Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ КЛАСТЕРИЗАЦІЇ ЧАСОВИХ РЯДІВ У ВИГЛЯДІ МЕДИЧНИХ ДАНИХ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2

Касумов А.І.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник асист. Кобилін І.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Касумов Артур Імранович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку для кластеризації часових рядів у вигляді медичних даних

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мова програмування Python, середовище розробки PyCharm, бібліотека для математичних обчислювань NumPy, бібліотека для графічного відображення даних Matlotlib, бібліотека для обробки та аналізу даних Pandas, бібліотека для створення графічного інтерфейсу PyQt.

4. Перелік питань, що потрібно опрацювати в роботі
1. Провести дослідження видів та особливостей послідовних даних.
2. Оцінити проблеми, пов'язані з кластеризацією часових рядів.
3. Ознайомитись з алгоритмами кластеризації.
4. Розробити застосунок для кластеризації.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми кластеризації часових рядів, постановка задачі, тестові зображення, методи кластеризації, розробка комп'ютерної моделі, аналіз результатів медичних даних.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-16.04.23	
3	Аналіз літератури з досліджуваної проблеми	17.04.23-20.04.23	
4	Аналіз технічних засобів	21.04.23-03.05.23	
5	Розробка методів	04.05.23-17.05.23	
6	Програмна реалізація	18.05.23-23.05.23	
7	Оформлення пояснювальної записки	24.05.23-26.05.23	
8	Перевірка на плагіат	27.05.23	
9	Рецензування	28.05.23	
10	Підготовка презентації та доповіді	29.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	05.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____асист. Кобилін І.О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 64 с., 57 рис., 35 джерел.

ЧАСОВИЙ РЯД, ЗАДАЧА КЛАСТЕРИЗАЦІЇ, МЕТОД *K*-СЕРЕДНІХ, PYTHON, ПОСЛІДОВНІСТЬ ДАНИХ.

Об’єктом роботи є послідовність даних у вигляді часового ряду.

Метою роботи є розробка застосунку з різними методами кластеризації, що базуються на використанні часових рядів, які дозволяють розподілити вхідні дані на кластери – групи однотипних екземплярів вибірки для подальшого прогнозування.

Проведено дослідження видів та особливостей послідовних даних, а також було вивчено основні проблеми, пов’язані з кластеризацією часових рядів. В рамках роботи було розглянуто загальний процес кластеризації часових рядів, включаючи основні алгоритми кластеризації, різноманітні методи отримання статистичних характеристик і їх вплив на ефективність кластеризації.

У результаті роботи здійснена програмна реалізація системи для кластеризації часових рядів.

TIME-SERIES DATA, CLUSTERING, *K*-MEANS METHOD, PYTHON, DATA STREAMS.

The object of the research is a sequence of data in the form of a time series.

The purpose of the research is to develop an application with various clustering methods based on the time series, which allow dividing the input data into clusters – groups of samples of the same type for further forecasting.

The types and features of sequential data were studied, and the main problems related to the clustering of time series were studied. As part of the work, the general process of time series clustering was considered, including the main clustering algorithms, various methods of obtaining statistical characteristics and their impact on clustering efficiency.

As a result, software for time series clustering has been implemented.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Огляд стану проблеми і постановка завдання 9
1.1 Види часових рядів та їх властивості 9
1.2 Проблеми кластеризації часових рядів 11
1.3 Методи ієрархічної кластеризації 13
1.4 Неієрархічний чіткий метод кластеризації *k*-середніх 16
1.5 Неієрархічний нечіткий метод кластеризації *c*-середніх 18
1.6 Постановка задачі 21
2 Математичні моделі фільтрації зображень 22
2.1 Модель масштабування даних 22
2.2 Модель кластеризації даних методом *k*-середніх 24
2.3 Модель вибору кількості кластерів 27
2.4 Модель кластеризації даних DTW метрикою 29
2.5 Модель кластеризації даних методом нечітких *c*-середніх 32
3 Комп’ютерна модель кластеризації часових рядів 35
3.1 Обґрунтування вибору середовища програмної реалізації 35
3.1.1 NumPy 37
3.1.2 Pandas 37
3.1.3 Matlotlib 38
3.1.4 Scikit-learn 39
3.1.5 PyQT 39
3.2 Вхідні дані 40
3.3 Програмна реалізація 42
3.3.1 Розробка інтерфейсу 42
3.3.2 Зчитування даних 46
3.3.3 Відображення усіх елементів з файлу 48
3.3.4 Кластеризація часового ряду 49

3.3.5 Метод «ліктя» 52
3.3.6 Нормалізація даних 53
3.3.7 Відображення центроїдів 54
3.3.8 Елементи в кластерах 55
3.4 Тестування розробленої моделі 57
Висновки 60
Перелік джерел посилання 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЦВТ – центральний венозний тиск

ВЧТ – внутрішньочерепний тиск

FCM – Fuzzy *C*-means (нечіткі *c*-середні)

DTW – Dynamic Time Warping (динамічне вирівнювання в часі)

WSCC – Within-Cluster Sum Of Squares (сума квадратів усередині кластера)

ВСТУП

Штучний інтелект в цілому, машинне навчання та можливості прогнозування, що розвиваються завдяки йому, відкриває унікальну можливість для поліпшення та підвищення якості процесів у різних галузях.

Особливо корисною є технологія кластеризації, яка успішно використовується в будь-якій сфері, де необхідно аналізувати дані залежностей між сигналами та минулими значеннями, виявляти порушення залежностей та відхилення від патернів нормальної поведінки. Завдяки передбачувальній аналітиці розвивається й інженерія – наприклад, обладнання допрацьовується, щоб використовувати його можна було без або за мінімального контролю з боку людини.

Сьогодні медицина займає найважливіше місце для людини, тому що здоров’я людини є найціннішим, і за ним треба слідкувати, як за автомобілем, щоб проїхати якомога більше. Саме тому була обрана сфера медицини, бо треба робити цей світ кращим. Після операції пацієнти зазвичай перебувають під медичним наглядом, щоб забезпечити оптимальний післяопераційний період і уникнути можливих ускладнень. З деякими інтервалами у неї беруться різні показники, такі як систолічний тиск, діастолічний тиск, температур, центральний венозний тиск, внутрішньочерепний тиск та інші. Якщо показники повертаються до норми, то людину виписують, інакше продовжують стежити за нею, до тих пір, поки показники не будуть в нормі.

У зв’язку з динамічною обробкою інформації та необхідністю аналізу часових рядів в інтелектуальному аналізі потоків даних, потрібно оцінити проблему кластеризації часових рядів та існуючі підходи до її вирішення. Для цього треба проаналізувати існуючі підходи з використанням стандартних метрик кластеризації часових рядів та їх обробки.

1 ОГЛЯД СТАНУ ПРОБЛЕМИ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Види часових рядів та їх властивості

Часовий ряд – послідовність спостережень, які протікають та вимірюються у певному часовому проміжку. Основна характеристика, яка відрізняє часовий ряд від простої вибірки даних – вказаний час вимірювання або номер зміни порядку.

Часові ряди використовуються для аналітики та прогнозування, коли важливо визначити, що відбуватиметься з показниками в найближчу годину, день, місяць або рік. Наприклад, скільки користувачів завантажують за день мобільний застосунок. Показники часових рядів можуть бути як технічними, так й економічними, соціальними і навіть природними [1].

Залежно від характеру часового параметра розрізняють моментні та інтервальні часові ряди. Моментний ряд характеризує значення показника на певний момент часу. Зокрема, моментними є ряди цін на різноманітні товари станом на певні моменти часу, курсів валют, кількості населення, маси об’єкта, електричного напруження у мережі, температури хворого та інше. В інтервальних рядах рівні характеризують значення показника за певні періоди (інтервали) часу. Прикладами є ряди динаміки виробництва продукції (річного, квартального, місячного, добового), міграції населення, прибутку підприємства, валового національного доходу, вимірювання показників хворої людини та інші [1].

Особливістю інтервальних рядів є можливість підсумовування їх рівнів. Результатом цього є так звані нагромаджені підсумки, які є значеннями певного показника за відповідний період часу. Наприклад, підсумовуючи добові обсяги виробництва продукції протягом місяця, отримаємо загальний обсяг її виробництва за відповідний місяць.

Окремі спостереження часового ряду називають його рівнями або елементами. Кожний рівень часового ряду відповідає певному моменту часу.

Якщо кожному моменту часу відповідають значення лише одного показника, ряд є одновимірним. В іншому випадку він є багатовимірним [2].

