МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра інформаційних систем
(повна назва кафедри)

Магістерська робота

РОЗРОБКА СИСТЕМИ КЛАСТЕРИЗАЦІЇ АНТИТІЛ НА ОСНОВІ КОЕФІЦІЄНТУ ПЕРЕХРЕСНОГО ЗВ'ЯЗУВАННЯ

Виконав: студент групи ПМІМ-22с
спеціальності
122 – комп'ютерні науки
(шифр і назва спеціальності)

308
(підпис) Зелінський О. А.
(прізвище та ініціали)

Керівник (підпис) Горлач В. М.
(прізвище та ініціали)

Рецензент (підпис) доц. Селіверстов Р.Г.
(прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет *прикладної математики та інформатики*

Кафедра *інформаційних систем*

Освітньо-кваліфікаційний рівень *магістр*

Спеціальність *122 – комп'ютерні науки*

«ЗАТВЕРДЖУЮ»

Зав. кафедрою *проф. Шинкаренко Г.А.*

« *5* » *вересня* 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Зелінський Олександр Анатолійович

(прізвище, ім'я, по батькові)

1. Тема роботи

Розробка системи кластеризації антитіл на основі коефіцієнту перехресного зв'язування

керівник роботи *доц. Горлач В.М.*

затверджена Вченою радою факультету від « *13* » *вересня* 20 *22* р., № *15*

2. Строк подання студентом роботи *12.12.2022 р.*

3. Вихідні дані до роботи

Література та інтернет-ресурси за тематикою роботи

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

1. Огляд сучасного стану проблеми

2. Модель даних. Обрані методи та технології

3. Програмна реалізація

4. Результати

5. Висновки та підсумки

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

Схеми, діаграми, скріншоти

Презентація дипломної роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *05.09.22 р.*

КАЛЕНДАРНИЙ ПЛАН

№	Найменування етапів дипломної (кваліфікаційної) роботи	Строк виконання етапів роботи	Примітка
1.	*Постановка задачі*	*Вересень*	
2.	*Огляд існуючих моделей та технологій*	*Вересень*	
3.	*Апробація алгоритмів кластеризації*	*Жовтень*	
4.	*Розробка веб застосунку*	*Жовтень*	
5.	*Розробка веб застосунку*	*Листопад*	
6.	*Апробація веб застосунку*	*Листопад*	
7.	*Оформлення роботи*	*Грудень*	

Студент _____ *Зелінський О.А.*
підпис

Керівник роботи _____ *доц. Горлач В.М.*
підпис

ЗМІСТ

ВСТУП

Епідемія Covid-19 показала, що людству досі досить важко контролювати гострі респіраторні вірусні інфекції та боротися з ними. За даними ВООЗ, майже 613 мільйонів людей у всьому світі були інфіковані Covid-19 і понад 6,5 мільйонів людей померли внаслідок цієї хвороби [1].

Однак загальновідомо, що це була не перша і, мабуть, не остання така пандемія. Тому вкрай важливо проводити дослідження якомога швидше, щоб захворювання можна було легко виявити та вилікувати. Наступний крок – розробка вакцин, а також тестів, які показують кількість антитіл до того чи іншого вірусу. Зрозуміло, що швидке виявлення захворювання допомагає ізолювати розповсюдження вірусу та ефективніше лікувати хворого, а вакцинація підвищує імунітет до конкретного вірусу та знижує ймовірність негативних (зокрема летальних) наслідків.

У наш час комп’ютери є дуже потужним інструментом, який дозволяє вирішувати не тільки математичні задачі, а й біологічні, хімічні та медичні. Для цього використовуються різні типи моделей і алгоритмів (включаючи алгоритми машинного навчання). Крім того, використання комп’ютерів допомагає вченим скоротити кількість експериментів і рутинної роботи в лабораторіях по всьому світу.

Мета цієї роботи – розглянути проблему пошуку двох оптимальних антитіл до будь-якого вірусу (наприклад, SARS-CoV-2) і запропонувати можливі шляхи її вирішення за допомогою алгоритмів машинного навчання, а точніше алгоритмів кластеризації.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

Вірус (від. лат. virus – отрута) – це не клітинні форми життя, які є внутрішньо клітинними абсолютними паразитами. Оскільки віруси вражають всі клітинні організми та спричиняють масові захворювання (епідемії) [2], людство розробило заходи боротьби з цими захворюваннями серед яких:

а) ізоляція хворих організмів від здорових (карантин);

б) лікування за допомогою хімічних антивірусних препаратів (хіміотерапія);

в) профілактичні щеплення для підвищення стійкості організму (імунізація).

В свою чергу людський організм має власну захисну систему – імунітет, складовими якого є лімфоцити, антитіла, тощо.

Антитіла (імуноглобуліни (Ig)) – це речовини, що утворюються в організмі людей внаслідок введення в нього сторонніх білків, бактерій, отрут та інших речовин, з метою знищити або нейтралізувати потенційну небезпеку [3, 4].

У задачі, що буде розглянута у цій магістерській роботі, дано молекулу протеїну (нуклеопротеїну), яка є загально визнаною ціллю для діагностики SARS-CoV-2 (не для вакцини), що використовується практично всіма швидкими тестами на антиген. До цієї молекули приєднуються два антитіла (вони можуть бути як різними так і однаковими), для того щоб можна було їх відрізнити одне з них помічається *. Ці два антитіла формують „сендвіч”, це стандартна схема визначення будь-якої білкової речовини.

Для простоти вважатимемо, що все відбувається на площині, антитіла це два круги однакового розміру, що мають невеликий „дзьоб” для взаємодії з вірусом. Антитіла в свою чергу приєднуються до меншого круга який представляє молекулу протеїну. Схематичне зображення цього процесу можна побачити на рисунку 1.1.

У нашому випадку конкуренція йде між антитілами з молярною вагою 180 кДа за зв’язування з вірусом з молярною вагою 45 кДа. З цієї причини відбувається запекла конкуренція.

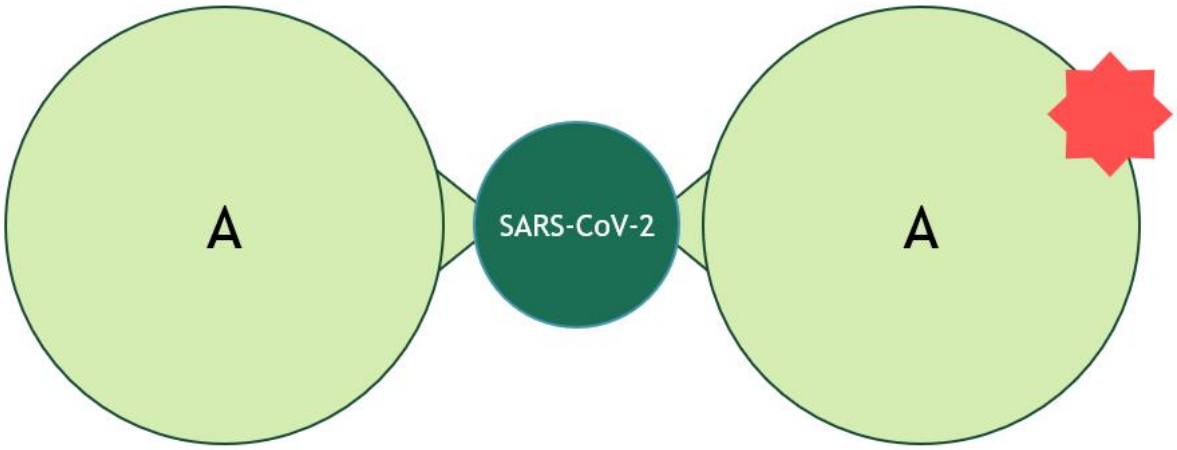

Рисунок 1.1 – Модель приєднання антитіл до вірусної молекули

Основне завдання полягає у знаходженні двох антитіл (можуть бути однаковими), таких що знаходяться на оптимальній відстані одне від одного тобто не перетинаються та не знаходяться занадто близько один до одного, щоб почати конкурувати між собою.

Зручним теоретичним методом вирішення цієї проблеми є розбиття списку антитіл на групи за ознакою того наскільки вони заважають один одному або іншими словами чи приєднуються вони до вірусу в одній і тій же області. Те що антитіла належать до різних груп означає, що вони прив’язуються в різних областях та взаємодіють краще ніж якби вони були з однієї.

В сучасних комп’ютерних науках популярним способом для розбиття даних на групи є група алгоритмів, яка називається кластерним аналізом.

Кластерний аналіз (англ. Data clustering) – задача розбиття заданої вибірки об'єктів (ситуацій) на підмножини, що називаються кластерами, так, щоб кожен кластер складався зі схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Метою кластерного аналізу є класифікація об’єктів на відносно гомогенні (однорідні) групи, виходячи із досліджуваної кількості ознак (показників, змінних). Об’єкти в групі є відносно подібними з огляду на досліджувані показники і відрізняються від об’єктів у інших групах [5]. Завдання

кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

Поняття кластеру тяжко однозначно визначити, тому існує доволі велика кількість алгоритмів кластеризації. Оскільки деякі алгоритми працюють зі схожими, або навіть однаковими об’єктами, які вони називають кластерами, то такі об’єкти було названо „кластерні моделі”, та поділено на групи, такі як:

а) моделі зв’язності;
б) центроїдні моделі;
в) статистичні моделі;
г) моделі засновані на щільності;
д) групові моделі;
е) графові моделі;
ж) нейронні моделі.

В даній роботі буде розглянуто перші дві моделі, а саме модель зв’язності для алгоритмів ієрархічної кластеризації та центроїдну модель для алгоритму k-means та його модифікації k-mods.

2 ОПИС ДАНИХ ТА АЛГОРИТМІВ

2.1 Опис даних

Дані для виконання роботи були надані компанією Xema OY, Лаппеенранта, Фінляндія. Для їх отримання був проведений експеримент, який складався з декількох кроків, серед яких:

а) Лабораторні дослідження (дослідження інгібування прямого зв'язування)

1) **Кон'югація HRP з моноклональними антитілами (МАТ)**. Для цього використовувався один з найпопулярніших методів кон'югації HRP (Faizyme, SAR) з антитілами, вперше описаний Nakane і Kawaoi в 1974 році [6]. Періодично окислений HRP утворює ковалентний зв'язок з МАТ після відновлення основи Шиффа бор гідридом натрію.

2) **Пряме зв'язування МАТ з варіантами N-Ag**. Препарати N-Ag розводили до 0,1 мкг/мл карбонатним буфером pH 9,5. Сто мікролітрів розчину вносили в лунки полістиролового мікропланшета з високою адсорбційною здатністю (KHB, Китай) та інкубували протягом ночі при температурі +4 °C. Після видалення вмісту мікролунок вакуумом, мікролунки одноразово відмивали промивним розчином для ІФА - 0,1% Tween 20 (Serva, Німеччина) в 0,9% хлориді натрію (Merck, Німеччина) і заливали блокуючим розчином для ІФА (0,1М фосфатний буфер, що містить 0,9% NaCl і 0,5% гідролізованого казеїну) на 2 години при кімнатній температурі, а потім висушували при кімнатній температурі протягом 48 годин. МАТ розводили буфером для ІФА (0,1М фосфатний буфер, що містить 0,9% NaCl і 0,1% гідролізованого казеїну) до рівномірної концентрації 1 мкг/мл. Сто мл розчину mAb інкубували в лунках протягом 30 хвилин при 37 °C. Лунки тричі промивали промивним розчином для ІФА і додавали HRP-кон'югований овечий антимишачий Ig-HRP кон'югат (Cat# AS302-HRP, Xema) в робочому розведенні в лунки ще

на 30 хвилин при 37 °C. Після 5 відмивань промивним розчином для ІФА в лунки вносили хромогенний субстрат ТМВ (Cat#R055, Xema) на 15 хвилин, реакцію зупиняли додаванням 5% сірчаної кислоти і вимірювали оптичну щільність при 450 нм (OD450) на зчитувачі для мікропланшетів HiPo (Biosan, Латвія).

б) **Перехресне інгібування МАТ шляхом прямого зв'язування з твердофазним N-Ag.**

На поверхню полістирольних лунок наносили повнорозмірний N-Ag у концентрації 0,5 мкг/мл (див. попередній параграф). У попередньому тесті кожен HRP-кон'югований МАТ послідовно розводили (у 10 разів) у мікролунках від 1:100 до 1:1 млн. та інкубували протягом 30 хвилин при 37 °C. Потім реакцію завершували відмиванням, додаванням субстрату ТМБ і стоп-розчину, як описано в попередньому пункті.

