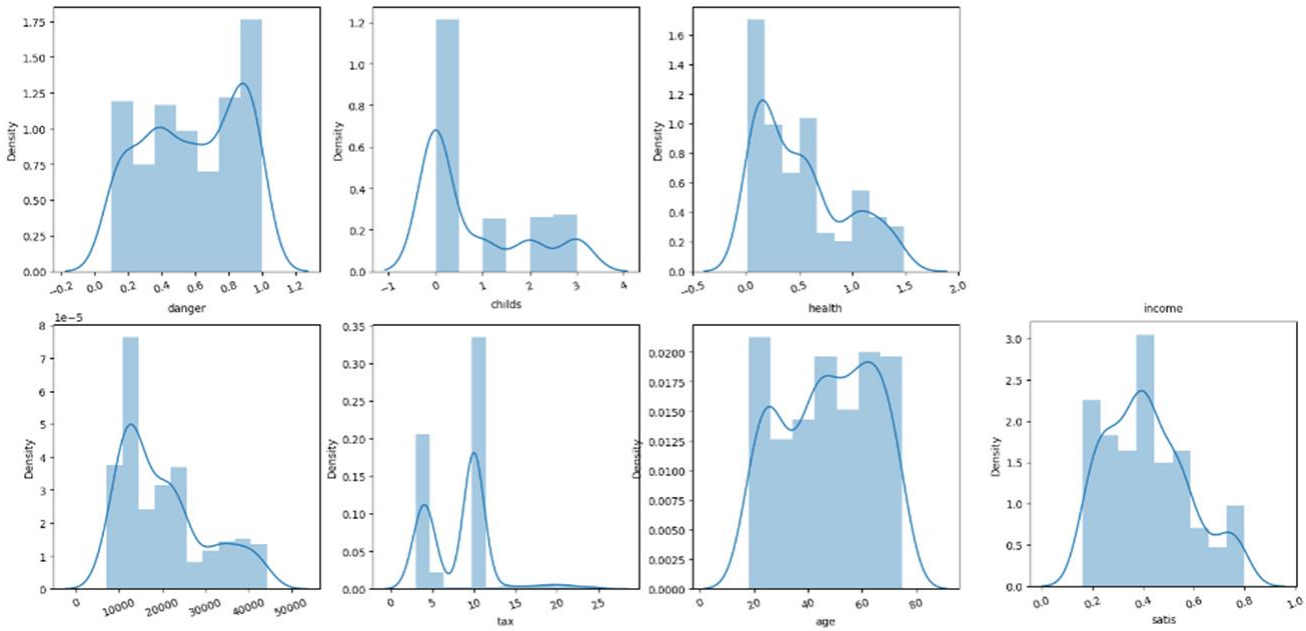

Рис. 3.26 – Гістограми ознак набору даних

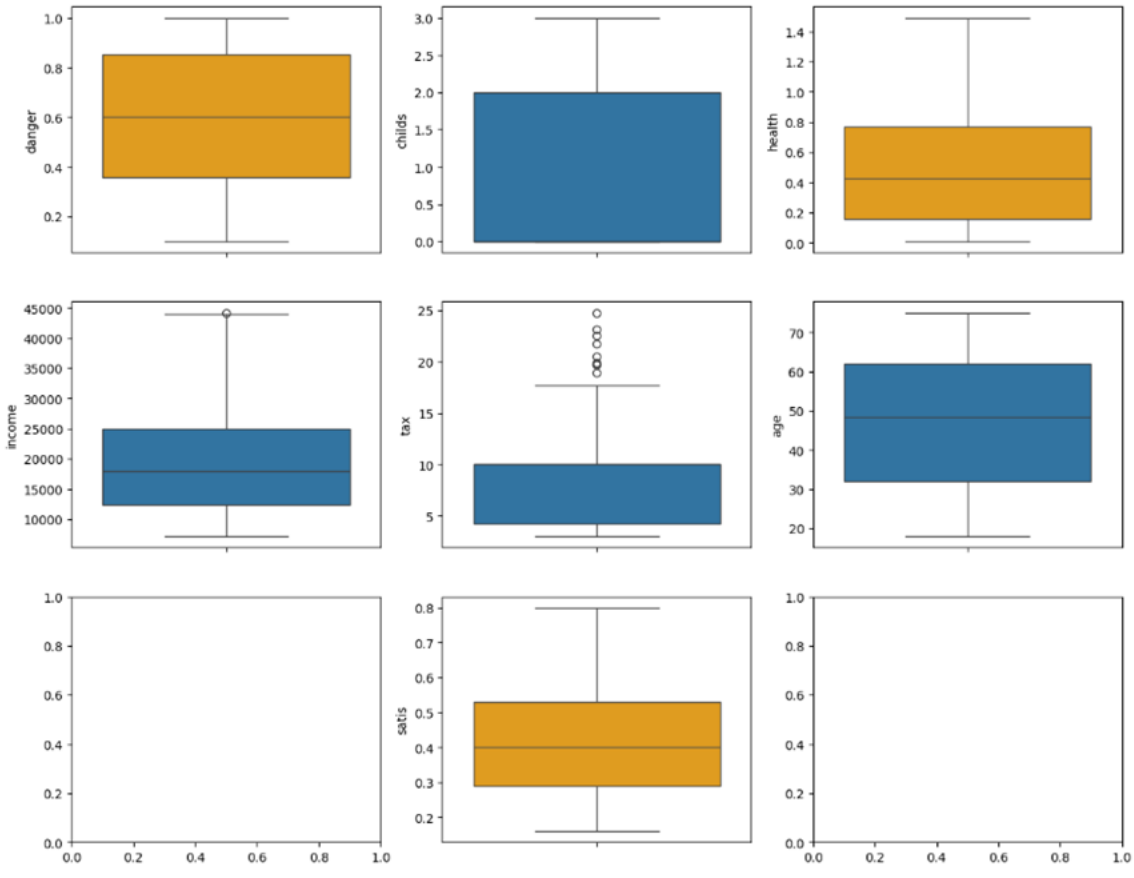

Рис. 3.27 – Діаграми розкиду для кожної ознаки набору даних

Ознаки health, total_fer мають викиди на вищому рівні. Імплементуємо викиди до верхньої межі (0,99 відсотків). Ознака life_exprec має викиди нижче нижньої межі, але є зацікавленість в цих значеннях, тому не будемо їх імплементувати.