Існують повні та неповні часові ряди, які відрізняються наявністю пропущених значень. Повні часові ряди містять значення для всіх моментів часу в певному періоді. У таких рядах немає пропущених значень або вони доповнюються шляхом заповнення пропусків, наприклад, за допомогою інтерполяції або заповненням попередніми значеннями. Неповні часові ряди містять пропущені значення для деяких моментів часу в послідовності даних. Це може статися через різні причини, наприклад, технічні проблеми, відсутність збору даних в певні періоди або втрату даних.

На спостереження за показниками та їх систематизацію впливають тенденції та сезонні ефекти. Від умов залежить складність моделювання системи прогнозування. Інтервальні ряди діляться за наявністю чи відсутністю тенденцій та сезонних ефектів на стаціонарні та нестаціонарні.

У стаціонарних часових рядах статистичні властивості не залежать від часу, тому результат легко передбачити. Більшість статистичних методів припускають, що всі тимчасові ряди мають бути стаціонарними. Приклад стаціонарних часових рядів – народжуваність в Україні. Звичайно, вона залежить від багатьох факторів, але її зменшення або збільшення можна передбачити: у народжуваності немає яскраво вираженої сезонності [3].

У нестаціонарних часових рядах статистичні властивості змінюються з часом. Вони відображають сезонні ефекти, тренди та інші структури, які залежать від тимчасового показника. Приклад – міжнародні перельоти авіакомпаній. Кількість пасажирів у ті чи інші напрямки змінюються залежно від сезонності [3].

1.2 Проблеми кластеризації часових рядів

На сьогодні методи кластеризації для обробки часових рядів мають значні недоліки, такі як чутливість до початкових параметрів, відсутність універсального підходу, висока вартість обробки даних, велика обчислювальна складність та інше. Однак метою кластеризації часових рядів є виявлення невідомих закономірностей у ряді, оскільки ряди є неперіодичними і не мають однозначної інтерпретації значень. Тому одним з основних завдань в області інтелектуального аналізу даних є кластеризація, тобто розбиття вихідних даних на однорідні групи в режимі навчання без вчителя (самонавчання).

Кластеризація – це процес об’єднання об’єктів у групи, члени яких подібні до певної міри, а кластер – це безліч об’єктів, близьких між собою у смислі деякої міри схожості. На рисунку 1.1 зображено графічний приклад, що обумовлює необхідність кластеризації часових рядів, а на рисунках 1.2, 1.3 та 1.4 зображено можливі результати кластеризації.

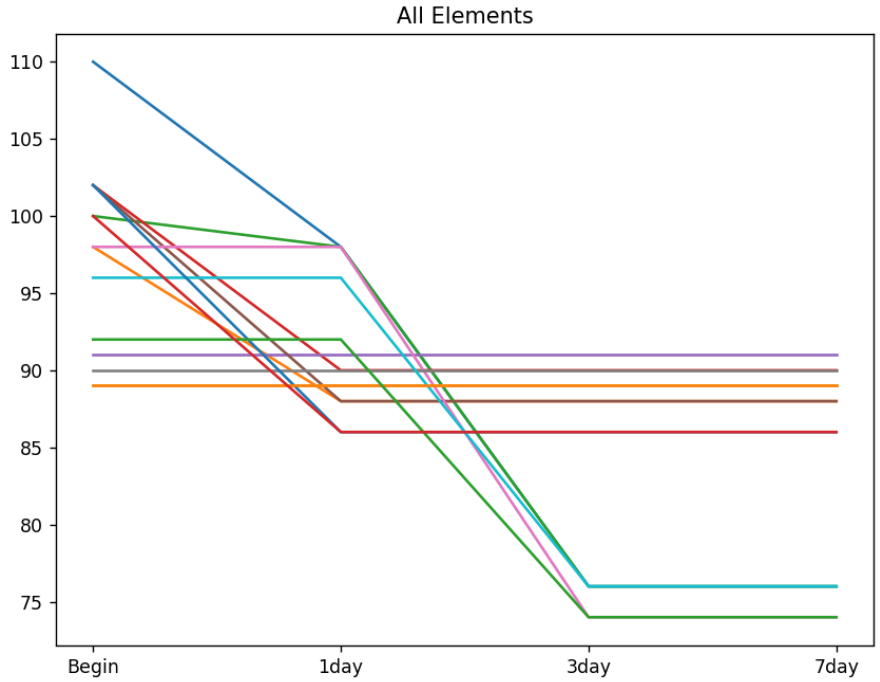

Рисунок 1.1 – Графічна інтерпретація часових рядів, що подаються на обробку

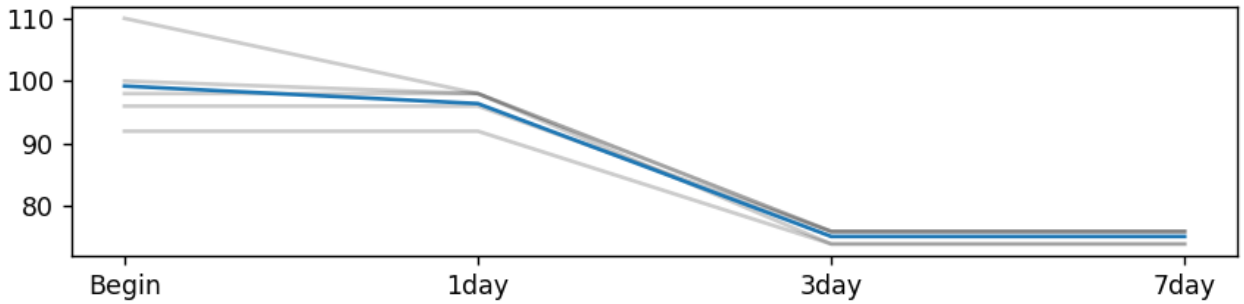

Рисунок 1.2 – Графічна інтерпретація результату кластеризації часових рядів по кластеру 1

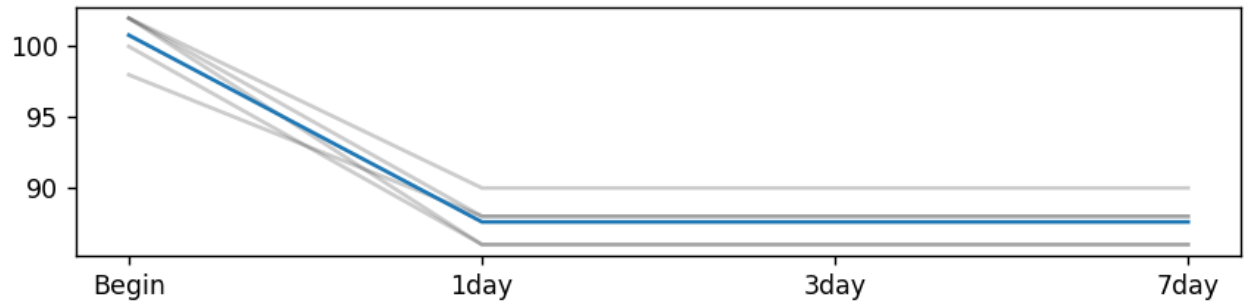

Рисунок 1.3 – Графічна інтерпретація результату кластеризації часових рядів по кластеру 2

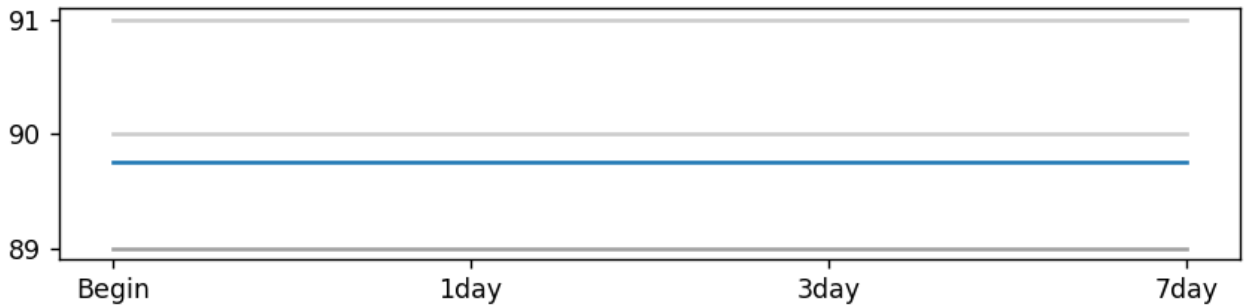

Рисунок 1.4 – Графічна інтерпретація результату кластеризації часових рядів по кластеру 3

Кластеризація – це одна з найбільш важливих проблем неконтрольованого навчання, що стикається з виявленням прихованої структури в наборі даних. Організація об’єктів у групи з подібними характеристиками, яку називають об’єднанням в кластери, є процесом, що допомагає у розумінні цієї структури (рис. 1.5) [4].