Для основного експерименту з перехресного інгібування як робоче використовували, розведення з коефіцієнт розведення кожного кон'югату, що дає OD450 в межах 1,0-1,5. Це відбувалось наступним чином: П'ятдесят мікролітрів робочого розведення кожного HRP-кон'югованого МАТ вносили в мікролунки з антигенним покриттям одночасно з рівним об'ємом ІФА-буфера (референтні лунки) або всі МАТ розводили до 10 мкг/мл в тому ж буфері. Після 30 хвилин інкубації при 37 °C реакцію завершували, як описано вище. Всі комбінації проганяли в двох екземплярах. Дані для кожної комбінації мічених і немічених HRP МАТ представлені у вигляді відсотка інгібування:

$$\frac{(\text{середня } OD450 \text{ фактичної комбінації} - \text{середня } OD450 \text{ референтних лунок})}{\text{середня } OD450 \text{ референтних лунок}} \quad (2.1)$$

Результати експерименту подані у вигляді таблиці (рисунок 2.1), де кожна комірка це коефіцієнт перехресного зв'язування міченого антитіла (зі стовпця) та немміченого (з рядка). В рядку позначеному як "blank" надані максимальні значення коефіцієнтів перехресного зв'язування для відповідного міченого антитіла.

Labelled	NP1501*	NP1502*	NP1503*	NP1508*	NP1510*	NP1514*	NP1516*	NP1517*	NP1518*	NP1520*	NP1521*
blank	1.089	1.067	1.3664	1.412	1.67	1.07	1.1704	1.11	1.1616	1.007	1.084
NP1501	0.449	0.715	1.0664	1.0248	1.136	0.26	0.4172	0.268	0.8512	0.614	0.596
NP1502	0.893	0.425	0.336	0.196	0.349	0.3625	0.665	0.45	1	0.188	0.535
NP1503	0.768	0.309	0.068	0.052	0.095	0.305	0.7126	0.408	0.9888	0.075	0.57
NP1507	0.422	0.856	0.7216	0.6088	0.785	0.1825	0.4256	0.154	1.0352	0.482	0.583
NP1508	0.732	0.388	0.1456	0.0848	0.19	0.345	0.8456	0.411	1.1344	0.144	0.55
NP1510	0.781	0.382	0.2152	0.1056	0.233	0.495	0.7826	0.527	1	0.227	0.661
NP1512	0.79	0.789	0.876	0.7504	0.979	0.735	1.015	0.737	0.816	0.853	0.638
NP1514	0.448	0.822	1.0968	1.0088	1.189	0.2475	0.4858	0.253	1.0208	1.04	0.626
NP1516	0.385	1.034	0.9832	0.9304	1.053	0.1775	0.3052	0.152	0.9504	0.782	0.455
NP1517	0.425	0.644	0.7952	0.7304	0.885	0.155	0.2758	0.079	1.1504	0.636	0.414
NP1518	0.517	0.636	1.0272	0.9424	1.107	0.2425	0.5502	0.34	0.2736	0.757	0.394
NP1520	0.669	0.503	0.0856	0.0536	0.106	0.315	0.5866	0.406	0.8352	0.095	0.431
NP1521	0.538	0.629	0.9264	0.9016	1.057	0.4425	0.6342	0.574	1.2048	0.851	0.107

Рисунок 2.1 Фрагмент початкових даних

Перед тим як застосовувати той чи інший алгоритм необхідно перетворити дані з таблиці за допомогою формули:

$$new\_cell_{i,j} = \frac{-(cell_{i,j} - blank_j)}{blank_j}, \tag{2.2}$$

Нові значення представляють відсоткове співвідношення між значенням у клітинці та максимальним значенням для відповідного стовпця. Нові значення знаходяться в діапазоні від 0 до 1.

Основна задача, тепер, полягає у розбитті отриманої матриці перехресного зв'язування антитіл на групи за ознакою подібності показника зв'язування для того, щоб полегшити виявлення оптимальних пар та приблизної локалізації місця зв'язування.

Роботу з отриманими даними можна розділити на два типи:

а) як з категоріальними змінними;

б) як з кількісними змінними.

2.2 Алгоритм роботи з категоріальними змінними

На початковому етапі роботи, для простоти, було прийнято, що дані в кожній комірці поділяються на три категорії за наступним алгоритмом:

а) цифрою 3 (темно-зеленим кольором) помічаються антитіла з поганим зв'язуванням, якщо

$$new\_cell_{i,j} > 0.75 \tag{2.3}$$

б) цифрою 2 (світло-зеленим кольором) помічаються антитіла з середнім зв'язуванням, якщо

$$0.5 < new\_cell_{i,j} \leq 0.75 \quad (2.4)$$

в) цифрою 1 (білим кольором) помічаються антитіла з хорошим зв'язування – у всіх інших випадках.

label	NP1501*	NP1502*	NP1503*	NP1508*	NP1510*	NP1514*	NP1516*	NP1517*
NP1501	2	1	1	1	1	3	2	3
NP1502	1	2	3	3	3	2	1	2
NP1503	1	2	3	3	3	2	1	2
NP1507	2	1	1	2	2	3	2	3
NP1508	1	2	3	3	3	2	1	2
NP1510	1	2	3	3	3	2	1	2
NP1512	1	1	1	1	1	1	1	1
NP1514	2	1	1	1	1	3	2	3
NP1516	2	1	1	1	1	3	2	3
NP1517	2	1	1	1	1	3	3	3

Рисунок 2.2. Фрагмент позначених даних

В даній роботі для категоріальними змінних використовується метод кластеризації k-modes, який є модифікація методу k-середніх.

k-середніх – це популярний алгоритм кластеризації на основі центроїдів, який розділяє дані представлені у вигляді точок на k кластерів, кожен із яких має майже рівну кількість цих точок. Ідея цього алгоритму кластеризації полягає в тому, щоб знайти k центроїд, де кожна точка з набору даних буде належати будь-якій з k-множин з найближчим (найчастіше мінімальна евклідова відстань) до нього середнім значенням.

k-modes метод, розроблений Хуагном в 1998 році [7], що визначає кластери на основі відповідності категорій між точками даних. В цьому алгоритмі на відміну від k-середніх відстань між двома точками (проста міра не схожості) даних X та Y описується як сума не схожих елементів:

$$d(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i), \quad (2.5)$$

де

$$\delta(x_i, y_i) = \begin{cases} 0, & x_i = y_i \\ 1, & x_i \neq y_i \end{cases}. \quad (2.6)$$

Також, середнє змінюється на моду, тобто тепер елемент до кластера будуть відносити не тоді, коли цей кластер має середнє значення, що найближче до цього елемента, а коли мода цього кластера буде найближчою до елемента. Моди відповідно знаходиться на основі частоти появи значення в кластері (приклад наведено в таблиці 2.1).

Таблиця 2.1 Приклад знаходження центроїдів в алгоритмі k-mods

	Кластер 1	Кластер 2	Кластер 3	Кластер 4	Кластер 5
Елемент 1	1	0	1	1	1
Елемент 2	1	0	0	1	2
Елемент 3	1	1	0	1	2
Елемент 4	1	1	0	0	2
Елемент 5	1	0	0	0	1
Центроїди	1	0	0	1	2

Нижче наведено кроки алгоритму кластеризації на основі k-modes:

а) вибір k початкових мод;

б) розділення елементів на кластери на основі простої міри несхожості відносно початкових мод. Після додавання нового елементу відбувається оновлення мод кожного з кластерів;

в) після того, як усі елементи були віднесені до кластера, відбувається перевірка значення несхожості кожного спостереження з модою. Якщо виявляється, що найближча мода знаходиться в іншому кластері, відбувається переміщення елемента у відповідний кластер і оновлення мод обох кластерів;

г) повторення кроку 3, доки жоден із елементів не змінить кластер на інший.

Метод k-modes працює лише при початкові відомій кількості кластерів, оскільки вона нам не відома, то для визначення оптимальної кількості кластерів використовується так званий „ліктьовий метод” (elbow method). Даний метод

модифікований так, щоб використовувати відмінність всередині кластера (within-cluster difference) як вартість.

$$WCD = \sum_{j=1}^{k} \sum_{i=1}^{m} d(x_i, y_c), \tag{2.7}$$

де WCD – відмінність всередині кластера, k – кількість кластерів, m – кількість спостережень у кожному кластері, c – центроїд кластера, а d – проста міра несхожості.

Суть методу в тому щоб, для різних значень k вибирати значення k у тій точці, де значення WCD істотно не зменшуватиметься зі збільшенням значення k [8].

У якості початкового розбиття на кластери використовується алгоритм розроблений Fuyuan Cao в 2009 році. Цей метод ініціалізації для категоріальних даних, у якому враховують відстань між об’єктами та щільність об’єкта, що визначається на основі частоти значень атрибутів [9].

2.3 Алгоритм роботи з кількісними змінними

Оскільки нові значення представляють відсоткове співвідношення між значенням у клітинці та максимальним значенням для відповідного стовпця, то їх можна розглядати як кількісні змінні.

Для кластеризації кількісних змінних було вирішено застосувати один із найпопулярніших видів кластеризації – ієрархічні алгоритми, а саме їх агломеративний підвид. Використання агломеративного підвиду („знизу вгору”) означає, що при побудові кластерів спочатку кожна точка даних належить власному кластеру, і з кожним кроком кластери об’єднуються в більші доки всі точки даних не опиняться в одному кластері. На рисунку 2.3 зображено приклад виконання алгоритму. В результаті уварюється ієрархія, яку зображають з використанням дендрограми (рисунок 2.4).

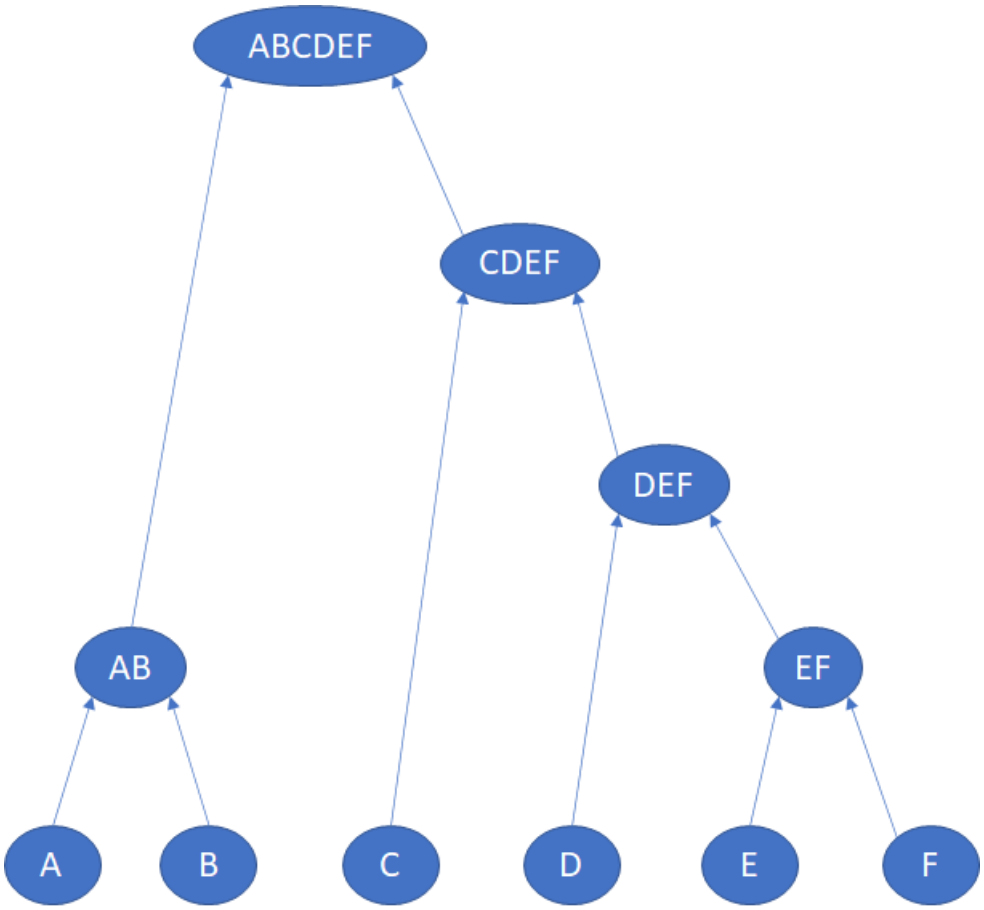

Рисунок 2.3. Приклад агломератної фільтрації

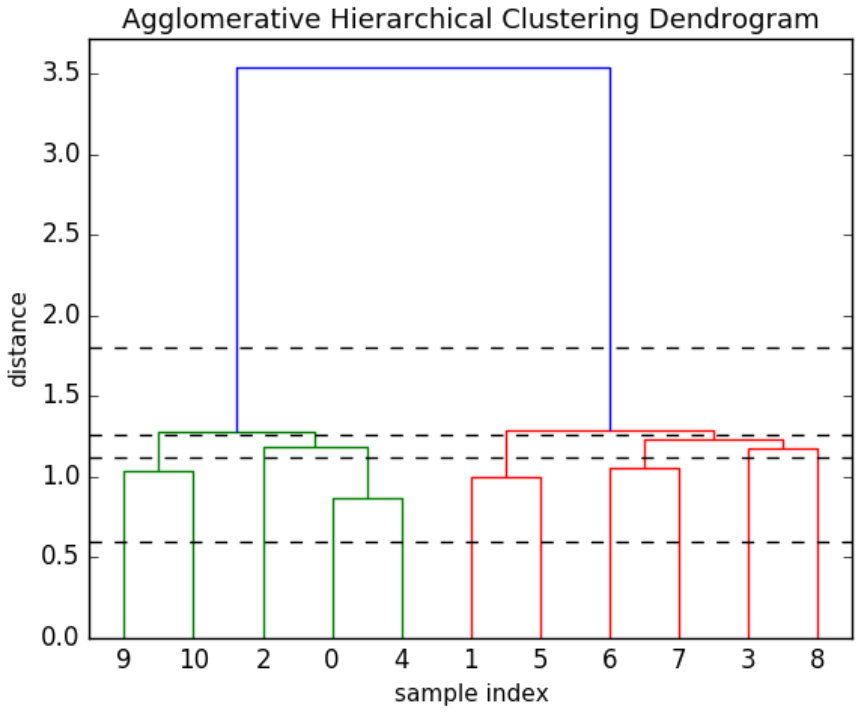

Рисунок 2.4 Приклад дендрограми

Також було обрано декілька стратегій зв’язування [10, 11]:

а) метод Уорда (Ward linkage) – на скільки збільшуються дисперсії для кластерів, які об’єднуються;

$$\frac{|A| * |B|}{|A \cup B|}\|\mu_A - \mu_B\|^2 = \sum_{x \in A \cup B}\|x - \mu_{A \cup B}\|^2 - \sum_{x \in A}\|x - \mu_A\|^2 - \sum_{x \in B}\|x - \mu_B\|^2 \quad (2.7)$$

де μ_A, μ_B – центроїди A та B відповідно.