На рисунку 3.28 виведено особи, які мають найбільш уразливі значення по кожній з ознак окремо.

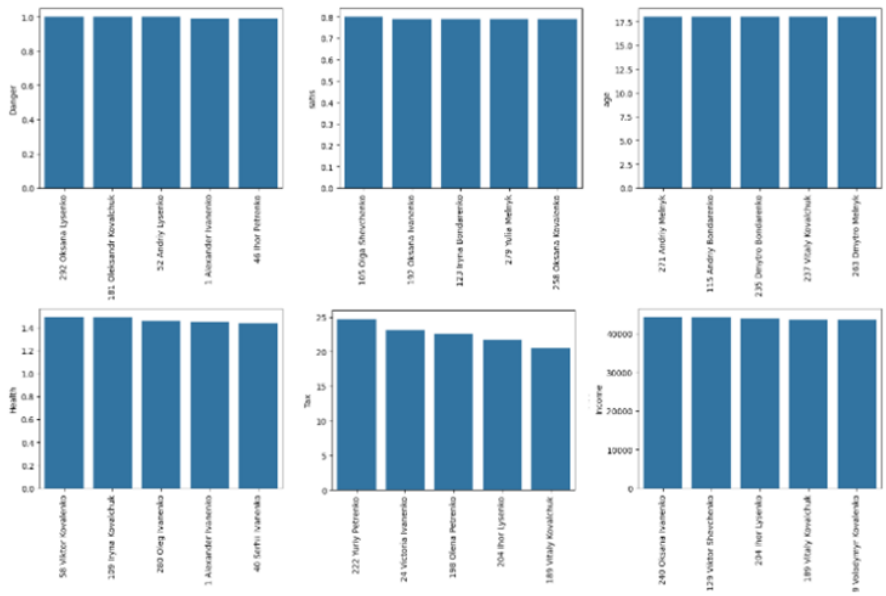

Рис. 2.28 – Виведення осіб, які мають найбільш уразливі значення по кожній з ознак окремо

Здійснення первинного статистичного аналізу показало, що дані різних ознак набору даних представлено у різних числових діапазонах, тому було здійснено нормалізацію значень кожної змінної до діапазону від 0 до 1.