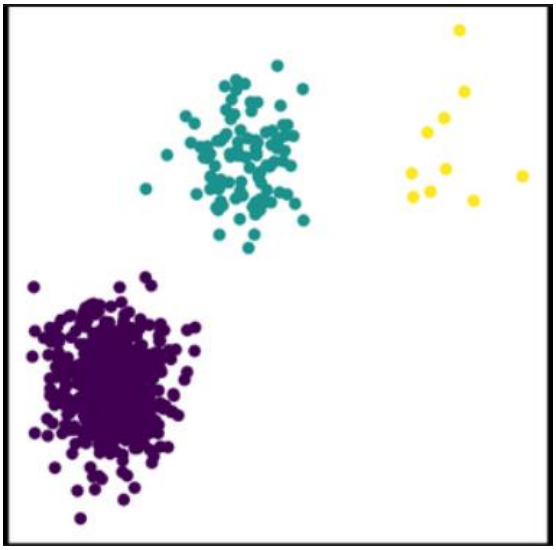

Рисунок 1.5 – Кластеризація

Часові ряди, як набір послідовних спостережень, наряду з великим обсягом мають ще певні особливості. До них можна віднести: дані можуть мати велику кількість унікальних значень, можуть бути зашумлені викидами, що вказує на наявність аномалій, що є недоліком для використання більшості статистичних методів аналізу даних [5]. Часові ряди можуть мати різну довжину, що створює виклики при порівнянні та кластеризації. Класичні алгоритми кластеризації можуть потребувати однакової довжини вхідних даних [6-8].

1.3 Методи ієрархічної кластеризації

Методи кластеризації розбиваються на два класи: ієрархічні та неієрархічні.

Ієрархічна кластеризація – сукупність алгоритмів упорядкування даних, спрямованих на створення ієрархії дерева вкладених кластерів, так звану дендрограму, зображену на рисунку 1.6. Ієрархічні алгоритми будують не одне розбиття вибірки на кластери, що не перетинаються, а систему вкладених

розбиттів. На виході ми отримуємо дерево кластерів, корінням якого є вся вибірка, а листями – найдрібніші кластери. Найнижчий рівень дерева складається з кожної точки у своєму кластері.

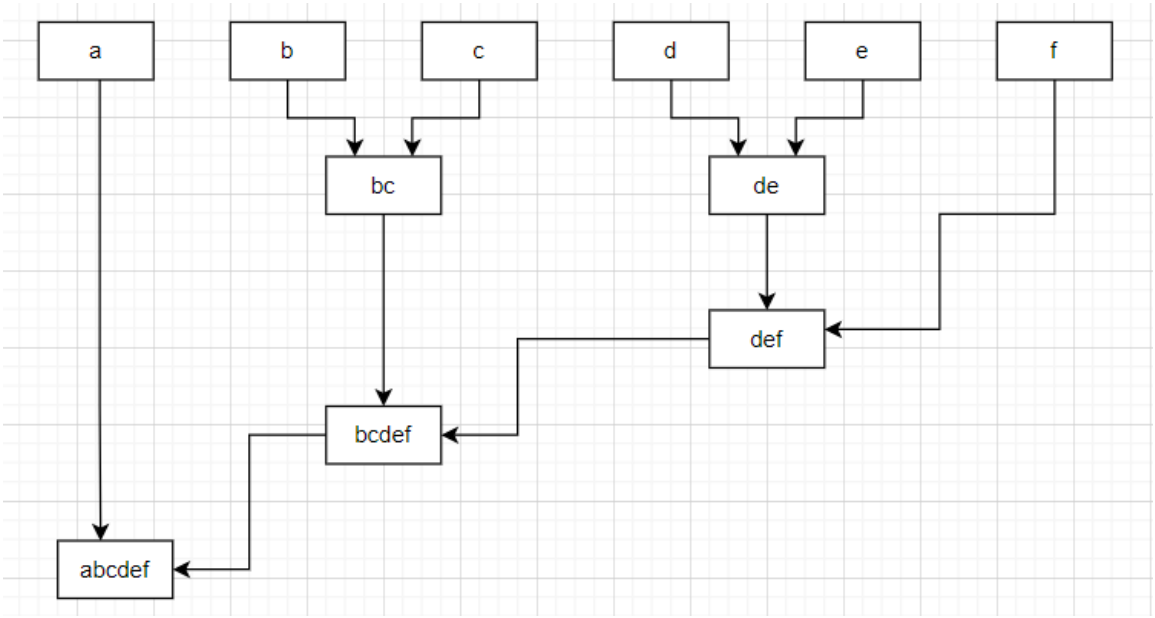

Рисунок 1.6 – Дендрограма

Неієрархічні алгоритми, або ж плоскі алгоритми будують одне розбиття об’єктів на кластери (фіксована кількість кластерів). Ієрархічне представлення дуже зручне, коли потрібно отримати інформацію про різні рівні кластерної структури або коли точна кількість кластерів, які треба відшукати, невідома.

Виділяють два класи методів ієрархічної кластеризації (рис. 1.7):

– агломеративні методи. Нові кластери створюються шляхом об’єднання дрібніших кластерів і, таким чином, дерево створюється від листя до стовбура;

– дивізивні методи. Нові кластери створюються шляхом поділу більших кластерів на дрібніші і, таким чином, дерево створюється від стовбура до листя.

Зазвичай під терміном дендрограма (рис. 1.6), мається на увазі дерево, яке будується на основі матриці мір близькості. Дендрограма дозволяє візуалізувати взаємозв’язки між об’єктами з певної множини. Для створення

дендрограми потрібна матриця подібності, яка визначає рівень подібності між парами кластерів.

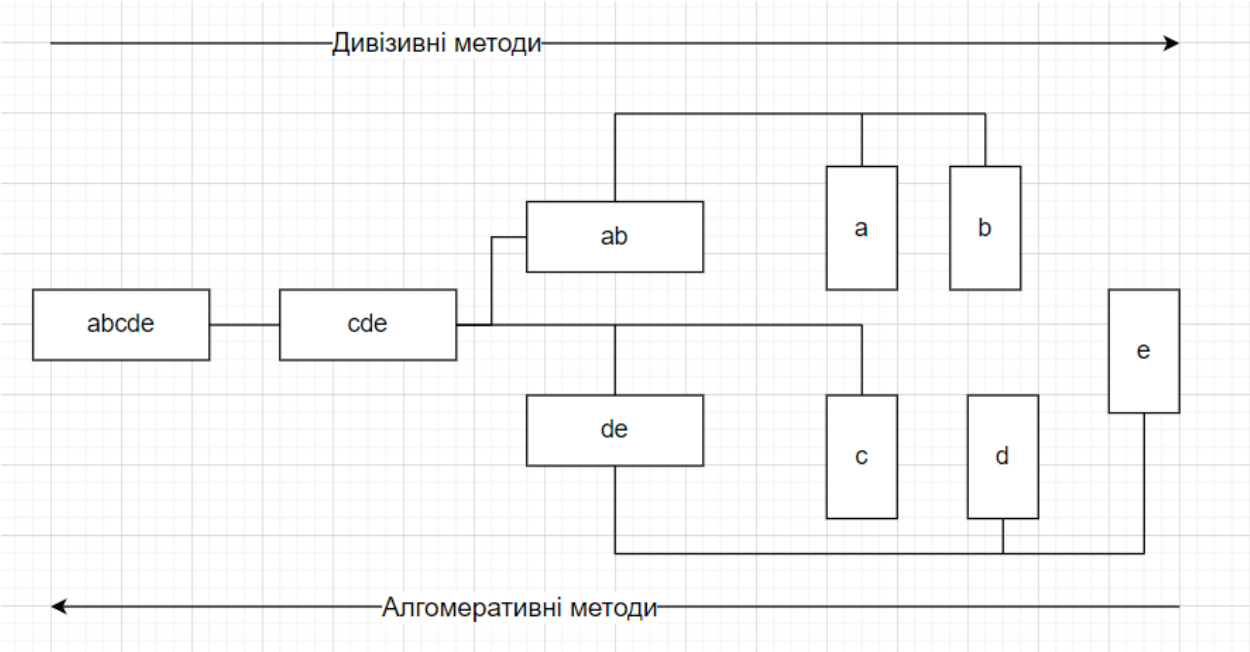

Рисунок 1.7 – Класи методів ієрархічної кластеризації

У разі використання ієрархічних алгоритмів постає питання, як поєднувати між собою кластери та як обчислювати схожість між об’єктами.