б) метод повного зв’язку (Complete linkage) – максимальна відстань між елементами кожного кластера;

$$\max_{a \in A, b \in B} d(a, b) \quad (2.8)$$

в) метод середнього зв’язку (Average linkage) – середня відстань між елементами кожного кластера;

$$\frac{1}{|A| * |B|}\sum_{a \in A}\sum_{b \in B} d(a, b) \quad (2.9)$$

г) метод одиночного зв’язку (Single linkage) – мінімальна відстань між елементами кожного кластера;

$$\min_{a \in A, b \in B} d(a, b) \quad (2.10)$$

У всіх вище наведених $d(a, b)$ це метрика на основі якої визначають відстані між кластерами у цій роботі, було вирішено використовувати найпростішу метрику – Евклідову відстань.

$$d(a, b) = \sqrt{\sum_i (a_i - b_i)^2}. \quad (2.11)$$

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

В результаті виконання магістерської роботи було розроблено веб-застосунок, який складається з трьох основних частин:

Інтерфейс користувача, написаний на мові програмування TypeScript з використанням бібліотек React та Ant Design, реалізує функції:

а) вивантаження файлу з початковими даними;

б) перегляд результатів;

в) завантаження файлу з результатами у форматі .xlsx.

Прикладний програмний інтерфейс (API), який відповідає за кластеризацію написаний на мові програмування Python 3. Для завантаження та обробки даних використовувалась бібліотека pandas, для візуалізації бібліотека matplotlib, для кластеризації Scikit-learn, а саме агломеративну кластеризацію зі стратегією зв’язку, що реалізована методом Уорда, та kneed для знаходження оптимальної кількості кластерів.

Також реалізовано прикладний програмний інтерфейс, що поєднує між собою два попередні пункти та реалізує перетворення даних для відображення та збереження результатів. Цей інтерфейс написаний на мові програмування C# з використанням .NET 5 та ASP.NET Core 5.0.

Архітектура веб застосунку зображена на рисунку 3.1.

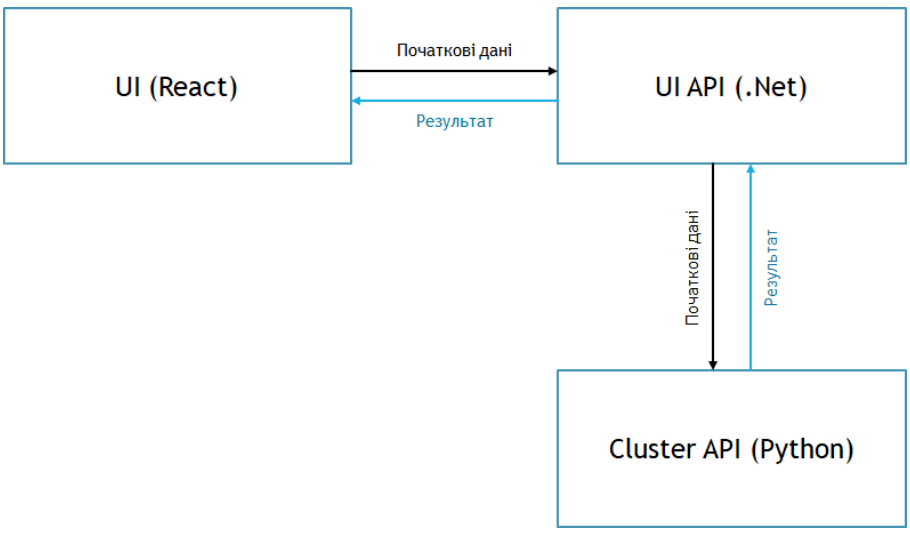

Рисунок 3.1 Архітектура веб застосунку

Користувач завантажує файл з початковими даними, які відправляються на UI API, який в свою чергу відправляє їх на Cluster API, де дані обробляються та відбувається процес кластеризації. Після цього результат відправляється на UI API де конвертується у зручну для відображення форму та віддається на інтерфейс користувача. Повний код UI API та інтерфейсу користувача наведений в додатках А та Б відповідно.

Для експорту результатів дані з UI знову відправляються на UI API, який у свою чергу формує та повертає файл у форматі .xlsx. Для експорту використано бібліотеку з відкритим вихідним кодом, під назвою SpreadsheetLight. Повний код функції для експорту файлу наведений в Додатку В.

Також було реалізовано програму мовою Python для тестування та порівняння різних методів кластеризації. Вона складається з двох частин, перша відповідає за кластеризацію методом k-modes.

Основна функція цієї частини приймає на вхід дані для обрахунку та складається з таких складових:

а) кластеризація на все можливу кількість кластерів (від 1 до кількості точок даних);

б) знаходження оптимальної кількості кластерів;

в) кластеризація для оптимальної кількості кластерів.

Для кластеризації використовувалась бібліотека kmodes та відповідний метод KModes з неї (фрагмент програми наведено на рисунках 3.2 та 3.3).

```
def setup_kmodes(num_clusters: int, data: pd.DataFrame) -> KModes:
    """..."""
    return KModes(n_clusters=num_clusters, init="Cao", n_init=5, verbose=verbose)
```

Рисунок 3.2 Створення класу KModes

```python
def get_clusters_for_optimal_model(data: pd.DataFrame, cluster_amount: int) -> npt.NDArray[np.uint16]:
    """..."""
    kmodes = setup_kmodes(cluster_amount, data)
    clusters = kmodes.fit_predict(data)
    return clusters
```

Рисунок 3.3 Функція кластеризації з заданою кількістю кластерів

Після того як проведено кластеризацію оптимальна кількість кластерів (фрагмент програми наведено на рисунках 3.4 та 3.5), зважаючи на яку відбувається повторне розбиття на кластери (рисунок 3.3). Для цього використовується KneeLocator з бібліотеки kneed.

```python
def build_elbow_curve(data: pd.DataFrame) -> tuple[range, list[int]]:
    """..."""
    cost = []
    cluster_amount_range = range(1, data.shape[0])
    for num_clusters in list(cluster_amount_range):
        kmodes = setup_kmodes(num_clusters, data)
        kmodes.fit_predict(data)
        cost.append(kmodes.cost_)

    if verbose == 1:
        build_elbow_plot(cluster_amount_range, cost)
    return cluster_amount_range, cost


def build_elbow_plot(cluster_amount_range: range, cost: list[int]):
    """..."""
    plt.plot(cluster_amount_range, cost, 'o-', color="blue", markerfacecolor='red', markeredgecolor='red')
    plt.xlabel('No. of clusters')
    plt.ylabel('Cost')
    plt.title('Elbow Method For Optimal k')
    plt.show()
```

Рисунок 3.4 Ліктьовий метод

```python
def get_optimal_cluster_amount(data: pd.DataFrame) -> int:
    """..."""
    cluster_amount_range, cost = build_elbow_curve(data)
    kl = KneeLocator(cluster_amount_range, cost, curve="convex", direction="decreasing")
    exact_cluster_amount = kl.elbow
    return exact_cluster_amount
```

Рисунок 3.5 Функція знаходження оптимальної кількості кластерів

Основна функція hierarchical, яка приймає на вхід дані для обрахунку, стратегію зв’язування, інтерполяційний метод для знаходження оптимального порогу відстані та підпис до графіка (рисунок 3.6). Також на рисунку 3.6 зображено приклад виклику функції для стратегії зв’язування за методом Уорда.

Функція hierarchical складається з 3-ьох основних частин:

г) кластеризація;

д) знаходження оптимального порогу відстань;

е) побудова дендрограми.

```
def hierarchical(data, method, dist_method, title):
    (clustering, distances, labels) = clastering(data, method, None)
    threshold = get_optimal_distance(distances, dist_method)
    (clustering, distances, labels) = clastering(data, method, threshold)

    data_index_copy = data.copy(deep=True)
    data_index_clusterize = data_index_copy
    data_index_clusterize.insert(0, CLUSTER_COLUMN_NAME, labels, True)
    data_index_clusterize = data_index_clusterize.sort_values(
        by=[CLUSTER_COLUMN_NAME])

    plt.title(title, size=18)
    plot_dendrogram(clustering, labels=data_index_copy.index, truncate_mode="level",
                    p=data_index_copy.shape[0], color_threshold=threshold)
    display(data_index_clusterize)

hierarchical(data_index, "ward", 'polynomial', "Ward Linkage")
```

Рисунок 3.6 Головна функція ієрархічної кластеризації

Для кластеризації використовувалась бібліотека Scikit-learn та метод AgglomerativeClustering з неї [12, 15] який приймає на вхід стратегію зв’язування, поріг відстані та інші параметри (фрагмент програми наведено на рисунку 3.7).

```
def clastering(data, method, threshold):
    if threshold is None:
        clustering = AgglomerativeClustering(linkage=method, compute_distances=True)
    else:
        clustering = AgglomerativeClustering(linkage=method, n_clusters=None, distance_threshold=threshold)
    result = clustering.fit(data)
    return (clustering, result.distances_, result.labels_)
```

Рисунок 3.7 Функція кластеризації

Після того як проведено кластеризацію знаходиться поріг відстані (фрагмент програми наведено на рисунку 3.8), зважаючи на яку відбувається розбиття на кластери. Для цього використовується KneeLocator з бібліотеки kneed [13, 14]. Який використовує один з двох інтерполяційних методів:

а) interp1d – використовує метод „scipy.interpolate.interp1d”;

б) polynomial – використовує метод „numpy.polyfit”, та призначається для гладкіших кривих.

```
def get_optimal_distance(distances, dist_method):
    # Знаходження оптимальної відстані
    K = range(0, len(distances))
    kl = KneeLocator(K, distances, curve="convex", direction="increasing", interp_method=dist_method)

    best_k = kl.elbow
    threshold = distances[best_k]

    # Побудова графіка
    resetFigSize()
    ax1 = plt.axes()
    ax1.plot(distances)
    ax1.scatter(best_k, threshold,  s=80, facecolors='none', edgecolors='r', linewidths=2)
    plt.show()
    setFigSize([25, 11])

    print(threshold)

    return threshold
```

Рисунок 3.8 Функція знаходження оптимального порогу відстані

Після того, як поріг відстані знайдено будується матриця зв’язку та на її основі малюється дендрограма (фрагмент програми наведено на рисунку 3.9).

```python
def plot_dendrogram(model, **kwargs):
    # Створення матриці зв'язку і побудова дендрограми

    # Обрахунок кількості точок під кожною вершиною
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # Вершина листок
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack(
        [model.children_, model.distances_, counts]).astype(float)

    # Побудова дендрограми
    dendrogram(linkage_matrix, **kwargs)
    plt.show()
```

Рисунок 3.9 Функція для побудови дендрограми

4 РЕЗУЛЬТАТИ

Розроблений веб застосунок має декілька сторінок. На рисунку 4.1 зображено початкову сторінку на якій можна завантажити файл з даними про перехресне зв’язування антитіл.