Перш ніж застосовувати будь-який алгоритм кластеризації до даних, важливо перевірити, чи є в них значущі кластери чи ні, що загалом означає, що дані не є випадковими. Для перевірки тенденції до кластеризації використовується тест Хопкінса. Здійснений аналіз для аналізованого набору даних показав значення, рівне 0,3, що є хорошим показником, який вказує на те, що дані добре підходять для кластерного аналізу.

Для алгоритму k-means передбачено застосування методу ліктя та методу силуетів, яке показало, що оптимальна кількість кластерів рівна трьом (рис. 3.29). У системі передбачено виведення та перегляд вмісту кожного кластера для інтерпретації отриманих рехультатів (рис. 3.30). Розподілення по кластерам було наступним – 121, 102, 77 (рис. 3.31).

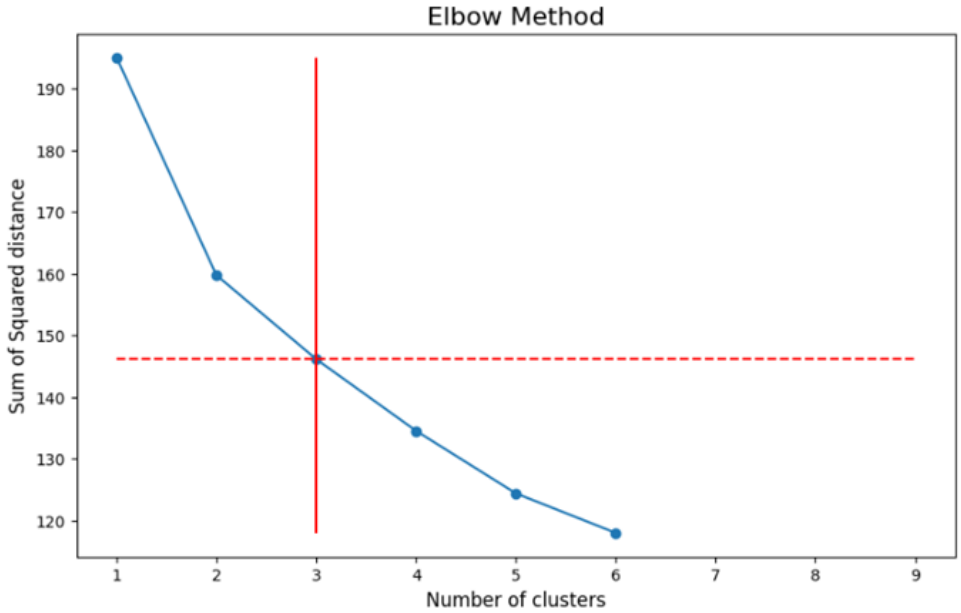

Рис. 3.29 – Використання методу ліктя для визначення оптимальної кількості кластерів

	danger	childs	health	income	tax	age	satis	KMean_clusterid
0	0.99	0.0	1.450	12326.0	10.00	56.0	0.38	0
1	0.58	2.0	0.250	16578.0	10.00	38.0	0.50	1
2	0.17	0.0	0.145	21685.0	4.44	53.0	0.38	2
3	0.81	1.0	0.682	13520.0	10.00	47.0	0.34	0
4	0.61	0.0	0.204	22096.0	4.22	75.0	0.30	0

Рис. 3.30 – Виведення ідентифікаторів належності до кластерів для осіб з вихідного набору даних

KMean_clusterid	
0	121
2	102
1	77

Рис. 3.31 – Розподілення осіб по кластерам

Візуалізація результатів кластерного аналізу за методом k-means у 2-х та 3-х мірному просторі ознак не є задовільним відображенням результатів у багатомірному просторі ознак, однак допомагає переглянути візуально результати розбиття набору даних на групи споріднених за ознаками осіб (рис. 3.32, 3.33).