Існує кілька методів:

– метод одиночного зв’язку. Алгоритм починається з пошуку двох найбільш близьких об’єктів, пара яких утворює первинний ключ. Кожний наступний об’єкт приєднується до того кластеру, до одному із об’єктів яких він ближче [9].

$$min\,\{d(a,b) : a \in A, b \in B\}, \qquad (1.1)$$

де $d(a,b)$ – відстань між елементами a і b, які належать до кластера A та B;

– метод повного зв’язку. Правило об’єднання цього метода має на увазі, що новий об’єкт приєднується к тому кластеру, самий далекий елемент

якого знаходиться ближче до нового об’єкта, ніж самі далекі елементи інших кластерів [9].

$$max\,\{d(a,b) : a \in A, b \in B\}, \tag{1.2}$$

де $d(a,b)$ – відстань між елементами a і b, які належать до кластера A та B;

– метод середнього зв’язку. На кожному кроці рахується середнє арифметична відстань між кожним об’єктом із одного кластера і кожним об’єктом із другого кластера. Об’єкт приєднується до кластера, якщо ця середня відстань менша, ніж середня відстань до любого іншого кластера. Цей метод повинен давати більш точні результати [9].

$$\frac{1}{|A| \times |B|} \sum_{a \in A} \sum_{b \in B} d(a,b), \tag{1.3}$$

де $d(a,b)$ – відстань між елементами a і b, які належать до кластера A та B;

$|A|$ та $|B|$ – потужності кластерів.

1.4 Неієрархічний чіткий метод кластеризації *k*-середніх

Метод *k*-середніх реалізує неієрархічний алгоритм, тобто при кластеризації ми задаємо кількість кластерів. Також неієрархічні алгоритми поділяються на чіткі та нечіткі. Чіткі алгоритми кожному об’єкту вибірки ставлять у відповідність номер кластера, тобто кожен об’єкт належить лише одному кластеру. Нечіткі алгоритми кожному об’єкту ставлять у відповідність набір речових значень, що показують ступінь ставлення об’єкта до кластерів. Тобто кожен об’єкт відноситься до кожного кластера з деякою ймовірністю. Метод *k*-середніх відноситься до чіткого алгоритму [2].

Алгоритм *k*-середніх – це один з найпоширеніших методів кластеризації даних, який дозволяє групувати дані на основі схожості між ними. Алгоритм полягає у визначенні *k* кластерів, в які будуть розподілені дані, де *k* – це задана заздалегідь кількість кластерів. Метод *k*-середніх поділяє *m* спостережень на *k* кластерів $S = \{S_1, S_2, \dots, S_k\}$, щоб мінімізувати сумарне квадратичне відхилення точок кластерів від центроїдів цих кластерів [10].

Алгоритм починається з вибору кількох випадкових точок в множині даних в якості центрів кластерів. Далі для кожної точки даних визначається найближчий центр кластеру і вона додається до відповідного кластеру. Потім розраховуються нові центри кластерів шляхом обчислення середнього значення всіх точок кожного кластеру. Ці кроки повторюються доти, поки центри кластерів не перестануть змінюватись (рис. 1.8).

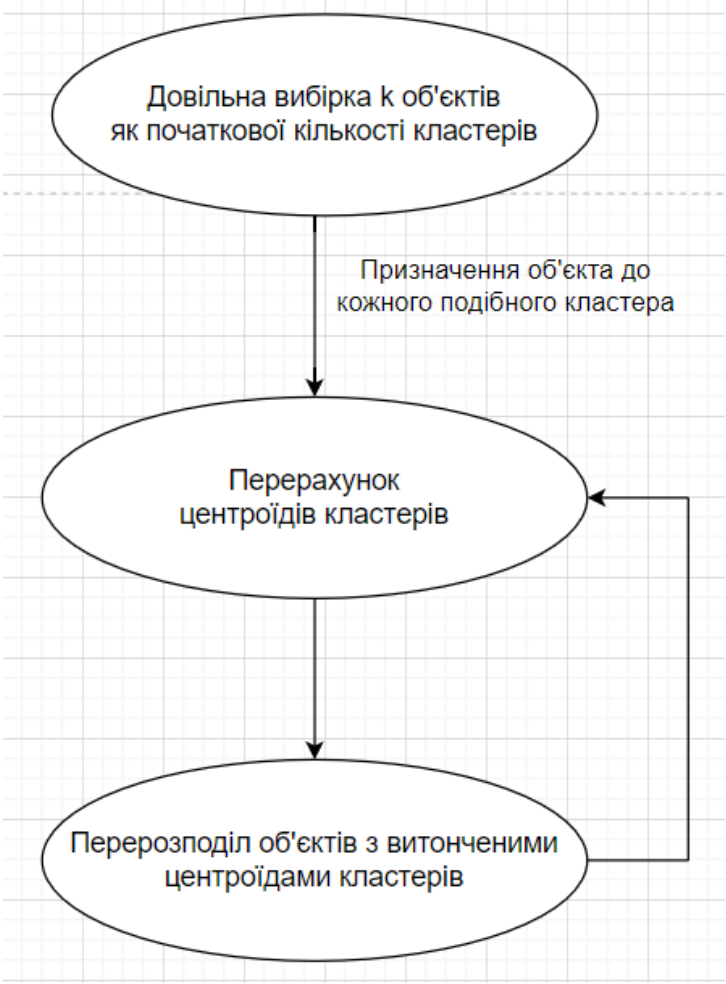

Рисунок 1.8 – Приклад роботи алгоритму *k*-середніх

Однак, важливо враховувати, що результати алгоритму можуть залежати від випадкової ініціалізації початкових центрів кластерів. Для вибору оптимальної кількості кластерів можна використовувати методи оцінки кількості кластерів, такі як метод ліктя або метод сітки пошуку.

Однією з основних переваг k-середніх є його швидкість та ефективність при роботі з великими наборами даних, що робить його популярним для багатьох задач машинного навчання. Крім того, алгоритм має досить просту реалізацію та візуалізацію результатів, що дозволяє легко зрозуміти результати кластеризації та зробити відповідні висновки.

Проте, метод k-середніх має свої недоліки. Він дуже чутливий до початкових значень центрів кластерів та може зійти з місця при наявності викидів або шумів у даних. Крім того, алгоритм працює краще на даних, що мають виражені кластери з невеликою кількістю перетинів.

Складністю алгоритму k-середніх є $O(k \cdot N \cdot r \cdot D)$, де k – кількість очікуваних кластерів, N – кількість об’єктів, що підлягають кластеризації, r – число ітерацій для досягнення кластеризації, D – розмірність простору об’єктів [11].

У загальному, k-середніх є потужним та ефективним методом кластеризації даних, який може бути застосований в багатьох задачах машинного навчання та аналізу даних. Однак, при використанні цього методу необхідно враховувати його переваги та недоліки та вибрати оптимальну конфігурацію для конкретної задачі.

1.5 Неієрархічний нечіткий метод кластеризації c-середніх

Для класичних методів кластеризації існує ефект «метелика», який характеризується тим, що кілька елементів вихідного набору даних відносяться до більше, ніж одного кластера. І тут використання нечітких

алгоритмів дозволяє побудувати ефективніші методи, ніж чіткі. Метод *c*-середніх називають Fuzzy *C*-means (FCM), що в перекладі з англійської означає нечіткі *c*-середні [12-14].

Нечітка кластеризація відіграє важливу роль у вирішенні проблем у галузях розпізнавання образів та нечіткої ідентифікації моделі. Алгоритм FCM більше підходить для даних, які більш менш рівномірно розподілені навколо кластерних центрів і дозволяє використовувати належність частини даних двом чи більше кластерам.

FCM алгоритм працює, призначаючи членство кожній точці даних, що відповідає кожному центру кластера, на основі відстані між центром кластера і точкою даних. Чим ближче дані до центру кластера, тим більша їхня приналежність до конкретного центру кластера. Сума належності кожної точки даних повинна дорівнювати одиниці [15].

Алгоритм нечітких *c*-середніх (рис. 1.9) дуже схожий на алгоритм *k*-середніх:

Крок 1. Спочатку треба задати кількість кластерів.

Крок 2. Задати матрицю приналежності (призначити коефіцієнти кожній точці даних для перебування в кластерах).

Крок 3. Вирахувати центроїди для кожного кластера.

Крок 4. Для кожної точці даних обчислити її коефіцієнти знаходження у кластерах.