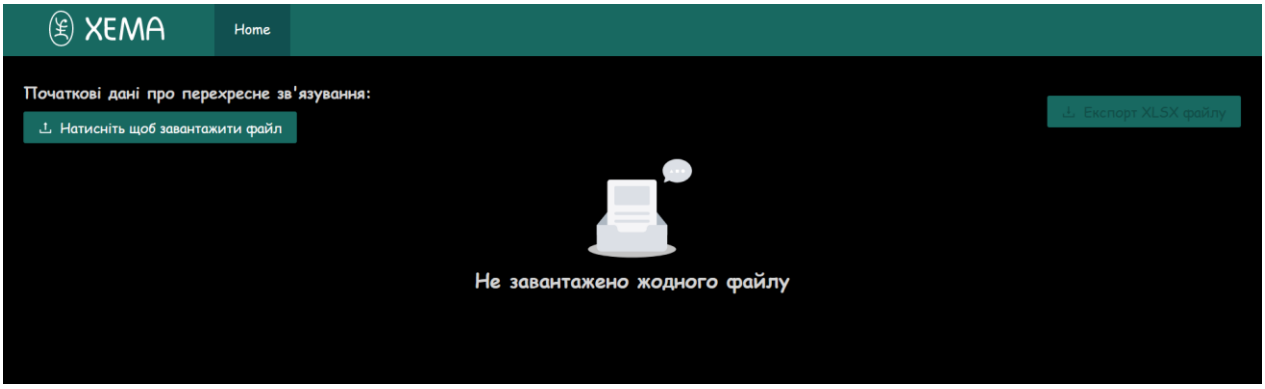

Рисунок 4.1 Початкова сторінка веб застосунку

Результуючі дані на інтерфейсі користувача відображаються у два способи:

а) у вигляді карток де заголовок вказує на групу, а в тілі міститься посортований список антитіл (рисунок 4.2);

б) у вигляді початкових даних, позначених кольорами (рисунок 4.3).

Співпадіння кольорів в тій чи іншій колонці в середині певного кластеру підтверджує, що вони справді належать до однієї групи.

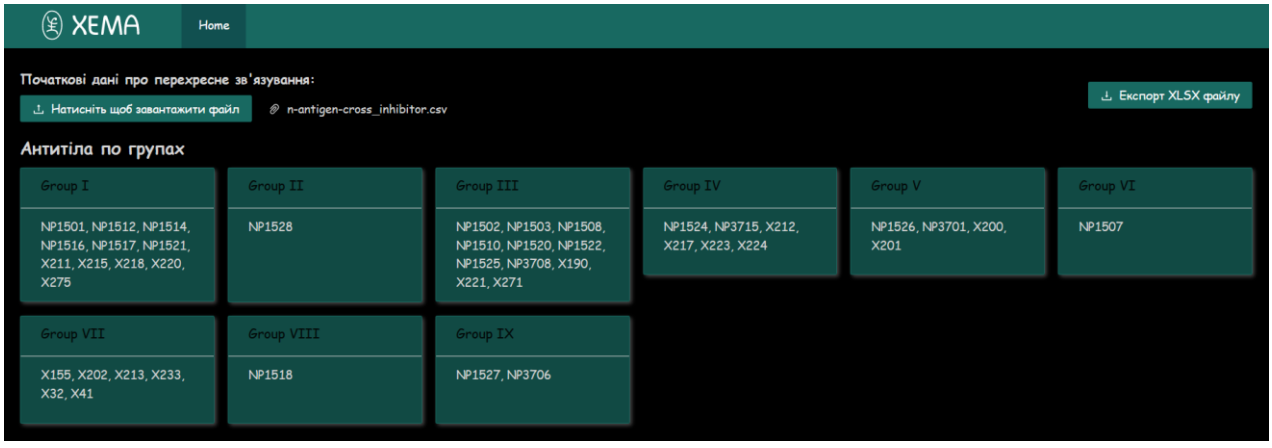

Рисунок 4.2 Результат кластеризації у вигляді карток

Рисунок 4.3 Фрагмент початкових даних розбитих на групи

Також реалізовано збереження результатів у файл формату .xlsx. Ці результати також подані у вигляді таблиці та схожі на результати подані на веб сторінці і складаються з:

а) таблиці де заголовок вказує на групу, а в тілі міститься посортований список антитіл (рисунок 4.4);

б) таблиця з початковими даними розбита на групи та позначена кольорами (рисунок 4.5).

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8	Group 9
1	NP1501	NP1528	NP1502	NP1524	NP1526	NP1507	X155	NP1518	NP1527
2	NP1512		NP1503	NP3715	NP3701		X202		NP3706
3	NP1514		NP1508	X212	X200		X213		
4	NP1516		NP1510	X217	X201		X233		
5	NP1517		NP1520	X223			X32		
6	NP1521		NP1522	X224			X41		
7	X211		NP1525						
8	X215		NP3708						
9	X218		X190						
10	X220		X221						
11	X275		X271						

Рисунок 4.4 Результат кластеризації записаний у файл

		NP1501*	NP1502*	NP1503*	NP1508*	NP1510*	NP1514*	NP1516*	NP1517*	NP1518*	NP1520*	NP1521*	NP1522*	NP1524*	NP1525*	NP1527*	NP3706*
Group 1	NP1501	0.449	0.715	1.0664	1.0248	1.136	0.26	0.4172	0.268	0.8512	0.614	0.596	0.7326	1.1392	1.061	0.852	0.9114
	NP1512	0.79	0.789	0.876	0.7504	0.979	0.735	1.015	0.737	0.816	0.853	0.638	0.7101	1.1096	0.836	0.841	0.7623
	NP1514	0.448	0.822	1.0968	1.0088	1.189	0.2475	0.4858	0.253	1.0208	1.04	0.626	0.7821	1.2512	1.136	1.024	0.84
	NP1516	0.385	1.034	0.9832	0.9304	1.053	0.1775	0.3052	0.152	0.9504	0.782	0.455	0.6399	1.0936	0.99	0.752	0.7833
	NP1517	0.425	0.644	0.7952	0.7304	0.885	0.155	0.2758	0.079	1.1504	0.636	0.414	0.5427	1.0624	0.92	0.71	0.8673
	NP1521	0.538	0.629	0.9264	0.9016	1.057	0.4425	0.6342	0.574	1.2048	0.851	0.107	0.729	1.0872	1.058	0.754	1.071
	X211	0.64	0.861	1.2072	1.1376	1.365	0.765	0.8428	0.836	0.7184	1.241	0.598	0.9387	0.7984	1.333	0.215	0.8043
	X215	0.615	0.824	1.1512	1.1752	1.344	0.605	0.4928	0.665	0.9296	1.017	0.339	0.9045	1.0648	1.332	1.23	0.9345
	X218	0.424	0.729	1.1096	1.1384	1.228	0.5475	0.4956	0.653	0.6256	0.968	0.39	0.9072	0.4816	1.219	0.947	0.6825
	X220	0.508	0.696	0.9744	0.9952	1.131	0.4375	0.4844	0.59	0.9872	0.828	0.393	0.8703	0.7072	1.173	0.616	0.8337
	X275	0.437	0.575	0.7456	0.6936	0.85	0.5575	0.3836	0.708	1.0864	0.718	0.378	0.5904	0.9064	0.915	0.511	0.8526
Group 2	NP1528	1.103	0.865	0.2632	0.1912	0.231	0.3075	0.5474	0.445	0.6592	0.258	0.273	0.2736	0.316	0.257	1.075	0.7728
Group 3	NP1502	0.893	0.425	0.336	0.196	0.349	0.3625	0.665	0.45	1	0.188	0.535	0.1719	1.1584	0.223	0.802	0.9303
	NP1503	0.768	0.309	0.068	0.052	0.095	0.305	0.7126	0.408	0.9888	0.075	0.57	0.063	1.1224	0.076	0.858	0.8169
	NP1508	0.732	0.388	0.1456	0.0848	0.19	0.345	0.8456	0.411	1.1344	0.144	0.55	0.1017	1.1632	0.135	0.806	0.8736
	NP1510	0.781	0.382	0.2152	0.1056	0.233	0.495	0.7826	0.527	1	0.227	0.661	0.1332	1.1216	0.146	0.836	0.8505
	NP1520	0.669	0.503	0.0856	0.0536	0.106	0.315	0.5866	0.406	0.8352	0.095	0.431	0.0702	1.028	0.069	0.765	0.8274
	NP1522	0.669	0.384	0.1608	0.0912	0.17	0.4725	0.7476	0.574	1.0064	0.205	0.441	0.1143	1.0672	0.128	0.795	0.9093
	NP1525	1.003	0.389	0.1272	0.0872	0.148	0.765	0.9506	0.67	0.9728	0.166	0.672	0.126	0.892	0.123	0.872	0.9429
	NP3708	0.676	0.378	0.2016	0.1336	0.254	0.185	0.392	0.175	0.9568	0.237	0.624	0.1476	0.9704	0.189	0.735	0.8484
	X190	0.442	0.386	0.0576	0.0448	0.068	0.3275	0.4452	0.239	0.904	0.077	0.349	0.063	0.9232	0.068	0.82	0.7581
	X221	0.646	0.343	0.2072	0.1576	0.245	0.56	0.6818	0.603	0.9632	0.237	0.484	0.1719	0.9912	0.209	1.348	0.8274
	X271	0.487	0.378	0.0808	0.0552	0.104	0.5375	0.4648	0.538	0.9472	0.109	0.399	0.0747	0.8736	0.084	0.81	0.8967
Group 4	NP1524	0.604	0.948	1.1128	1.0584	1.194	0.7425	0.9926	0.833	0.456	1.185	0.675	0.8163	0.0728	1.123	0.712	0.6027
	NP3715	0.605	0.551	0.7296	0.6112	0.848	0.41	0.7686	0.554	0.2864	0.855	0.612	0.522	0.0712	0.725	0.768	0.6762
	X212	0.712	0.805	1.2536	1.2192	1.414	0.87	0.9436	0.872	0.2224	1.268	0.664	0.972	0.0728	1.406	1.94	0.8085
	X217	0.476	0.733	0.9376	0.956	1.072	0.5325	0.4494	0.609	0.2096	0.844	0.305	0.7965	0.0568	1.124	0.738	0.5775
	X223	0.694	0.79	1.0064	1.0096	1.103	0.745	0.7616	0.849	0.2016	1.113	0.539	0.7848	0.06	1.14	1.618	0.7665
	X224	0.895	0.939	1.0288	0.9728	1.169	0.8475	0.9408	0.882	0.2304	0.825	0.562	0.7947	0.0656	1.154	2.033	0.8295

Рисунок 4.5 Фрагмент початкових даних розбитих на групи та записаних у файл

Тепер перейдемо безпосередньо до аналізу числових результатів. Очікувалось, що початкові дані будуть погруповані так як зображено в таблиці 4.1.

Таблиця 4.1 Очікуваний результат кластеризації

1A	1B	1B/2	2	2B/3	3A	3B	4A	4B	4C	5
NP1501	X190	NP1512	NP1502	NP1528	X202	X32	NP3706	X211	X215	X220
NP1514	NP1526	NP1521	NP1503		X218	X41				X275
NP1516	X200		NP1508		NP1518	X155				
NP1517	X201		NP1510		NP1527	X212				
NP1507			NP1520			X213				
			NP1522			X217				
			NP1525			X223				
			X221			X224				
			X271			X233				
			NP3701			NP1524				
			NP3708			NP3715				

4.1 Результати роботи алгоритм для категоріальних змінних

Спочатку знаходиться WCD для кожного k – кількість кластерів від 0 до розміру вибірки, та на основі цих даних будується графік та з використанням ліктьового методу знаходять оптимальна кількість кластерів. Для наших даних з

графіку WCD до кількості кластерів (рисунок 4.6) видно, що оптимальна кількість кластерів буде 9.

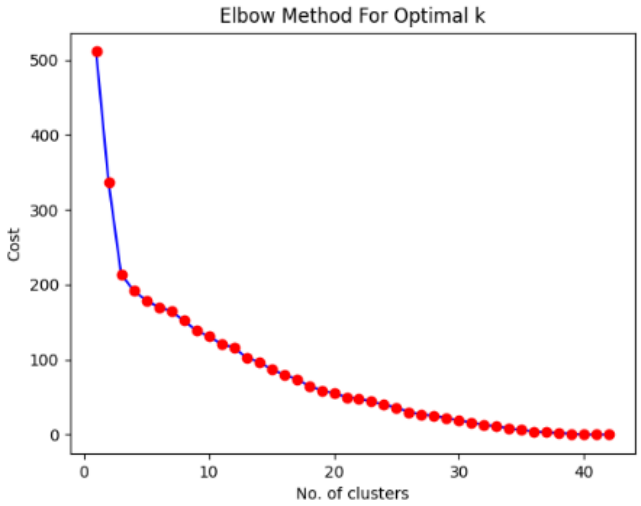

Рисунок 4.6 Графік кількості кластерів до відстані в середині кластера

Коли оптимальна кількість кластерів знайдена, відбувається безпосередня кластеризація методом k-modes. Результат кластеризації наведено в таблиці 4.2.

Таблиця 4.2 Результат кластеризації методом k-modes

1	2	3	4	5	6	7	8	9
NP1501	NP1528	NP1502	NP1524	NP1526	NP1507	X155	NP1518	NP1527
NP1512		NP1503	NP3715	NP3701		X202		NP3706
NP1514		NP1508	X212	X200		X213		
NP1516		NP1510	X217	X201		X233		
NP1517		NP1520	X223			X32		
NP1521		NP1522	X224			X41		
X211		NP1525						
X215		NP3708						
X218		X190						
X220		X221						
X275		X271						

Як можна побачити з таблиці, в методі k-modes лише один кластер 2 повністю співпадає з групою 2В/3. Кластер 3 майже повторює групу 2 за винятком елемента NP 3701. Кластер 4 та кластер 7 містять в собі групу 3В та елемент X202.