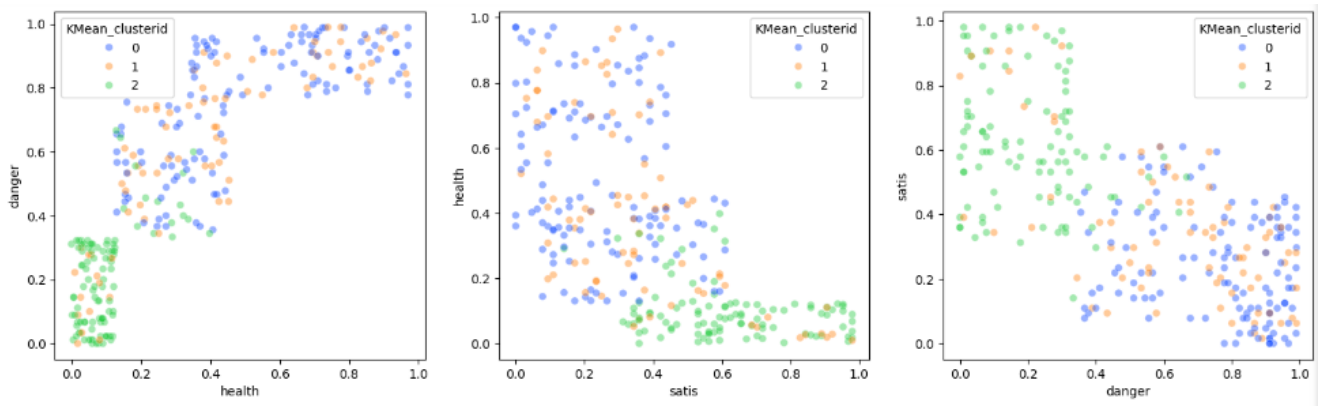

Рис. 3.32 – Візуалізація результатів кластеризації за алгоритмом у 2-х мірному просторі ознак

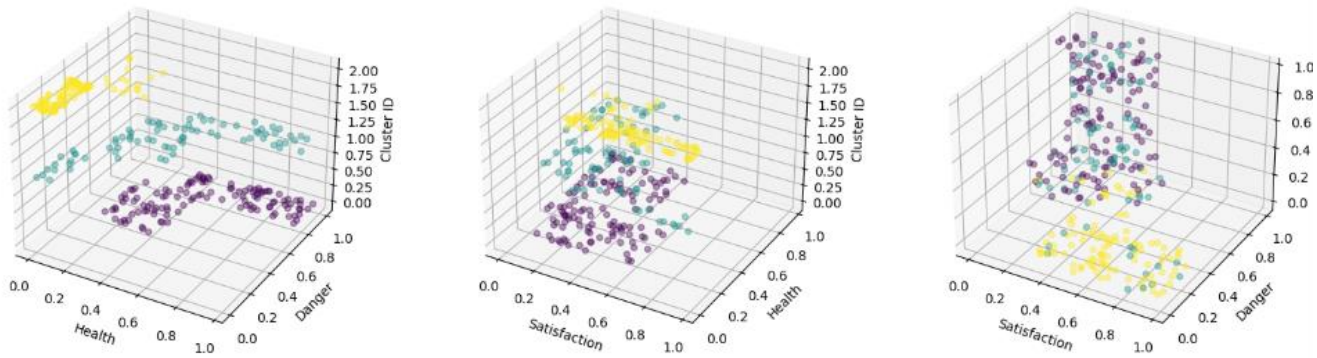

Рис. 3.33 – Візуалізація результатів кластеризації за алгоритмом у 3-х мірному просторі ознак

У випадку вибору для аналізу ієрархічного алгоритму кластеризації виводиться така ж інформація стосовно аналізованого набору даних та дендрограма, яка відображає етапи ітерації цього алгоритму (рис. 3.34).