Крок 5. Повторювати Крок 3 та Крок 4 до тих пір, покати алгоритм не завершиться.

Алгоритм закінчить роботу у разі, коли виконання заданого заздалегідь деякого кінцевого числа ітерацій закінчиться, або коли мінімальна абсолютна різниця між значеннями функцій приналежності на двох послідовних ітераціях не стане меншою за деяке заздалегідь задане значення ε.

Алгоритм FCM заснований на мінімізації функції [16]:

$$J_m = \sum_{i=1}^{m}\sum_{j=1}^{c} u_{i,j}^{m} d^2(x_i, c_j)\,, \qquad (1.4)$$

де $m(1 \le m \le \infty)$ – будь-яке дійсне число, більше 1, яке характеризує нечітку міру належності елемента кластеру;

$u_{i,j}$ – ступінь приналежності елемента x_i j-му кластеру;

c_j – центр кластер;

$d(x, c)$ – будь-яка норма, що виражає подібність між будь-якими вимірюваними даними.

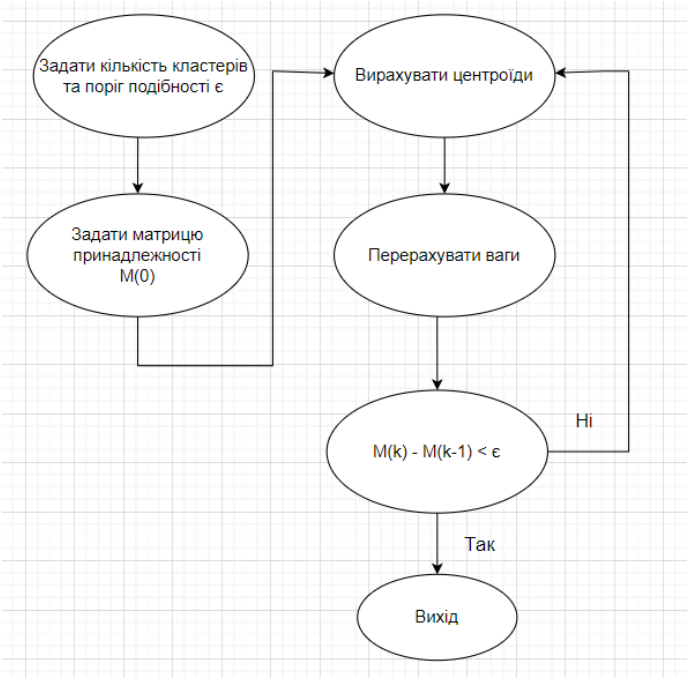

Рисунок 1.9 – Алгоритм обчислення нечіткої кластеризації

До переваг FCM метода можна віднести:

– на відміну від *k*-середніх, де точка даних має належати виключно одному кластера, тут точці даних призначається приналежність до кожного кластеру [17];

– дає найкращий результат для набору даних, що перекривається, і порівняно кращий, ніж алгоритм *k*-середніх.

До недоліків можна віднести:

– потрібно заздалегідь визначити кількість кластерів, як і в методі *k*-середніх;

– за меншого значення ε отримуємо кращий результат, але за рахунок більшої кількості ітерацій.

1.6 Постановка задачі

Таким чином, кластеризація часових рядів у вигляді медичних даних є актуальним завданням на сьогоднішній день для обробки і розпізнавання різних патернів. Тому ставиться завдання у здійсненні програмної реалізації системи для кластеризації часових рядів з різними методами кластерізації.

Об’єктом роботи є послідовність даних у вигляді часового ряду.

Метою роботи є розробка застосунку з різними методами кластеризації, що базуються на використанні часових рядів, які дозволяють розподілити вхідні дані на кластери – групи однотипних екземплярів вибірки для подальшого прогнозування.

Для досягнення мети необхідно вирішити такі завдання:

– провести дослідження видів та особливостей послідовних даних;

– оцінити проблеми, пов’язані з кластеризацією часових рядів;

– ознайомитись з алгоритмами кластеризації.

2 МАТЕМАТИЧНІ МОДЕЛІ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ

2.1 Модель масштабування даних

Масштабування даних перед кластеризацією є важливим кроком, який допомагає покращити якість кластеризації та зробити її більш об’єктивною. Масштабування даних означає приведення значень кожної ознаки в наборі даних до одного і того ж діапазону значень, зазвичай до інтервалу між 0 та 1 або до середнього значення 0 та стандартного відхилення 1.

Оскільки кластеризація заснована на відстанях між точками даних, масштабування даних допомагає уникнути спотворення результатів кластеризації через різницю в діапазонах значень між різними ознаками. Наприклад, якщо у наборі даних є ознака, яка має значення в діапазоні від 0 до 500, і інша ознака, яка має значення в діапазоні від 0 до 5, то відстань між точками буде домінуючою за першою ознакою, що може призвести до неправильної кластеризації. Масштабування даних знижує цей вплив та допомагає отримати більш точні та коректні результати кластеризації.

Масштабування часових рядів може включати різні методи. Одним з таких методів є міні-максне масштабування, де кожне значення часового ряду масштабується в діапазоні від 0 до 1, що відповідає мінімальному та максимальному значенню часового ряду [18].

$$\grave{x_i} = \frac{x_i - min(x)}{max(x) - min(x)}\ , \tag{2.1}$$

де x_i – елемент з часового ряду;

$min(x)$ – мінімальне значення часового ряду;

$max\ (x)$ – максимальне значення часового ряду.

Інший метод масштабування, який використовується в роботі з часовими рядами, – це стандартизація. Після стандартизації значення даних

матимуть нульове середнє значення та одиничне стандартне відхилення. Саме цей метод використовується у кваліфікаційній роботі.

$$x_i^{`} = \frac{x_i - \mu}{\sigma}, \tag{2.2}$$

де x^i – елемент часового ряду;

μ – середнє значення часового ряду;

σ – стандартне відхилення часового ряду.

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{2.3}$$

де n – кількість елементів в часовому ряді;

x_i – елемент з часового ряду.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}, \tag{2.4}$$

де n – кількість елементів в часовому ряді;

x_i – елемент з часового ряду;

μ – середнє значення часового ряду.

Стандартизація – метод масштабування даних, як зазначено в (2.2), при якому дані перетворюються таким чином, щоб їхнє середнє значення дорівнювало 0, а стандартне відхилення дорівнювало 1. Цей метод полягає у тому, що з кожного значення даних віднімається їхнє середнє значення, як зазначено в (2.3), та отриманий результат ділиться на стандартне відхилення, як зазначено в (2.4).

2.2 Модель кластеризації даних методом *k*-середніх

Метод *k*-середніх – це метод кластерного аналізу, мета якого є поділ m спостережень (з простору R^n) на k кластерів, при цьому кожне спостереження відноситься до того кластера, до центру (центроїду) якого воно найближче. Особливістю методу *k*-середніх є те, що як метрика (відстань між двома об'єктами) використовується евклідова відстань, число кластерів наперед не відоме і вибирається заздалегідь, а якість кластеризації залежить від початкового розбиття [10]. Нехай у n вимірному просторі задані дві точки: $x(x_1, x_2, \dots, x_n)$ та $y(y_1, y_2, \dots, y_n)$, тоді евклідова відстань між ними розраховується наступним чином:

$$\rho(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,. \tag{2.5}$$

Основна мета методу *k*-середніх полягає у тому, щоб мінімізувати сумарне квадратичне відхилення точок кластерів від центроїдів цих кластерів.

$$\min \left[\sum_{i=1}^{k} \sum_{x \in S^i} \rho(x, \mu_i)^2 \right], \tag{2.6}$$

де k – кількість кластерів;

S_i – набір елементів у кластері;

μ_i – центроїд для кластера S_i.

Алгоритм методу *k*-середніх полягає у наступному:

Крок 1. Вибрати кількість кластерів. $Y = \{1,2, \dots, n\}$ – номера кластерів.

Крок 2. Треба встановити центроїди кластерів. Центроїди кластерів вибираються випадково або за певним правилом, наприклад, можна вибрати n перших об’єктів і взяти їх за центроїди.

Крок 3. Віднести елементи до кластерів. Кожний елемент x_i відноситься до найближчого центроїда μ_a , $a \in Y$ відповідно до вибраної метрики.

$$a_i = arg\, min\, \rho(x_i, \mu_a)\,, \tag{2.7}$$

де ρ – евклідова відстань;

x_i – певний елемент;

μ_a – центроїда.