4.1 Результати роботи алгоритм для кількісних змінних

В результаті роботи програми ми отримали 4 висновки для кожної стратегії зв’язування.

Спершу розглянемо результати використання метода Уорда. Для нього оптимальний поріг відстані буде рівним 1.67, що видно з рисунку 4.7.

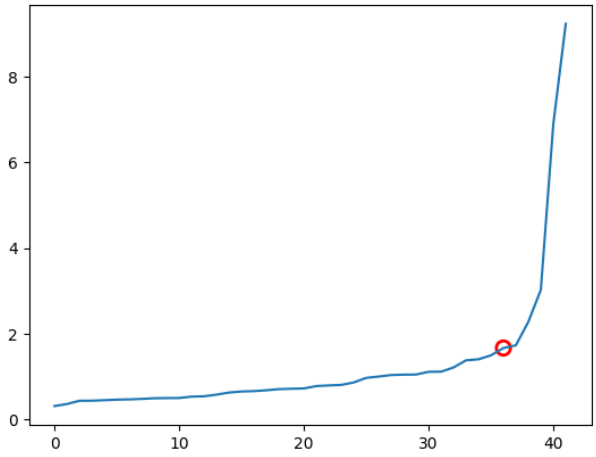

Рисунок 4.7 Графік відстаней для методу Уорда

На дендрограмі (рисунок 4.8) чітко видно, що в результаті роботи алгоритму було виділено 7 кластерів. У таблиці 4.3 наведено результат кластеризації, де кожна колонка містить перелік антитіл, які належать до кластеру.

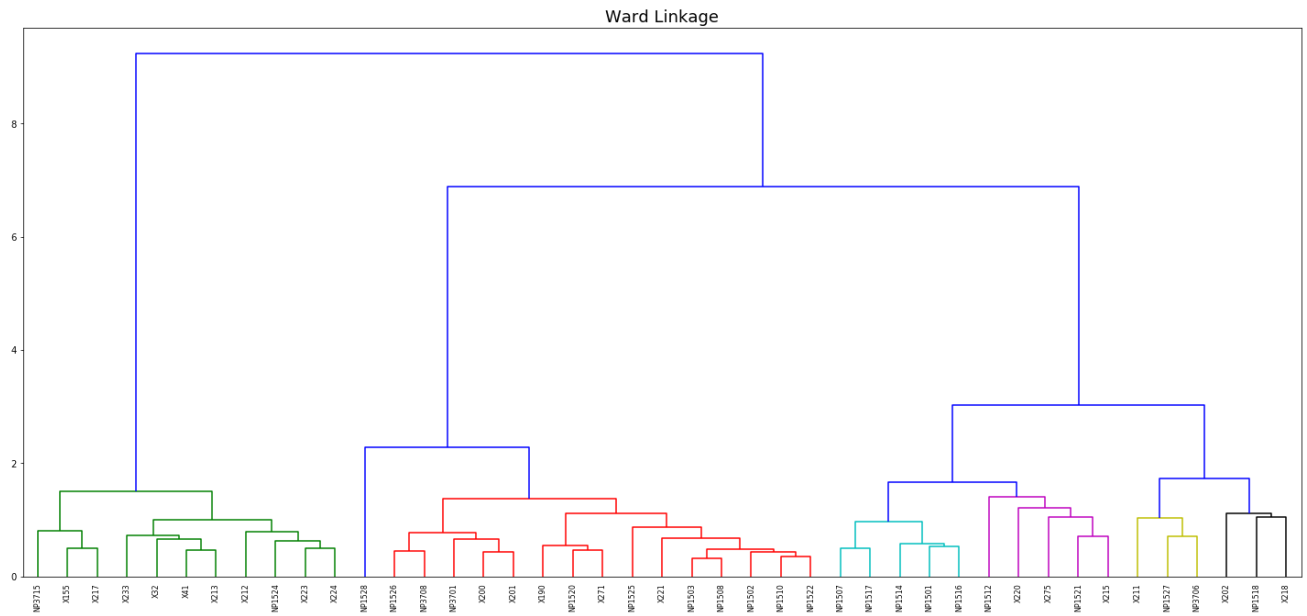

Рисунок 4.8 Дендрограма кластеризації з використанням методу Уорда

Чітко видно, що кластер під номером 1 відповідає групі 3B, кластер 5 повністю відповідає групі 2B/3 і кластер 7 відповідає групі 1A (вони позначені зеленим кольором). Також кластер 3 об'єднує групи 1B і 2, кластер 4 відповідає групі 3A без елемента NP1527, який знаходиться в кластері 6, що також містить групи 4A і 4B (вони позначені жовтим і помаранчевим кольорами).

Таблиця 4.3 Результат кластеризації з використанням методу Уорда

1	2	3	4	5	6	7
X41	NP1521	X200	X202	NP1528	NP1527	NP1517
X32	X275	X190	NP1518		X211	NP1514
X233	X215	X221	X218		NP3706	NP1516
X224	NP1512	X201				NP1507
X223	X220	X271				NP1501
X217		NP3708				
X213		NP3701				
X212		NP1526				
X155		NP1525				
NP371		NP1522				
NP152		NP1520				
		NP1502				
		NP1503				
		NP1510				
		NP1508				

Після цього розглянемо результати використання методу повного зв'язку. Для нього оптимальний поріг відстані буде рівним 1.16, що видно з рисунку 4.9.

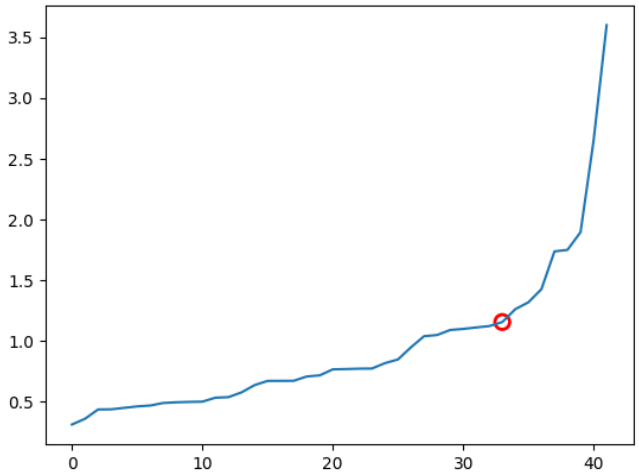

Рисунок 4.9 Графік відстаней для методу повного зв'язку

На дендрограмі (рисунок 4.10) чітко видно, що в результаті роботи алгоритму було виділено 9 кластерів. У таблиці 4.4 наведено результат кластеризації.

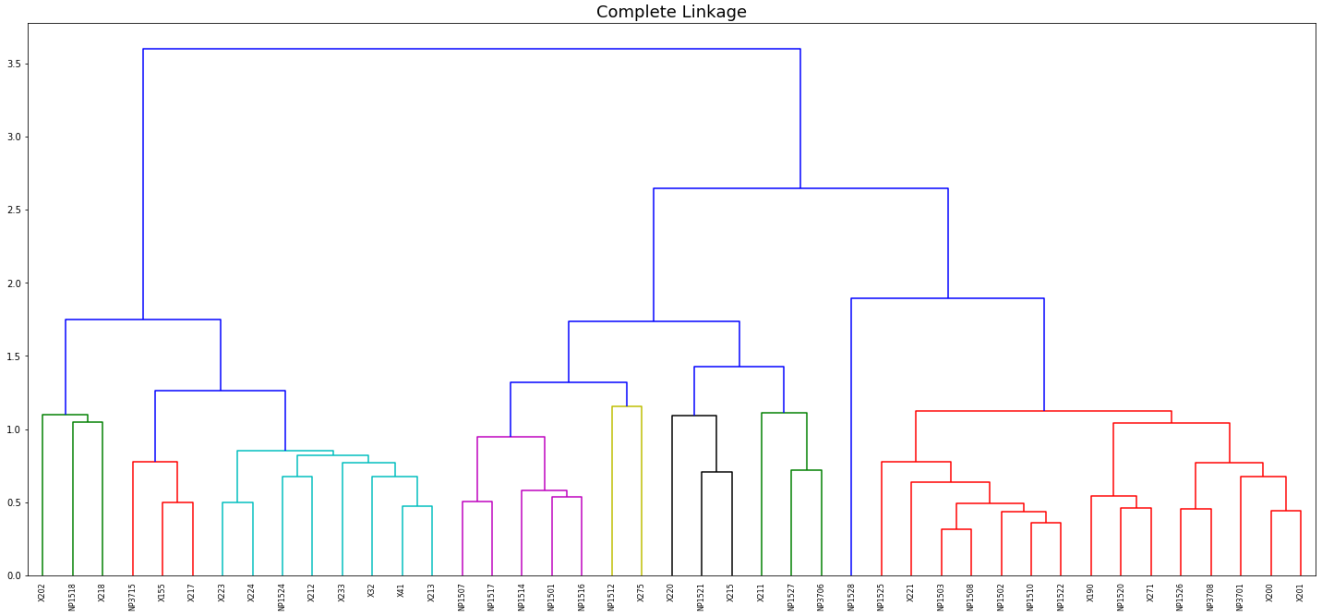

Рисунок 4.10 Дендрограма кластеризації з використанням методу повного зв’язку

Таблиця 4.4 Результат кластеризації з використанням методу повного зв’язку

1	2	3	4	5	6	7	8	9
X275	X221	X211	NP1507	X202	X213	NP1521	NP1528	X217
NP1512	X201	NP1527	NP1501	NP1518	X32	X215		X155
	X200	NP3706	NP1516	X218	X41	X220		NP3715
	X190		NP1517		X233			
	X271		NP1514		X224			
	NP3701				NP1524			
	NP1526				X212			
	NP1525				X223			
	NP1522							
	NP3708							
	NP1502							
	NP1503							
	NP1520							
	NP1508							
	NP1510							

Чітко видно, що кластер номер 4 відповідає групі 1A, а кластер номер 8 повністю відповідає групі 2B/3 (вони позначені зеленим кольором). Також кластер

2 об’єднує групи 1B і 2, кластер 5 відповідає групі 3A без елемента NP1527, який знаходиться в кластері 3, який також містить групи 4A і 4B, крім того кластер 6 і кластер 9 містять елементи з групи 3B (вони виділені жовтим і помаранчевим кольорами).

Третім розглянемо метод середнього зв’язку. Для нього оптимальний поріг відстані буде рівним 0.99, що видно з рисунку 4.11.

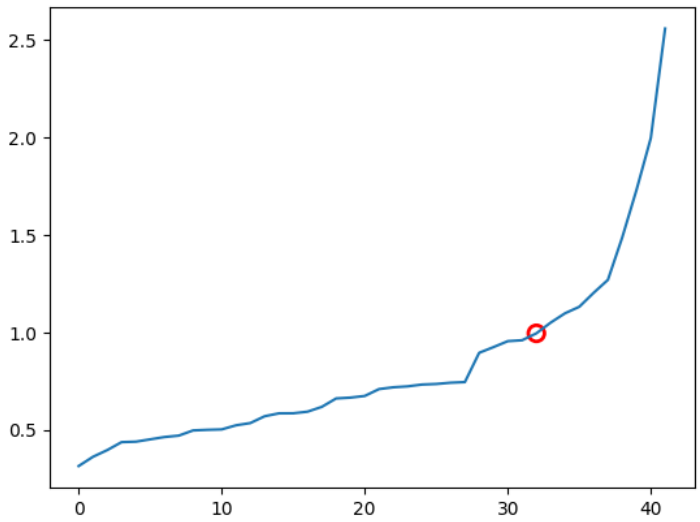

Рисунок 4.11 Графік відстаней для методу середнього зв’язку

На дендрограмі (рисунок 4.12) чітко видно, що в результаті роботи алгоритму було виділено 11 кластерів. У таблиці 4.5 наведено результат кластеризації.

Чітко видно, що кластер номер 2 відповідає групі 3B, кластер 5 повністю відповідає групі 1A і кластер 8 відповідає групі 2B/3 (вони позначені зеленим кольором). Також кластер 6 об’єднує групи 1B і 2, кластери 9, 10 і 11 відповідають групі 3A без елемента NP1527, який знаходиться в кластері 6, який також містить групи 4A і 4B (вони позначені жовтим і помаранчевим кольорами).