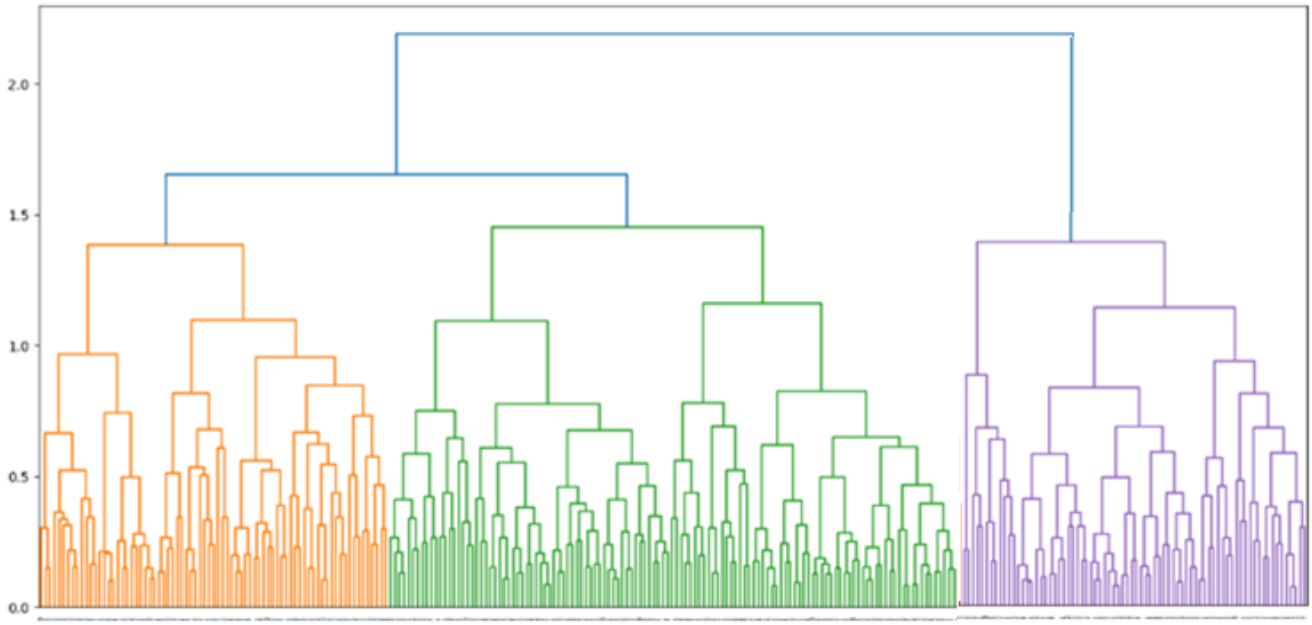

Рис. 3.34 – Гістограма проведеного ієрархічного кластерного аналізу

За результатами проведеного ієрарахічного кластерного аналізу також було отриману оптимальну кількість кластерів, рівну 3. Здійснений аналіз вмісту кластерів, отриманих за двома алгоритмами показав, що розподіл осіб по кластерам у них співпав на 84%. Що свідчить про непогану якість роботи алгоритмів.

Виведення середніх значень ознак кластерів дозволяє зробити інтерпретацію результатів з метою визначення того, особи якого кластера потребують першочергової допомоги (рис. 3.35, рис. 3.36). Здійснений аналіз дозволяє стверджувати, що найбільшу потребу у гуманітарній допомозі мають особи, які потрапили до першого кластера – там найнижчий рівень задоволеності життям, найвищі показники витрат та гірші показники здоров’я. На рисунку 3.37 показано список осіб, відібраних за алгоритмом k-means для надання гуманітрної допомоги.

KMean_clusterid	danger	childs	health	income	tax	age	satis
0	0.753820	0.052342	0.509981	0.167761	0.303343	0.593011	0.247934
1	0.627013	0.822511	0.400474	0.297555	0.267108	0.283436	0.357492
2	0.208824	0.098039	0.097006	0.620718	0.133332	0.601995	0.618088