Крок 4. Робимо перерахунок розташування центроїдів.

$$\mu_a = \frac{1}{S_a} \sum_{x^a \in S_a} x^a\,, \tag{2.8}$$

де S_a – кількість елементів у певному кластері;

x^a – елемент, що належить до певного кластеру.

Крок 5. Крок 3 та Крок 4 робиться до тих пір поки положення центроїдів не буде змінюватися, тобто коли ${\mu_i}^{\text{шаг } t} = {\mu_i}^{\text{шаг } t+1}$ [10, 11].

На першій ітерації, зображеній на рисунку 2.1 можна побачити центроїди красного кольору, які були вибрані випадково. Завдяки евклідовій відстані до першого та другого кластерів були віднесені усі елементи синього та зеленого кольорів відповідно. Після перерахунку центроїдів та віднесення до них елементів маємо інший результат. Продовжуємо до тих пір центроїди не змінюють своє положення. Приклад зображений на рисунках 2.2, 2.3 та 2.4.

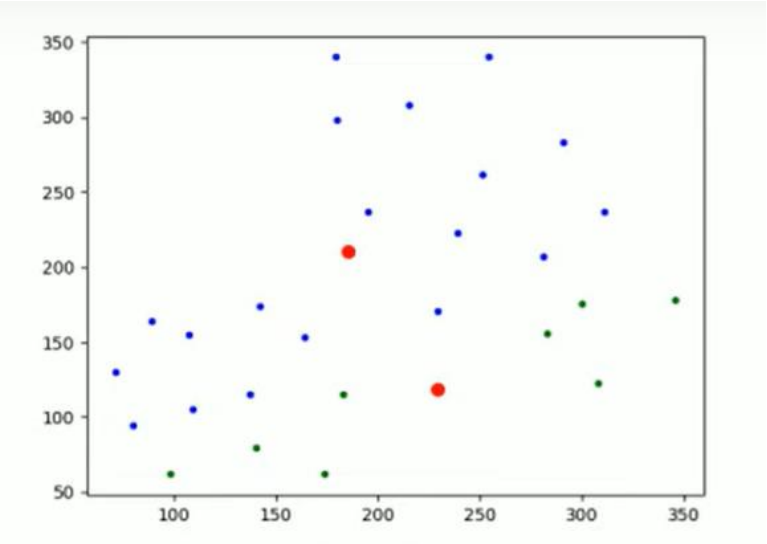

Рисунок 2.1 – Перша ітерація алгоритму *k*-середніх

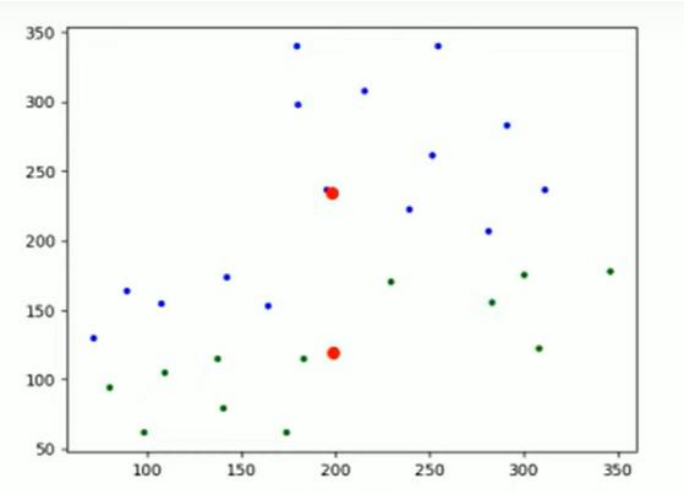

Рисунок 2.2 – Друга ітерація алгоритму *k*-середніх

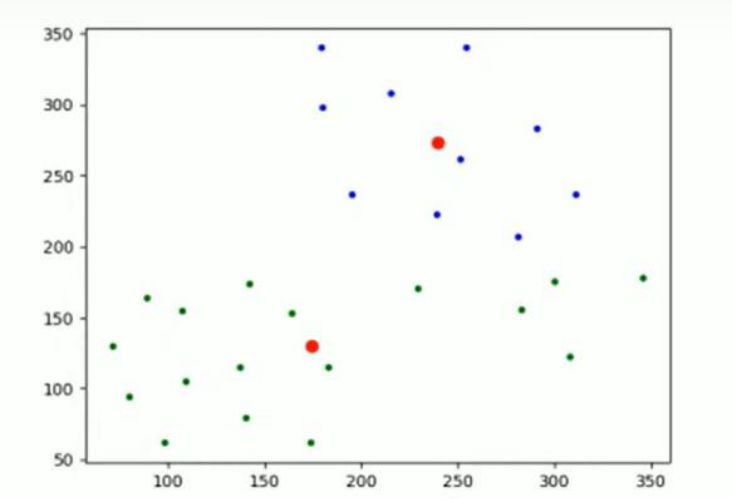

Рисунок 2.3 – Третя ітерація алгоритму *k*-середніх

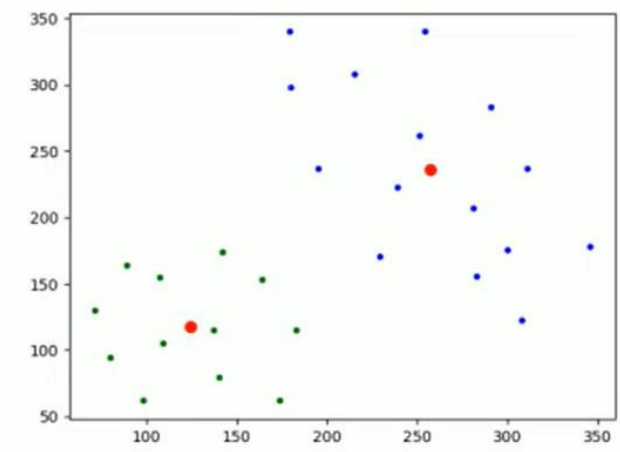

Рисунок 2.4 – Четверта ітерація алгоритму *k*-середніх

2.3 Модель вибору кількості кластерів

Використовуючи метод *k*-середніх постає питання, яку кількість кластерів треба зазначити. Саме для цього і потрібен метод ліктя.

Метод ліктя є одним з найпоширеніших методів для визначення кількості кластерів в задачі кластеризації даних. Цей метод полягає у визначенні точки вигину на графіку функції критерію кластеризації в залежності від кількості кластерів.

Для застосунку методу ліктя необхідно спочатку виконати кластеризацію даних для різної кількості кластерів, наприклад в діапазоні від 1 до 20 кластерів. Для кожної кількості кластерів обчислюється значення критерію кластеризації, яке відображає якість кластеризації. Як критерій кластеризації використовують метрику WSCC (within-cluster sum of squares). WSCC – це сума квадратів відстаней між точками і центроїдами кластерів. Для кожного часового ряду в кожному кластері обчислюється відстань між рядом та центроїдом кластеру. Ця відстань обчислюється як квадрат евклідової відстані між відповідними значеннями часового ряду та центроїдом. Після чого усі ці суми складуються та отримуємо критерій кластеризації [19, 20].

$$WSCC = \sum_{i=1}^{k} \sum_{x \in S^i} \rho(x, c_i)^2 , \quad (2.9)$$

де k – кількість кластерів;

c_i – центроїда кластера;

S^i – набір елементів, шо належать до певного кластеру;

x^j – елемент з кластеру;

ρ – евклідова відстань.

Далі потрібно побудувати графік критерію кластеризації відносно кількості кластері. На цьому графіку вісь X представляє кількість кластерів, а вісь Y – значення критерію кластеризації. На графіку (рис. 2.5) знаходиться точка, в якій відбувається зміна темпу зменшення значення критерію кластеризації. Ця точка і представляю точку вигину на графіку – «ліктя». Кількість кластерів, що відповідає цій точці, і є оптимальною кількість кластерів для даної задачі. На рисунках 2.5 та 2.6 зображені приклади використання методу ліктя. На першому рисунку можна побачити, що чотири кластери є досить оптимальним рішенням, тому що далі ми не бачимо значного покращення критерія кластеризації, а на другому – десь біля десяти кластерів буде достатньо.