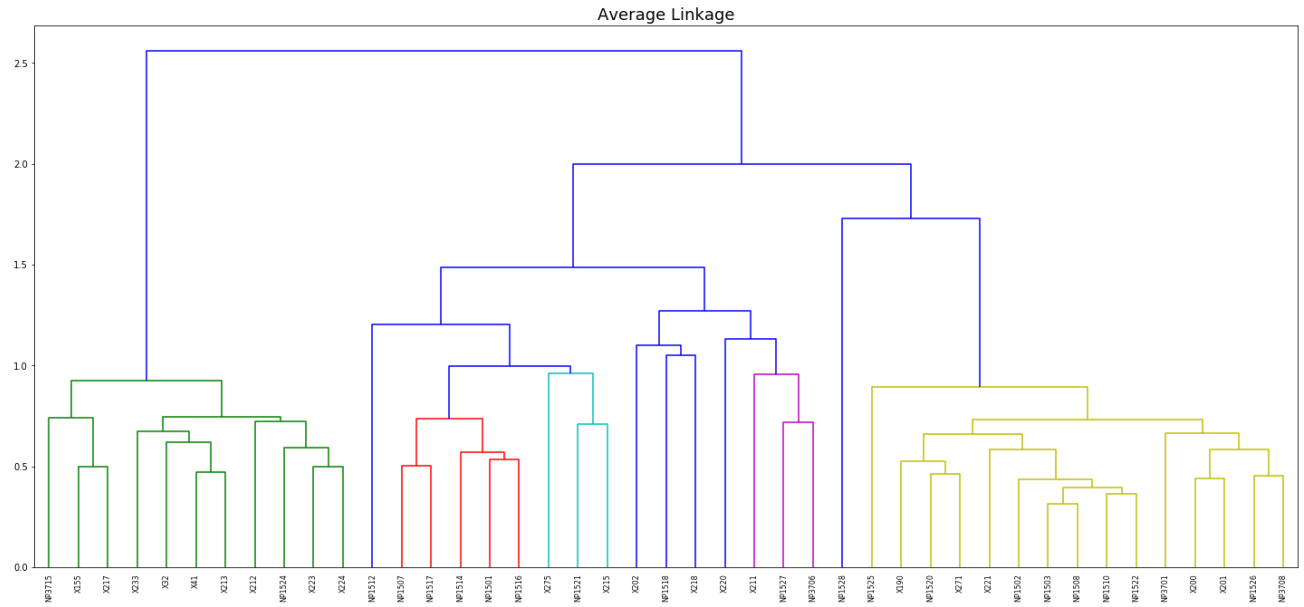

Рисунок 4.12 Графік відстаней для методу середнього зв’язку

Таблиця 4.5 Результат кластеризації з використанням методу середнього зв’язку

1	2	3	4	5	6	7	8	9, 10, 11		
X275	NP3715	X211	X220	NP1501	NP1502	NP1512	NP1528	X202	NP1518	X218
NP1521	NP1524	NP3706		NP1517	NP1503					
X215	X32	NP1527		NP1507	X221					
	X41			NP1514	NP1508					
	X155			NP1516	NP1510					
	X212				NP3701					
	X213				X200					
	X217				X190					
	X223				NP1520					
	X224				NP1522					
	X233				NP1525					
					X271					
					NP1526					
					X201					
					NP3708					

Останнім розглянемо метод одиночного зв’язку. Для нього оптимальний поріг відстані буде рівним 1.095, що видно з рисунку 4.13.

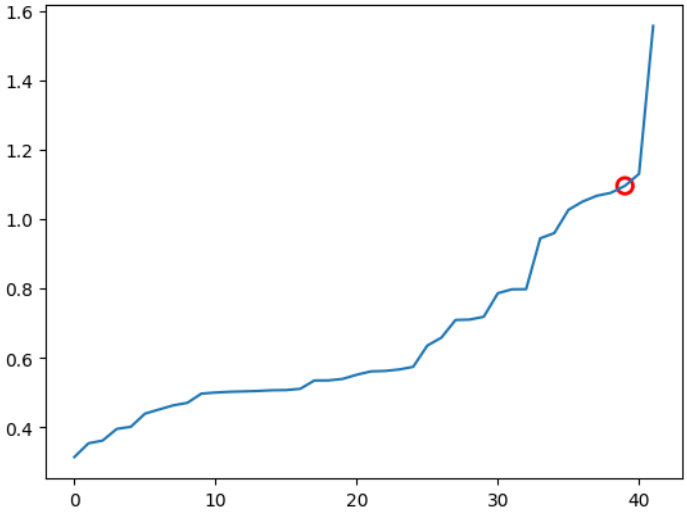

Рисунок 4.13 Графік відстаней для методу одиничного зв’язку

На дендрограмі (рисунок 4.14) чітко видно, що на виході алгоритму було виділено 4 кластери. У таблиці 4.6 наведено результат кластеризації.

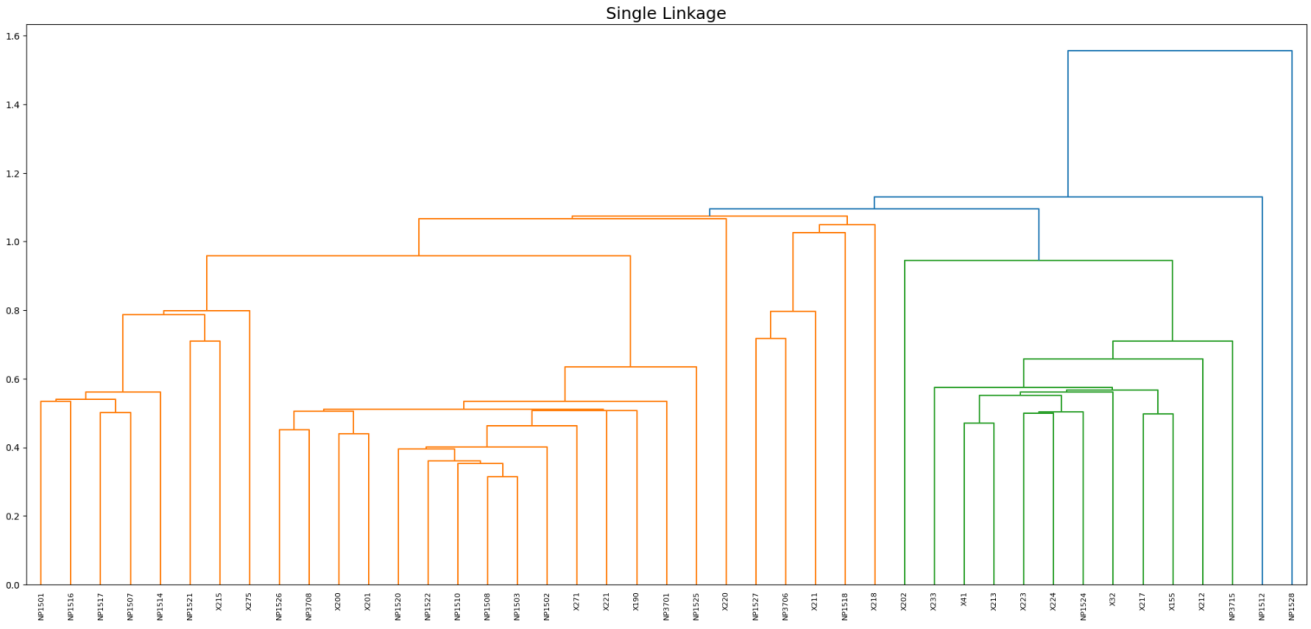

Рисунок 4.14 Графік відстаней для методу одиничного зв’язку

З розподілу даних очевидно, що метод одиничного зв’язку отримав завелике порогове значення, тому що кластер під номером 1 об’єднав в собі майже всі групи. Кластер номер 2 складається з групи 3B та елемента X202, а кластер 8 відповідає групі 2B/3.

Таблиця 4.6 Результат кластеризації з використанням методу одиничного зв’язку

1	2	3	4
NP1527	X213	NP1512	NP1528
NP3706	X32		
X201	X41		
X200	X155		
X221	NP1524		
X190	X212		
NP3708	NP3715		
NP3701	X217		
NP1502	X223		
NP1526	X224		
NP1525	X233		
NP1522	X202		
NP1503			
NP1508			
NP1520			
X271			
NP1510			
NP1517			
NP1514			
NP1507			
NP1516			
NP1501			
X220			
X218			
NP1521			
X215			
NP1518			
X275			
X211			

Отже, якщо за метрику точності взяти сумарну кількість елементів у кластерах, які повністю відповідають очікуваному результату, то очевидно, що найкращий результат дає алгоритм, який використовував метод Уорда та метод повного зв’язку. Алгоритм, який використовував метод середнього зв’язку дав не набагато гірший результат. А алгоритм одиночного зв’язку дав найгірший результат зважаючи на те що він об’єднав багато груп в один кластер.

ВИСНОВОК

В результаті виконання магістерської було розглянуто та проаналізовано декілька алгоритмів кластеризації серед яких:

а) k-modes (модифікація методу k-середніх);

б) агломеративна кластеризація (підвид ієрархічної кластеризації) з використанням наступних стратегії зв’язування:

1. метод Уорда (Ward linkage);
2. метод повного зв’язку (Complete linkage);
3. метод середнього зв’язку (Average linkage);
4. метод одиночного зв’язку (Single linkage).

Було показано, що методи ієрархічної кластеризації краще показали себе ніж метод k-modes.

Також розроблено веб застосунок, який полегшує задачу розбиття списку антитіл на групи на основі індексу перехресного зв’язування. Цей застосунок представляє з себе три мікросервіси:

а) інтерфейс користувача для вивантаження файлів, перегляду та завантаження результатів;

б) прикладний програмний інтерфейс, який відповідає за кластеризацію;

в) API, що поєднує інтерфейс для кластеризацію та інтерфейс користувача, а також реалізує завантаження результатів.

Для розробки системи було використано NET 5 з ASP.NET, Python 3 з Flask та Type Script з React і Ant Design.

Також було виконано кластеризацію антитіл для конкретного вірусного протеїну. Зважаючи на те, які антитіла опинились в однакових групах та порівнюючи очікуваний та отриманий результат можна сказати, що правильність розбиття на кластери склала ~41%.

Важливим аспектом, є те, що попри невеликий об’єм даних (матриця 40x30) розроблений застосунок за невеликий час (1-2 хвилини) робить об’єм роботи, для виконання якого людині потрібно було б декілька днів. Це показує, що даний

застосунок є корисним, оскільки при збільшенні вибірки кількість часу, що потрібна на виконання алгоритму комп’ютером буде залишатись невеликою в порівнянні з часом який потрібен буде людині для виконання такого ж завдання.

СПИСОК ВИКОРИСТОНАЇ ЛІТЕРАТУРИ

1. WHO Coronavirus (COVID-19) Dashboard, 28 September 2022, URL: https://covid19.who.int/
2. Соболь В. І. Біологія: підручник для 9 класу загальноосвітніх навчальних закладів / В. І. Соболь. – Кам'янець-Подільський: Абетка, 2017 р. –с. 217-218.
3. Скок М. В. Антитіло / М. В. Скок // Велика українська енциклопедія. URL: https://vue.gov.ua/Антитіло (дата звернення: 6.11.2022). Режим доступу: https://pidruchnyk.com.ua/912-biologiya-sobol-9-klas.html
4. Словник української мови: в 11 томах. – Том 1, 1970. – с. 50. Режим доступу: http://sum.in.ua/s/antytila
5. Еколого-економічна оптимізація виробництва: методи та засоби кластерного аналізу // методичні вказівки до виконання лабораторних робіт .– Київ, 2016.– Режим доступу: https://ela.kpi.ua/bitstream/123456789/15376/1/Klasternyi_analis.pdf
6. Nakane PK., Peroxidase-labeled antibody. A new method of conjugation / PK. Nakane, A. Kawaoi // Journal of Histochemistry & Cytochemistry, 1974; 22(12), 1084-1091. https://doi.org/10.1177/22.12.1084
7. Huang Z. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values (1998). Data Mining and Knowledge Discovery. 2(3): 283–304.
8. Aprilliant A. The k-modes as Clustering Algorithm for Categorical Data Type / A. Aprilliant [Electronic resource]. – 2021. – Available from: https://medium.com/geekculture/the-k-modes-as-clustering-algorithm-for-categorical-data-type-bcde8f95efd7.
9. Cao F. A new initialization method for categorical data clustering / F. Cao, J. Liang, L. Liang Bai [Electronic resource]. – 2009. – Available from: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.8181&rep=rep1&type=pdf

10. Nielsen F., Introduction to HPC with MPI for Data Science, Chapter 8: Hierarchical Clustering / F. Nielsen // Springer, Switzerland, 2016, 195-211. https://www.doi.org/10.1007/978-3-319-21903-5_8.
11. SAS/STAT(R) 9.2 User's Guide, Second Edition, [Electronic resource]. – 2022. – Available from: https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_cluster_sect012.htm
12. Scikit-learn documentation, AgglomerativeClustering, [Electronic resource]. – 2022. – Available from: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html?highlight=ag#sklearn.cluster.AgglomerativeClustering.
13. Kneed documentation, Parameter Example, [Electronic resource].– 2020, .– Available from: https://kneed.readthedocs.io/ en/stable/parameters.html
14. Baruah I. D. Cheat sheet for implementing 7 methods for selecting the optimal number of clusters in Python / I. D. Baruah // [Electronic resource].– 2020.– Available from: https://towardsdatascience.com/cheat-sheet-to-implementing-7-methods-for-selecting-optimal-number-of-clusters-in-python-898241e1d6ad
15. Alam B. Implementation of Hierarchical Clustering using Python / B. Alam // [Electronic resource].– 2022.– Available from: https://hands-on.cloud/implementation-of-hierarchical-clustering-using-python/.
16. Зелінський О. Розробка системи кластеризації антитіл на основі коефіцієнту перехресного зв'язування / О. Зелінський, В. Горлач, Ю. Лебедін // Міжнародна студентська наукова конференція з питань прикладної математики та комп'ютерних наук (МСНКПМК-2022), 5-6 травня 2022 р. – Львів, 2022. – С. 8-12. Режим доступу: https://ami.lnu.edu.ua/wp-content/uploads/2022/05/ISSCAMCS-2022.pdf
17. Anaya JM. Autoimmunity: From Bench to Bedside / JM. Anaya, Y. Shoenfeld, A. Rojas-Villarraga // Bogota, Colombia, 2018, Available from: https://www.ncbi.nlm.nih.gov /books/NBK459443/.
18. Грицко Р.Ю. Що насправді означає тестування на коронавірус для пацієнта? / Р.Ю. Грицко, Г.І. Біла, Р.О. Білий // Інфекційні хвороби, Тернопільський

національний медичний університет, 2020, 65-72.– Режим доступу: https://ojs.tdmu.edu.ua/index.php/inf-patol/article/download/11287/10737/41013.