Рис. 3.35 – Виведення середніх значень ознак кластера

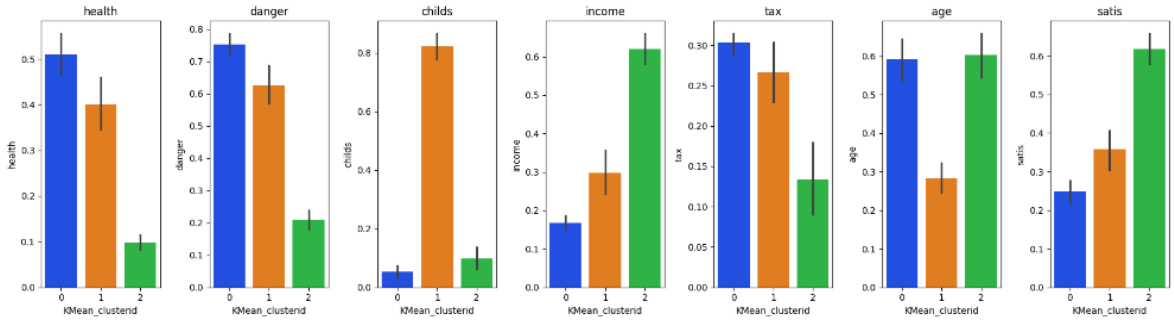

Рис. 3.36 – Візуалізація середніх значень ознак кластерів

surname	danger	childs	health	income	tax	age	satis
34 Iryna Kovalenko	80.0	0.0	7.4	12349.0	10.0	49.0	3.1
85 Oksana Kovalchuk	95.0	0.0	6.28	12204.0	10.0	52.0	3.8
298 Viktor Kovalenko	81.0	0.0	10.98	10079.0	10.0	53.0	2.8
65 Tatiana Shevchenko	76.0	0.0	4.61	14124.0	10.0	60.0	3.0
275 Julia Bondarenko	63.0	0.0	6.6	13946.0	10.0	61.0	2.6
247 Dmytro Melnyk	88.0	0.0	5.44	13981.0	10.0	45.0	4.1
154 Viktor Kovalenko	92.0	0.0	10.33	10844.0	10.0	53.0	2.6
116 Olga Lysenko	77.0	0.0	6.57	18190.0	10.0	48.0	4.7
4 Olena Bondarenko	81.0	1.0	6.82	13520.0	10.0	47.0	3.4
227 Julia Bondarenko	56.0	0.0	5.25	14223.0	10.0	56.0	2.8
11 Vitaly Petrenko	79.0	0.0	5.78	12021.0	10.0	67.0	3.8
217 Ihor Shevchenko	89.0	0.0	9.6	11326.0	10.0	57.0	1.7
139 Dmytro Bondarenko	97.0	0.0	9.47	13982.0	10.0	62.0	2.7
61 Vitaly Kovalchuk	86.0	0.0	10.02	11880.0	10.0	41.0	2.1
244 Oksana Lysenko	97.0	0.0	9.91	14412.0	10.0	52.0	2.0
164 Olga Lysenko	72.0	0.0	3.96	15033.0	10.0	42.0	3.9
143 Volodymyr Melnyk	61.0	0.0	4.93	18943.0	10.0	53.0	4.1
214 Oksana Petrenko	83.0	1.0	10.4	14734.0	10.0	50.0	3.3
223 Andriy Melnyk	94.0	0.0	5.44	12235.0	10.0	62.0	4.3
187 Dmytro Bondarenko	96.0	0.0	8.0	8859.0	10.0	57.0	1.7

Рис. 3.36 – Список осіб для отримання гуманітарної допомоги, визначений за алгоритмом k-means

Висновки до розділу 3

Здійснено розробку та програмну реалізацію системи розподілу гуманітарної допомоги, яка дозволяє удосконалити механізм визначення потреб у гуманітарній допомозі та оптимізувати розподіл необхідних ресурсів. Вебзастосунок надає можливість для волонтерів завантажувати дані стосовно осіб, які потребують допомоги, здійснювати їх аналіз, виявляти споріднені за потребами групи людей та формувати послідовність розподілу ресурсів у відповідності з установленим пріоритетом уразливості груп населення.

ВИСНОВКИ

У результаті проведеного дослідження виявлено, що бурхливі темпи інформатизації сучасного суспільства супроводжуються впровадженням цифрових технологій в усі сфери оточуючої дійсності та у сферу обліку і розподілу гуманітарної допомоги зокрема. Обсяги гуманітарної допомоги різко зросли в умовах ведення воєнних дій на території країни й охоплюють велику кількість волонтерів, громадських вітчизняних та міжнародних організацій і благодійних фондів. Різко зросла потреба оперативного надходження допомоги тим, хто її потребує. Вирішення цієї проблеми обумовлює необхідність створення інформаційних систем для покращення прогнозування попиту та керування ланцюгами постачання і розподілу гуманітарної допомоги.

Необхідність кластеризації виникає з потреби групувати схожі об’єкти чи ситуації для ефективного визначення та реагування на події. Кластеризація дозволяє виділити групи об’єктів зі спільними характеристиками, спрощуючи аналіз та оптимізуючи розподіл ресурсів. Здійснений аналіз дозволив виділити основні алгоритми, які доцільно задіяні для кластеризації – алгоритм ієрархічної кластеризації та алгоритми k-means і c-means. Визначено основні етапи даних алгоритмів, підходи до визначення якості кластеризації та оптимальної кількості кластерів: статистика Хопкінса, коефіцієнт силуета та метод ліктя.