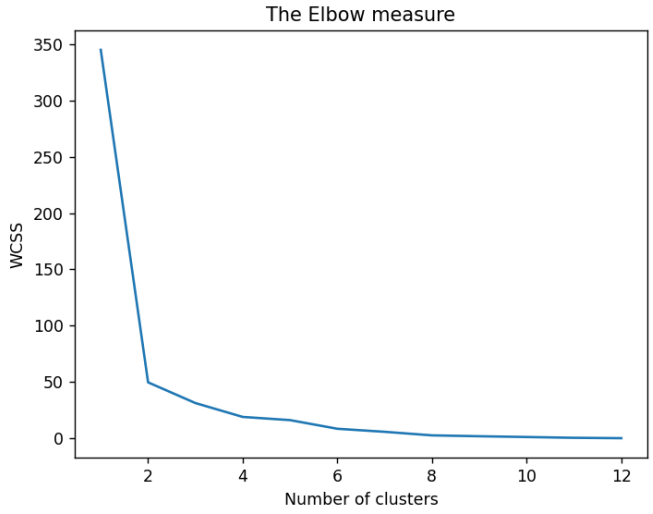

Рисунок 2.5 – Метод ліктя для кластеризації 12 елементів

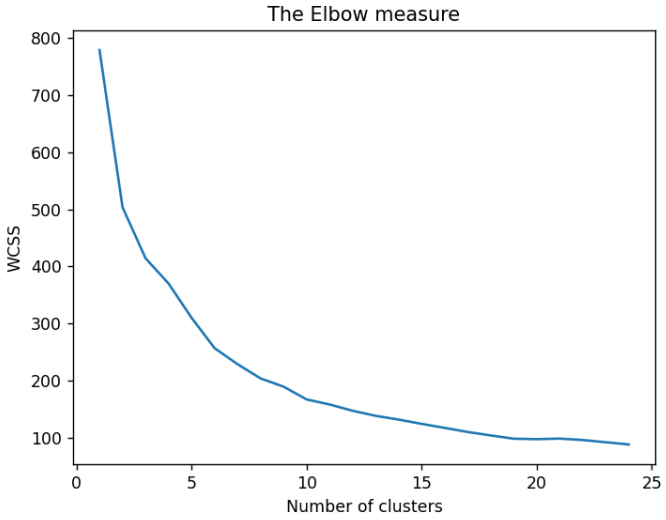

Рисунок 2.6 – Метод ліктя для кластеризації 180 елементів

Таким чином, метод ліктя допомагає визначити оптимальну кількість кластерів для задачі кластеризації даних. Він є досить простим у використані та дає можливість зекономити час та зусилля на пошук оптимальної кількості кластерів шляхом автоматизації.

2.4 Модель кластеризації даних DTW метрикою

При кластеризації часових рядів важливо враховувати той факт, що ряди можуть бути майже однаковими, але в той же час ці ряди можуть бути зміщені в часі або якщо один ряд довший за інший (рис. 2.7). У такій ситуації евклідова метрика не підходить, бо вона буде вважати, що ряди відрізняються один від одного, тому що ця метрика рахує відстань від кожної *n*-ої точки однієї послідовності до *n*-ої точки іншої. Dynamic Time Warping (DTW) алгоритм було введено для того, щоб подолати цей недолік і надати можливість вимірювати відстані між рядами, не звертаючи уваги на зрушення по часовій шкалі. На рисунках 2.7 та 2.8 зображено порівняння між евклідовою та DTW метриками.

DTW, яке означає динамічне вирівнювання в часі – це метод порівняння двох часових послідовностей, який враховує різну швидкість та зсув між ними.

Основна ідея DTW полягає в пошуку найкращого вирівнювання між двома послідовностями. Вирівнювання відбувається шляхом знаходження найкращого шляху через сітку, де кожна точка сітки представляє собою можливе вирівнювання двох елементів послідовностей. DTW дозволяє порівнювати послідовності різної довжини та враховувати їх зсув та масштабування. Це особливо корисно в ситуаціях, коли темп або тривалість подій можуть варіюватись між двома послідовностями [21].

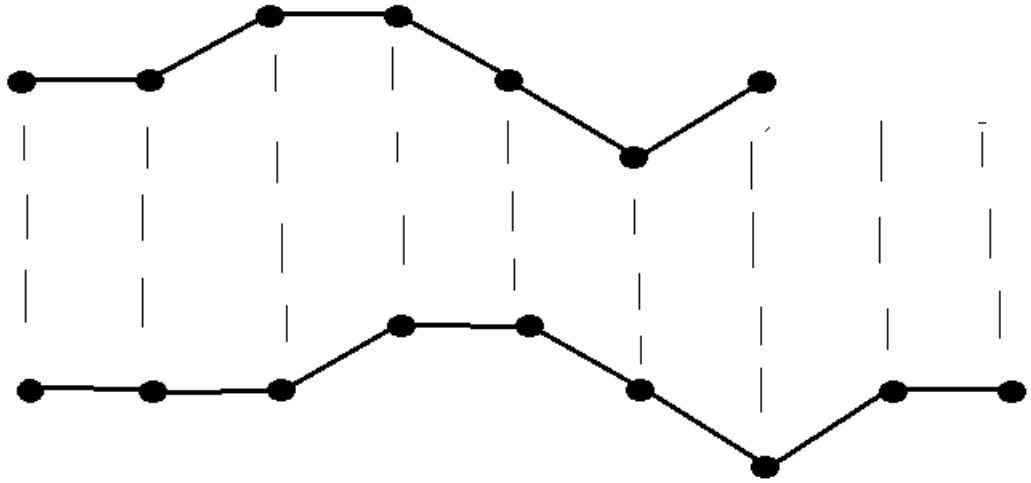

Рисунок 2.7 – Розрахунок відстані по евклідовій метриці

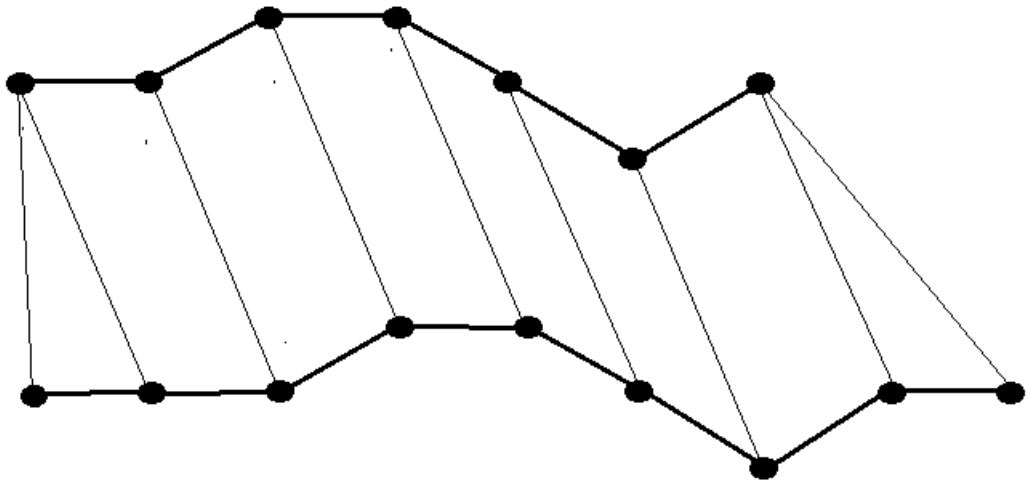

Рисунок 2.8 – Розрахунок відстані по DTW алгоритму

Алгоритм динамічного трансформування часових рядів полягає у наступному:

Припустимо маємо часові ряди: $Q = q_1, q_2, \dots, q_n$ та $\mathrm{C} = \mathrm{c}_1, \mathrm{c}_2, \dots, \mathrm{c}_m$.

Крок 1. Побудувати матрицю трансформації D порядку (n+1)×(m+1).

Крок 2. Проініціалізувати матрицю трансформації.

$$D_{0,0} = 0\,, \tag{2.10}$$

$$D_{i,0} = \infty\,, \tag{2.11}$$

де $i \in (1, n)$.

$$D_{0,j} = \infty\,, \tag{2.12}$$

де $j \in (1, m)$.

Крок 3. Заповнюємо матрицю трансформації:

$$D_{i,j} = d(x_i, y_j) + min\begin{cases} D_{i-1,j-1} \\ D_{i-1,j} \\ D_{i,j-1} \end{cases}, \tag{2.13}$$

де $d(x_i, y_j)$ – евклідова відстань між точками x_i та y_j.

Крок 4. Будуємо оптимальний шлях трансформації W від точки (n,m) до точки (1,1).

Шлях трансформації W – це набір суміжних елементів матриці, який встановлює відповідність між рядами Q та C. Він являє собою послідовність пар координат, яка представляє оптимальний шлях або вирівнювання між двома часовими послідовностями. Він вказує, які точки послідовності однієї осі відповідають точкам іншої осі. Кожен k-ий елемент визначається як $w_k = (i, j)_k$. DTW шукає найкращий шлях вирівнювання, який мінімізує

сумарну відстань між відповідними точками двох послідовностей. При побудові матриці шляху трансформації враховуються всі можливі шляхи, які можуть з'єднувати дві послідовності. Шлях, що відповідає найменшій відстані або найкращому вирівнюванню, вважається оптимальним [21].

$$DTW(Q,C) = min\left\{\frac{\sum_{k=1}^{K} d(w_k)}{K}\right\}, \quad (2.14)$$

де $d(w_k)$ – відстань між елементами.