19. Lippi G. Potential preanalytical and analytical vulnerabilities in the laboratory diagnosis of coronavirus disease 2019 (COVID-19) / G. Lippi, A.-M. Simundic, M. Plebani // Clinical Chemistry and Laboratory Medicine (CCLM), 2020, 58 (7), 1070-1076. https://doi.org/10.1515/cclm-2020-0285.

ДОДАТОК А. КОД ПРИКЛАДНОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ ДЛЯ ВЗАЄМОДІЇ З ІНТЕРФЕЙСОМ КОРИСТУВАЧА (UI API)

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Threading.Tasks;
using Xema.Core.Models;
using Xema.Services.Infrastructure;

namespace Xema.WebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CrossInhibitonController : ControllerBase
    {
        private readonly ICrossInhibitionService _crossInhibitionService;
        private readonly IExcelService _excelService;

        public CrossInhibitonController(ICrossInhibitionService crossInhibitionService,
IExcelService excelService)
        {
            _crossInhibitionService = crossInhibitionService;
            _excelService = excelService;
        }

        [HttpPost("upload")]
        public async Task<ActionResult<CrossInhibitorRawDataModel>> UploadFile([FromForm]
IFormFile file)
        {
            var result = await _crossInhibitionService.Cluster(file);
            return Ok(result);
        }

        [HttpPost("export-xlsx")]
        public ActionResult DownloadExcel(CrossInhibitorRawDataModel dataModel)
        {
            var result = _excelService.GetFileStream(dataModel);
            return File(result.GetBuffer(), "application/octet-stream", "test.xlsx");
        }
    }
}
```

```
using Microsoft.AspNetCore.Http;
using System.Threading.Tasks;
using Xema.Core.Models;

namespace Xema.Services.Infrastructure
{
    public interface ICrossInhibitionService
    {
        Task<CrossInhibitorRawDataModel> Cluster(IFormFile file);
    }
}
```

```
using Microsoft.AspNetCore.Http;
using System.Threading.Tasks;
using Xema.ClusterAPI;
using Xema.Core.Models;
```

```
using Xema.Services.Infrastructure;

namespace Xema.Services
{
    public class CrossInhibitonSevice : ICrossInhibitionService
    {
        private readonly IClusterApiClient _clusterApiClient;

        public CrossInhibitonSevice(IClusterApiClient clusterApiClient)
        {
            _clusterApiClient = clusterApiClient;
        }

        public async Task<CrossInhibitorRawDataModel> Cluster(IFormFile file)
        {
            var response = await _clusterApiClient.Cluster(file);
            return response;
        }
    }
}
```

```
using Microsoft.AspNetCore.Http;
using System.Threading.Tasks;
using Xema.Core.Models;

namespace Xema.ClusterAPI
{
    public interface IClusterApiClient
    {
        public Task<CrossInhibitorRawDataModel> Cluster(IFormFile file);
    }
}
```

```
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Xema.ClusterAPI.Models;
using Xema.Core.Constants;
using Xema.Core.Enums;
using Xema.Core.Models;
using Xema.Core.Models.Configuration;

namespace Xema.ClusterAPI
{
    public class ClusterApiClient : IClusterApiClient
    {
        private readonly IHttpClientFactory _httpClientFactory;
        private readonly ClusterApiSettings _clusterApiSettings;

        public ClusterApiClient(IHttpClientFactory httpClientFactory, ClusterApiSettings
clusterApiSettings)
        {
            _httpClientFactory = httpClientFactory;
            _clusterApiSettings = clusterApiSettings;
        }

        public async Task<CrossInhibitorRawDataModel> Cluster(IFormFile file)
```

```
        {
            // Set up http client
            var httpClient = _httpClientFactory.CreateClient(_clusterApiSettings.Name);
            var formContent = PrepareContent(file);

            // Make a request
            var responseMessage = await
httpClient.PostAsync(_clusterApiSettings.ClusterEndpoint, formContent);
            responseMessage.EnsureSuccessStatusCode();

            // Response processing
            var response = await MapResponse(responseMessage);
            return response;
        }

        private MultipartFormDataContent PrepareContent(IFormFile file)
        {
            var fileStreamContent = new StreamContent(file.OpenReadStream());
            fileStreamContent.Headers.ContentType = new
MediaTypeHeaderValue(file.ContentType);

            var formContent = new MultipartFormDataContent
            {
                { fileStreamContent, "file", file.FileName }
            };

            return formContent;
        }

        private async Task<CrossInhibitorRawDataModel> MapResponse(HttpResponseMessage
responseMessage)
        {
            var responseAsString = await responseMessage.Content.ReadAsStringAsync();
            var clusterResponce =
JsonConvert.DeserializeObject<ClusterResponce>(responseAsString);

            // Map labels
            var antigenLabels = clusterResponce.Indexes.Keys.ToList();
            var markedAntigenLabels =
clusterResponce.Indexes.FirstOrDefault().Value.Keys.ToList();

            // Map clusters
            var clusters = new Dictionary<string, List<string>>();
            foreach (var cluster in clusterResponce.Clusters)
            {
                var key = cluster.Value.ToString();
                if (clusters.ContainsKey(key))
                {
                    clusters[key].Add(cluster.Key);

                    clusters[key] = clusters[key]
                        .OrderBy(x => x)
                        .ToList();
                }
                else
                {
                    clusters.Add(key, new List<string> { cluster.Key });
                }
            }

            // Add wrong data cluster
            if (clusterResponce.Wrong.Count > 0)
            {
```

```
                clusters[ClusterLabel.UnknownGroup] =
clusterResponce.Wrong.Keys.ToList();
            }

            // Create list
            var crossInhibitionIndexes = new List<List<List<IndexCell>>>();
            foreach (var key in clusters.Keys)
            {
                crossInhibitionIndexes.Add(new List<List<IndexCell>>());
            }

            // Map values
            foreach (var antigenLabel in clusterResponce.Indexes.Keys)
            {
                var row = new List<IndexCell>();
                foreach (var markedAntigenLabel in
clusterResponce.Indexes[antigenLabel].Keys)
                {
                    var initialValue =
clusterResponce.InitialValue[antigenLabel][markedAntigenLabel];
                    var index =
clusterResponce.Indexes[antigenLabel][markedAntigenLabel];
                    var color = clusterResponce.Colors[antigenLabel][markedAntigenLabel];

                    var cell = new IndexCell
                    {
                        Value = initialValue,
                        MarkerIndex = index,
                        MarkerColor = color
                    };

                    row.Add(cell);
                }

                var clusterIndex = clusterResponce.Clusters[antigenLabel];
                crossInhibitionIndexes[clusterIndex].Add(row);
            }

            // Map wrong data
            foreach (var antigenLabel in clusterResponce.Wrong.Keys)
            {
                var row = new List<IndexCell>();
                foreach (var markedAntigenLabel in
clusterResponce.Wrong[antigenLabel].Keys)
                {
                    var initialValue =
clusterResponce.InitialValue[antigenLabel][markedAntigenLabel];
                    var index = clusterResponce.Wrong[antigenLabel][markedAntigenLabel];

                    var cell = new IndexCell
                    {
                        Value = initialValue,
                        MarkerIndex = index,
                        MarkerColor = InhibitionColors.None
                    };

                    row.Add(cell);
                }

                // Last cluster should be "Unknown group"
                crossInhibitionIndexes[crossInhibitionIndexes.Count - 1].Add(row);
            }

            // Split lables by cluster
```

```
            var antigenLabelsResult = new List<List<string>>();
            var skip = 0;
            foreach (var clusterIndexes in crossInhibitionIndexes)
            {
                var labelByCluster = antigenLabels
                    .Skip(skip)
                    .Take(clusterIndexes.Count)
                    .ToList();

                antigenLabelsResult.Add(labelByCluster);
                skip += clusterIndexes.Count;
            }

            // Add wrong data antigenLabels
            if (clusterResponce.Wrong.Count > 0)
            {
                antigenLabelsResult[antigenLabelsResult.Count - 1] =
clusterResponce.Wrong.Keys.ToList();
            }

            var result = new CrossInhibitorRawDataModel
            {
                AntigenLabels = antigenLabelsResult,
                MarkedAntigenLabels = markedAntigenLabels,
                Clusters = clusters,
                CrossInhibitionIndexes = crossInhibitionIndexes
            };

            return result;
        }
    }
}
```

```
using Newtonsoft.Json;
using System.Collections.Generic;
using Xema.Core.Enums;

namespace Xema.ClusterAPI.Models
{
    public class ClusterResponce
    {
        [JsonProperty("clusters")]
        public Dictionary<string, int> Clusters { get; set; }

        [JsonProperty("initial")]
        public Dictionary<string, Dictionary<string, double>> InitialValue { get; set; }

        [JsonProperty("index")]
        public Dictionary<string, Dictionary<string, double>> Indexes { get; set; }

        [JsonProperty("color")]
        public Dictionary<string, Dictionary<string, InhibitionColors>> Colors { get;
set; }

        [JsonProperty("wrong")]
        public Dictionary<string, Dictionary<string, double>> Wrong { get; set; }
    }
}
```

```
using System.Collections.Generic;
```

```
namespace Xema.Core.Models
{
    public class CrossInhibitorRawDataModel
    {
        public List<List<string>> AntigenLabels { get; set; }

        public List<string> MarkedAntigenLabels { get; set; }

        public List<List<List<IndexCell>>> CrossInhibitionIndexes { get; set; }

        public Dictionary<string, List<string>> Clusters { get; set; }
    }
}
```

```
using Xema.Core.Enums;

namespace Xema.Core.Models
{
    public class IndexCell
    {
        public double Value { get; set; }

        public double MarkerIndex { get; set; }

        public InhibitionColors MarkerColor { get; set; }
    }
}
```

ДОДАТОК Б КОД ІНТЕРФЕЙСУ КОРИСТУВАЧА

```
import React, { FC, useState } from 'react';
import CustomLayout from '../components/layout/layout';
import CrossInhibition from './cross-inhibition/cross-inhibition';
import { ConfigProvider } from 'antd';
import ukUA from 'antd/es/locale/uk_UA';

import './app.scss';

const App: FC = () => {
    const [loading, setLoading] = useState<boolean>(false);

    return (
        <ConfigProvider locale={ukUA}>
            <CustomLayout loading={loading}>
                <CrossInhibition setLoading={setLoading} />
            </CustomLayout>
        </ConfigProvider>
    );
}

export default App;
```

```
import { FC, useCallback, useState } from 'react';
import { Button, Upload, message, Typography, Empty, Row, Col } from 'antd';
import { RcFile } from 'antd/lib/upload';
import { DownloadOutlined, UploadOutlined } from '@ant-design/icons';
import { saveAs } from "file-saver";
import moment from 'moment';

import { exportXlsxApi, uploadFileApi } from '../../api/cross-inhibition-api';
import { CrossInhibitonRawDataModel } from './../../types/cross-inhibiton-raw-data-
model';
import ClusterResult from './component/cluster-result/cluster-result';

import './cross-inhibition.scss';
import CrossInhibitionGrid from './component/cross-inhibition-grid/cross-inhibition-
grid';

interface ICrossInhibitionProps {
    setLoading: (loading: boolean) => void
}

const CrossInhibition: FC<ICrossInhibitionProps> = ({ setLoading }:
ICrossInhibitionProps) => {
    const [parseResult, setParsedResult] = useState<CrossInhibitonRawDataModel>();
    const [exportLoading, setExportLoading] = useState<boolean>(false);
```