Використання кластеризацієї сприяє більш точній і швидкій ідентифікації важливих патернів та забезпечує оптимальний розподіл гуманітарної допомоги в ситуаціях, коли час є критичним фактором. Створення подібної системи спрощує та економить час під час винесення рішення про розподіл гуманіторної допомоги, що дасть приріст продуктивності працівникам подібних організацій.

При розробці системи для забезпечення вирішено застосувати мову програмування Python та середовищє розробки Jupyter Notebook. Бібліотеки Python Pandas, Matplotlib, Scikit-learn було використано для роботи з файлами даних, здійснення клстерного аналізу даних та оцінки його якості, візуалізації отриманих

результатів. Для розробки функціоналу backend використовувався фреймворк Flask. JavaScript фреймворк React, HTML та CSS застосовувалися для розробки UI-інтерфейсу та взаємодії з Flask через API сервера.

Здійснено розробку, програмну реалізацію системи розподілу гуманітарної допомоги дозволяє удосконалити механізм визначення потреб у гуманітарній допомозі та оптимізувати розподіл необхідних ресурсів. Вебзастосунок надає можливість для волонтерів завантажувати дані стосовно осіб, які потребують допомоги, здійснювати їх аналіз, виявляти споріднені за потребами групи людей та формувати послідовність розподілу ресурсів у відповідності з установленим пріоритетом уразливості груп населення.

Поставлені завдання виконано повністю, однак функціонал розробленого застосунку може бути вдосконалений у подальшому шляхом врахування розширення предметів для гуманітарного допомоги та надання можливості онлайн доступу до розробленої системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. International Science Group. Theoretical foundations in practice and science. Bilbao, Spain. С. 148-152(дата звернення 02.01.2024).

2. Chakrabarti B. Why do Hurst Exponents of Traded Value Incriese as the Logaritm of Company Size. Econophysics of Stock and Other Markets: конспект лекцій. 2007. P. 50–55 (дата звернення 02.01.2024).

3. Бахрушин В. Є. Методи аналізу даних: навч. посіб. Vladimir Bakhrushin, 2011. С. 160-165(дата звернення 02.01.2024).

4. Журавльов О. О. "Розпізнавання; Математичні методи; Програмна система; Практичні застосування" навч. посіб. 2006. С. 74–76(дата звернення 02.01.2024).

5. Miyamoto S. Theory of Agglomerative Hierarchical Clustering. Singapore: Springer Singapore, 2022. P. 35-39. URL: https://doi.org/10.1007/978-981-19-0420-2 (дата звернення 03.01.2024).

6. Varmuza K., Filzmoser P. Introduction to Multivariate Statistical Analysis in Chemometrics: навчальний посібник. CRC Press. 336 p. (дата звернення 03.01.2024).

7. Software Engineering Trends and Techniques in Intelligent Systems / ed. by Pyotr Silkhavy et al. Springer International Publishing, 2017. 140-142 p(дата звернення 03.01.2024).

8. Організація Об'єднаних Націй. Humanitarian Data Exchange. https://data.humdata.org/. URL: https://data.humdata.org/ (дата звернення 03.01.2024)

9. Digital Humanitarian Network. https://digitalhumanitarians.com/. URL: https://digitalhumanitarians.com/ (дата звернення 04.01.2024).

10. Welcome to Python.org. Python.org. URL: https://www.python.org/ (дата звернення 04.01.2024)

11. Python Deep Learning Tutorial. Online Tutorials, Courses, and eBooks Library | Tutorialspoint. URL: https://www.tutorialspoint.com/python_deep_learning/index.htm (дата звернення 04.01.2024).

12. Creation of virtual environments. https://docs.python.org/3/library/venv.html. URL: https://docs.python.org/3/library/venv.html (дата звернення: 04.01.2024).

13. NumPy. https://numpy.org/doc/stable/release/1.26.0-notes.html. URL: https://numpy.org/doc/stable/release/1.26.0-notes.html (дата звернення 04.01.2024).