```
    const handleSubmit = useCallback((value) => {
        setLoading(true);
        const formData = new FormData();
        formData.append('file', value.file);

        uploadFileApi(formData).then(response => {
            setParsedResult(response);
            value.onSuccess();
            setLoading(false);
        }).catch((error) => {
            value.onError(error);
            setLoading(false);
        });
    }, [setLoading]);

    const onExport = useCallback(() => {
        if (parseResult) {
            setExportLoading(true);

            exportXlsxApi(parseResult).then((response) => {
                saveAs(new Blob([response.data]), `result-${moment().format("DD-MM-
YYYY-HH-mm")}.xlsx`);
                setExportLoading(false);
            }
            ).catch((error) => {
                message.error(error);
                setExportLoading(false);
            });
        }
    }, [parseResult]);

    const beforeUpload = (file: RcFile) => {
        const extensions = ['application/vnd.ms-excel',
'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;charset=utf-8',
'text/csv'];
        if (!extensions.includes(file.type)) {
            message.error(`Файл '${file.name}' має некоректний формат`);
        }
        return extensions.includes(file.type) ? true : Upload.LIST_IGNORE;
    };

    return (
        <div>
            <div className="upload-container">
                <Row align="middle" justify="space-between">
                    <Col>
                        <div className='upload-header'>
                            <Typography.Title level={5}>{"Початкові дані про
перехресне зв'язування: "}</Typography.Title>
```

```
                        </div>

                        <div>
                            <Upload
                                accept='.csv,.xls,.xlsx'
                                customRequest={handleSubmit}
                                beforeUpload={beforeUpload}
                                maxCount={1}
                            >
                                <Button type="primary" icon={<UploadOutlined
/>}>{"Натисніть щоб завантажити файл"}</Button>
                            </Upload>
                        </div>
                    </Col>
                    <Col>

                        <Button
                            type="primary"
                            icon={<DownloadOutlined />}
                            onClick={onExport}
                            loading={exportLoading}
                            disabled={!parseResult}
                        >
                            {"Експорт XLSX файлу"}
                        </Button>
                    </Col>
                </Row>
            </div>

            {
                parseResult
                    ? <div>
                        <ClusterResult data={parseResult?.clusters}>
                        </ClusterResult>

                        <CrossInhibitionGrid data={parseResult} />
                    </div>
                    : <Empty description={<Typography.Title level={4}>{"Не завантажено
жодного файлу"}</Typography.Title>} />
            }

        </div>
    );
}

export default CrossInhibition;


import { FC, useEffect, useState } from 'react';
import { Row, Typography, List, Card } from 'antd';
```

```
import { toRoman } from 'roman-numerals'
import { Clusters } from '../../../../types/cross-inhibiton-raw-data-model';

import './cluster-result.scss';

interface IClusterResultProps {
  data?: Clusters
}

const ClusterResult: FC<IClusterResultProps> = ({ data }: IClusterResultProps) => {
  const [localClusters, setClusters] = useState<[string, string[]][]>();

  useEffect(() => {
    if (data) {
      setClusters(Object.entries(data));
    }
  }, [data]);

  return (
    <>
      <Typography.Title level={4}>Антитіла по групах</Typography.Title>
      <Row>
        <List
          grid={{
            gutter: 16,
            xs: 1,
            sm: 2,
            md: 4,
            lg: 4,
            xl: 6,
            xxl: 3,
          }}
          dataSource={localClusters}
          renderItem={(item, index) => {
            const groupTitle = Number.isNaN(parseInt(item[0])) ? item[0] : `Group
${toRoman(parseInt(item[0]) + 1)}`;
            return (
              <List.Item key={`card-group-${index}`}>
                <Card
                  title={groupTitle}
                  className="cluster-result-card"
                >
                  {item[1].join(', ')}
                </Card>
              </List.Item>
            );
          }}
        />
      </Row>
    </>
```

```
  );
}

export default ClusterResult;
```

```
import { FC, useEffect, useState } from 'react';
import { Row, Typography } from 'antd';

import { CrossInhibitonRawDataModel } from '../../../../types/cross-inhibiton-raw-data-model';
import CrossInhibitionGridItem from '../cross-inhibition-grid-item/cross-inhibition-grid-item';

import './cross-inhibition-grid.scss';

interface ICrossInhibitionGridProps {
  data?: CrossInhibitonRawDataModel
}

const CrossInhibitionGrid: FC<ICrossInhibitionGridProps> = ({ data }: ICrossInhibitionGridProps) => {
  const [parseResult, setParsedResult] = useState<CrossInhibitonRawDataModel>();

  useEffect(() => {
    setParsedResult(data);
  }, [data]);

  return (
    <>
      <Typography.Title level={4}>Матриця перехресного зв'язування</Typography.Title>
      {parseResult?.crossInhibitionIndexes.map((crossInhibition, index) => (
        <Row justify="center" className='cross-inhibition-grid-item-container' key={`grid-${index}`} >
          <CrossInhibitionGridItem
            crossInhibitionIndexes={crossInhibition}
            antigenLabels={parseResult?.antigenLabels[index]}
            markedAntigenLabels={parseResult?.markedAntigenLabels}
          />
        </Row>
      ))
      }
    </>
  );
}

export default CrossInhibitionGrid;
```

```
import { FC, useMemo } from 'react';
import { Tag, Table } from 'antd';
import { ColumnType } from 'antd/lib/table';

import { CrossInhibitonIndexCell } from '../../../../types/cross-inhibiton-index-
cell';
import { InhibitionColors } from '../../../../types/enums/InhibitionColors';

import './cross-inhibition-grid-item.scss';

interface ICrossInhibitionGridItemProps {
  antigenLabels: string[];
  markedAntigenLabels: string[];
  crossInhibitionIndexes: CrossInhibitonIndexCell[][];
}

const CrossInhibitionGridItem: FC<ICrossInhibitionGridItemProps> = ({
  antigenLabels,
  markedAntigenLabels,
  crossInhibitionIndexes
}: ICrossInhibitionGridItemProps) => {
  const column = useMemo(() => {
    const headerLabels = markedAntigenLabels || [];

    const result = headerLabels.map((label, jIndex) => {
      return {
        title: label,
        dataIndex: jIndex,
        render: (value: CrossInhibitonIndexCell, _: CrossInhibitonIndexCell[], iIndex:
number) => {
          const color = value?.markerColor === InhibitionColors.DarkGreen
            ? "#00b04f"
            : (value?.markerColor === InhibitionColors.LightGreen ? "#92d050" :
undefined);

          return (
            <Tag
              className="index-tag"
              key={`tag-${iIndex}-${jIndex}`}
              color={color}
            >
              {value?.value}
            </Tag>
          );
        }
      } as ColumnType<CrossInhibitonIndexCell[]>;
    });

    const labelColumn = {
      title: 'Label',
```

```
      key: 'label',
      fixed: 'left',
      render: (value: CrossInhibitonIndexCell, record: CrossInhibitonIndexCell[],
iIndex: number) => {
        return <span style={{ fontWeight: 500 }}>{antigenLabels[iIndex]}</span>
      }
    } as ColumnType<CrossInhibitonIndexCell[]>;

    result.splice(0, 0, labelColumn);
    return result;
  }, [antigenLabels, markedAntigenLabels]);

  return (
    <Table
      className="cross-inhibition-table"
      columns={column}
      dataSource={crossInhibitionIndexes}
      pagination={false}
      scroll={{ x: (markedAntigenLabels.length || 0) * 72 }}
    />
  );
}

export default CrossInhibitionGridItem;
```

```
import instanceApi from '../utils/instance-api';
import { AxiosRequestConfig, AxiosResponse } from 'axios';
import { CrossInhibitonRawDataModel } from '../types/cross-inhibiton-raw-data-model';

export const uploadFileApi = async (formData: FormData):
Promise<CrossInhibitonRawDataModel> => {
  var config: AxiosRequestConfig = { headers: { 'Content-Type': 'multipart/form-data'
} };
  const result = await
instanceApi.post<CrossInhibitonRawDataModel>('CrossInhibiton/upload', formData,
config);
  return result.data;
};

export const exportXlsxApi = async (values: CrossInhibitonRawDataModel):
Promise<AxiosResponse<Blob>> => {
  return await instanceApi.post<Blob>('CrossInhibiton/export-xlsx', values, {
responseType: 'blob' });
};
```

ДОДАТОК В. КОД ЕКСПОРТУ ФАЙЛА У ФОРМАТІ .XLSX

```
using SpreadsheetLight;
using System.IO;
using System.Linq;
using Xema.Core.Enums;
using Xema.Core.Models;
using Xema.Services.Infrastructure;

namespace Xema.Services
{
    public class ExcelService : IExcelService
    {
        private readonly IStyleProvider _styleProvider;

        public ExcelService(IStyleProvider styleProvider)
        {
            _styleProvider = styleProvider;
        }

        public MemoryStream GetFileStream(CrossInhibitorRawDataModel dataModel)
        {
            var sl = new SLDocument();

            var rowIndex = 2;
            var columnIndex = 2;

            var groupHeaderStyle = _styleProvider.GetGroupHeaderStyle();
            var groupCellStyle = _styleProvider.GetGroupCellStyle();
            var lastGroupCellStyle = _styleProvider.GetGroupLastCellStyle();
            var darkGreenStyle =
_styleProvider.GetFilledCellStyle(System.Drawing.Color.FromArgb(0, 176, 79));
            var lightGreenStyle =
_styleProvider.GetFilledCellStyle(System.Drawing.Color.FromArgb(146, 208, 80));

            // Add clusters to file
            foreach (var keyValue in dataModel.Clusters)
            {
                var isInt = int.TryParse(keyValue.Key, out var number);
                var label = keyValue.Key;
                if (isInt)
                {
                    label = $"Group {number + 1}";
                }

                sl.SetCellValue(rowIndex, columnIndex, label);
                sl.SetCellStyle(rowIndex, columnIndex, groupHeaderStyle);

                rowIndex++;
                for (var i = 0; i < keyValue.Value.Count; i++)
                {
                    var item = keyValue.Value[i];
                    sl.SetCellValue(rowIndex, columnIndex, item);

                    if (i == keyValue.Value.Count - 1)
                    {
                        // Add border in the bottom to close table
                        sl.SetCellStyle(rowIndex, columnIndex, lastGroupCellStyle);
                    }
                    else
                    {
                        sl.SetCellStyle(rowIndex, columnIndex, groupCellStyle);
                    }
```

```
                rowIndex++;
            }

            columnIndex++;
            rowIndex = 2;
        }

        var intialRawDataRow = rowIndex + dataModel.Clusters.Values.ToList().Max(x =>
x.Count) + 2;

        // Add marked antigen labels to file
        rowIndex = intialRawDataRow;
        columnIndex = 3;
        foreach (var label in dataModel.MarkedAntigenLabels)
        {
            sl.SetCellValue(rowIndex, columnIndex, label);
            columnIndex++;
        }

        // Add antigen labels to file
        rowIndex++;
        columnIndex = 1;
        for (var i = 0; i < dataModel.AntigenLabels.Count; i++)
        {
            var keyValue = dataModel.Clusters.ElementAt(i);
            var isInt = int.TryParse(keyValue.Key, out var number);
            var groupLabel = keyValue.Key;
            if (isInt)
            {
                groupLabel = $"Group {number + 1}";
            }

            sl.SetCellValue(rowIndex, columnIndex, groupLabel);
            columnIndex++;

            var labelGroup = dataModel.AntigenLabels[i];
            foreach (var label in labelGroup)
            {
                sl.SetCellValue(rowIndex, columnIndex, label);
                rowIndex++;
            }

            rowIndex++;
            columnIndex = 1;
        }

        // Add cross inhibition indexes to file
        rowIndex = intialRawDataRow + 1;
        columnIndex = 3;
        foreach (var clusterGroup in dataModel.CrossInhibitionIndexes)
        {
            foreach (var row in clusterGroup)
            {
                foreach (var cell in row)
                {
                    sl.SetCellValue(rowIndex, columnIndex, cell.Value);
                    switch (cell.MarkerColor)
                    {
                        case InhibitionColors.DarkGreen:
                            sl.SetCellStyle(rowIndex, columnIndex, darkGreenStyle);
                            break;
                        case InhibitionColors.LightGreen:
                            sl.SetCellStyle(rowIndex, columnIndex, lightGreenStyle);
```

```
                            break;
                        default:
                            break;
                    }

                    columnIndex++;
                }

                rowIndex++;
                columnIndex = 3;
            }

            rowIndex++;
        }

        using var stream = new MemoryStream();
        sl.SaveAs(stream);

        return stream;
    }
  }
}


using DocumentFormat.OpenXml.Spreadsheet;
using SpreadsheetLight;
using Xema.Services.Infrastructure;

namespace Xema.Services.Provider
{
    public class StyleProvider : IStyleProvider
    {
        public SLStyle GetGroupHeaderStyle()
        {
            var style = new SLStyle();
            style.SetTopBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetLeftBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetRightBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetBottomBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetHorizontalAlignment(HorizontalAlignmentValues.Center);

            return style;
        }

        public SLStyle GetGroupCellStyle()
        {
            var style = new SLStyle();
            style.SetLeftBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetRightBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);

            return style;
        }

        public SLStyle GetGroupLastCellStyle()
        {
            var style = new SLStyle();
            style.SetLeftBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
```

```
            style.SetRightBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);
            style.SetBottomBorder(BorderStyleValues.Thin,
SLThemeColorIndexValues.Dark1Color);

            return style;
        }

        public SLStyle GetFilledCellStyle(System.Drawing.Color color)
        {
            var style = new SLStyle();
            style.Fill.SetPattern(PatternValues.Solid, color, color);

            return style;
        }
    }
}
```