14. Chin L. (.-H., Dutta T. NumPy Essentials. Packt Publishing, Limited, 2016. 24-26 p.

15. Matplotlib – Visualization with Python. Matplotlib – Visualization with Python. URL: https://matplotlib.org/ (дата звернення 04.01.2024).

16. pandas - Python Data Analysis Library. pandas - Python Data Analysis Library. URL: https://pandas.pydata.org/ (дата звернення 05.01.2024).

17. Chen D. Pandas for Everyone: Python Data Analysis. Pearson Education, Limited, 2017.24 p(дата звернення 05.01.2024).

18. scikit-learn: machine learning in Python – scikit-learn 1.4.1 documentation. scikit-learn: machine learning in Python – scikit-learn 0.16.1 documentation. URL: https://scikit-learn.org/stable/ (дата звернення 05.01.2024).

19. Burns S. Python Deep learning: Develop your first Neural Network in Python Using TensorFlow, Keras, and PyTorch. Independently published, 2019. 19p (дата звернення 05.01.2024).

20. Project Jupyter. Project Jupyter | Home. URL: https://jupyter.org/ (дата звернення 05.01.2024).

21. Toomey D. Jupyter for Data Science: Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter. Packt Publishing - ebooks Account, 2017. 242 p. (дата звернення 06.01.2024)

22. JetBrains. PyCharm: the Python IDE for Professional Developers by JetBrains. JetBrains. URL: https://www.jetbrains.com/pycharm/ (дата звернення 06.01.2024)

23. Nguyen Q. Hands-On Application Development with Pycharm: Accelerate Your Python Applications Using Practical Coding Techniques in Pycharm. Packt Publishing, Limited, 2019. (дата звернення 06.01.2024)

24. Thanh N. FLASK Web Development: Developing Web Applications with Python. Independently Published, 2019. (дата звернення 07.01.2024)

25. Кобилін О.А., Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ.

26. Frain B. Responsive Web Design with HTML5 and CSS: Build Future-Proof Responsive Websites Using the Latest HTML5 and CSS Techniques. Packt Publishing, Limited, 2022. 293-295 p. (дата звернення 07.01.2024)

27. Learning TypeScript. Enhance Your Web Development Skills Using Type-Safe JavaScript. O'Reilly Media, 2022. 318 c

28. Гонсалес, Р., & Вудс, Р. (2019). Цифровая обработка изображений. Litres.

29. Как работает распознавание. URL: https://vc.ru/ml/96273-kak-rabotaet-raspoznavanie-rukopisnogo-teksta

30. Дакетт Джон HTML и CSS. Разработка и дизайн веб-сайтов (+ CD-ROM); Эксмо - М., 2013. 480 с

31. Python. URL: https://medium.com/@enduranceprog/machine-vision-digits-94eb258c6ff8

32. Т. М. Басюк, В. В. Литвин, Л. М. Захарія, Н. Е. Кунанець Машинне навчання: Навчальний посібник Львів: Видавництво «Новий Світ - 2000», 2021. - 315 с.

33. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.

34. Машинне навчання простими словами. Електронний ресурс. Код доступу: http://www.mmf.lnu.edu.ua/ar/1739

35. Matplotlib. URL: Matplotlib — Visualization with Python

36. Welcome to Python.org: вебсайт. URL: https://www.python.org/

37. NumPy: вебсайт. URL: https://numpy.org/

38. Matplotlib documentation. URL: Matplotlib documentation — Matplotlib 3.7.0 documentation

39. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language. O'Reilly Media, 2020. 706 с.

40. Amir S., Brenda J., Saurabh S. Designing Web APIs: Building APIs That Developers Love 1st Edition. O'Reilly Media, 2018. 232 с.

41. Фрейен Бен HTML5 и CSS3.Разработка сайтов для любых браузеров и устройств; Питер - Москва, 2014. 304 с.

42. Хантер Т., Инглиш Б. Многопоточный JavaScript; O'Reilly 2022. 188 с.

43. Мартин Фаулер. Рефакторинг кода на JavaScript. Улучшение проекта существующего кода; Діалектика, 2020. -464 с.

44. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media, Inc., 2014. – 832 с.

45. Eric A. Meyer, Estelle Weyl. CSS: The Definitive Guide: Visual Presentation for the Web. 4th Edition. O'Reilly Media, Inc., 2018. – 984 с.