2.5 Модель кластеризації даних методом нечітких *c*-середніх

Метод *c*-середніх – це алгоритм кластерного аналізу, який використовується для поділу набору даних на групи або кластери на основі їхньої подібності. Він є одним із найпоширеніших і найпростіших алгоритмів кластеризації. Основна ідея методу *c*-середніх полягає в тому, щоб знайти оптимальне розбиття даних на кластери таким чином, щоб мінімізувати середньоквадратичне відхилення між кожним об'єктом та центроїдом кластера [16, 17].

Нехай потрібно згрупувати відомі дані $x_i(i = 1,2,\dots,n)$, тоді алгоритм кластеризації методом *c*-середніх полягає у наступному:

Крок 1. Задаємо кількість кластерів *C*, де $2 \le C \ge n$.

Крок 2. Задаємо коефіцієнт нечіткості $m > 1$.

Крок 3. Ініціалізувати матрицю приналежності $U = \{u_{i,j}\}$ випадковим чином, так щоб $u_{i,j} \in [0,1]$ та $\sum_{j=1}^{C} u_{i,j} = 1$.

Крок 4. Визначити центри кластерів c_j:

$$c_j = \frac{\sum_{i=1}^{n} u_{i,j}^{m} x_i}{\sum_{i=1}^{n} u_{i,j}^{m}}, \tag{2.15}$$

де m – коефіцієнт нечіткості;

n – кількість елементів;

$u_{i,j}$ – ступінь приналежності елемента до кластеру.

Крок 5. Оновлюємо матрицю приналежності U:

$$u_{i,j} = \left(\sum_{k=1}^{C} \left(\frac{d(x_i, c_j)}{d(x_i, c_k)} \right)^{\frac{2}{m-1}} \right)^{-1}, \tag{2.16}$$

де m – коефіцієнт нечіткості;

C – кількість кластерів;

d – евклідова відстань.

Крок 6. Продовжуємо виконувати Крок 4 та Крок 5 до тих пір, поки виконання заданого заздалегідь деякого кінцевого числа ітерацій не закінчиться, або коли мінімальна абсолютна різниця між значеннями функцій приналежності на двох послідовних ітераціях не стане меншою за деяке заздалегідь задане значення ε.

Алгоритм FCM заснований на мінімізації функції:

$$J_m = \sum_{i=1}^{m} \sum_{j=1}^{c} u_{i,j}^{m} d^2(x_i, c_j)\,. \tag{2.17}$$

Кластеризація k-середніх також намагається мінімізувати цільову функцію, показану вище, за винятком того, що в k-середніх значення приналежності дорівнюють нулю або одиниці і не можуть набувати проміжних значень, тобто $u_{i,j} \in \{0,1\}$.

Коефіцієнт нечіткості m у методі c-середніх є важливим параметром, який визначає ступінь нечіткості чи розмитості приналежності об’єктів до кластерів. У методу k-середніх кожен об’єкт може належати тільки одному кластеру з абсолютною впевненістю. Однак у нечітких c-середніх кожен об’єкт має приналежність до кожного кластера у вигляді ймовірності або ступеня належності.

Показник m контролює ступінь розмитості та впливає на те, наскільки об’єкти можуть бути невизначеними щодо приналежності до кластерів. Великі значення m призводять до розмиття, де об’єкти можуть мати значну приналежність до кількох кластерів. Маленькі значення m роблять розбиття жорсткішим, де об’єкти належать переважно одному кластеру.

Параметр m повинен бути обраний відповідно до конкретного завдання та типу даних. Зазвичай значення m вибирається в інтервалі від 1 до нескінченності. При значеннях m, близьких до 1, розбиття стає жорстким, а при значеннях m, близьких до нескінченності, розбиття стає більш розмитим.

Вибір оптимального значення m є складним завданням і може вимагати експериментів та перевірки реальних даних [16, 17].

3 КОМП'ЮТЕРНА МОДЕЛЬ КЛАСТЕРИЗАЦІЇ ЧАСОВИХ РЯДІВ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений застосунок для кластеризації часових рядів за допомогою різних методів кластеризації. Для реалізації було обрано мову програмування Python та середовище PyCharm. Це обумовлено тим, що Python є однією з найпопулярніших мов програмування, що має великий попит на ринку, а також найкраще підходить для розробки штучного інтелекту та машинного навчання. PyCharm – це найпопулярніше кросплатформове інтегроване середовище розробки для мови програмування Python. Надає користувачеві комплекс засобів для написання коду.

Python – високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована на підвищення продуктивності розробника, читабельності коду та її якості, і навіть забезпечення переносимості застосунків. Мова є повністю об'єктно-орієнтованою в тому плані, що все виступає у ролі об'єктів [22].

Python знайшов широке застосування у різних сферах:

– аналіз даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Ця мова особливо цінна тим, що, крім великої стандартної бібліотеки, вона надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідомішими бібліотеками Python для проведення аналізу даних є pandas та NumPy. Саме ці бібліотеки використовується для написання застосунку у рамках кваліфікаційної роботи [23];

– візуалізація даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати необроблену та оброблену інформацію у більш привабливій та змістовній формі. Тут Python знову входить у гру,

пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніший з них – matplotlib, якій також використовується для написання застосунку у рамках кваліфікаційної роботи [24];

– машинне навчання. Машинне навчання є основою більшості завдань науки даних. Воно є частиною штучного інтелекту, пов’язане з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних;

– вебсерверна частина;

– розробка програмного забезпечення.

До переваг Python можна віднести:

– легкість. Python має дуже легкий синтаксис для розуміння. Його дійсно легко зрозуміти та вивчити, тому багато людей навіть рекомендують Python новачкам, як першу мову. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими мовами, такими як C/C++ або Java [25];

– інтерпретована мова. Python є мовою, яка інтерпретується. Це означає, що вона безпосередньо виконує код по рядкам. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі є кілька помилок. Це спрощує відстеження помилок;

– динамічна типізація. Python не визначає тип змінної, доки код не буде запущеним. Він автоматично надає тип даних, коли відбувається процес виконання. Розробник застосунку може не хвилюватися про оголошення типів даних для змінних [25];

– підтримка великої кількості бібліотек. Стандартна бібліотека Python є дуже великою, розробник може знайти майже усі функції, необхідні для завдання. Таким чином розробнику не потрібно залежати від зовнішніх бібліотек;

– портативність. У багатьох мовах, таких як C або C++, потрібно змінити код, для того щоб запустити програму на різних операційних

системах. З Python все інакше. Застосунок, написаний на Python, можна запускати на будь-яких операційних системах.

3.1.1 NumPy

NumPy – це бібліотека Python, яку застосовують для математичних обчислень, починаючи з базових функцій і закінчуючи лінійною алгеброю. NumPy додає підтримку роботи з масивами та матрицями різних розмірностей, разом із великою кількістю високорівневих і дуже швидких математичних функцій для операцій із цими масивами [26].

Вона тягне на собі аналіз даних, машинне навчання та наукові обчислення, а також суттєво полегшує обробку векторів та матриць. Деякі провідні пакети Python використовують NumPy як основний елемент своєї інфраструктури. До них відносяться scikit-learn, SciPy, pandas та tensorflow.

NumPy має широкий спектр використання:

– наукові обчислення. NumPy користуються вчені для вирішення багатовимірних завдань з математики та фізики, біоінформатики, обчислювальної хімії і навіть когнітивної психології;

– аналіз даних. В основі екосистеми для аналізу даних лежить NumPy. Бібліотека використовується на всіх етапах роботи з даними: вилучення та перетворення, аналіз, моделювання та оцінка, репрезентація [26];

– машинне навчання. Бібліотеки для машинного навчання scikit-learn та SciPy також працюють завдяки обчислювальним потужностям NumPy.

3.1.2 Pandas

Pandas – це програмна бібліотека, написана мовою Python, для обробки та аналізу даних. Робота Pandas з даними будується поверх бібліотеки NumPy,

що є інструментом нижчого рівня. Назва цієї бібліотеки походить від «panel data», що в перекладі означає панельні дані. Панельними даними називають інформацію, яка була отримана у ході досліджень та структурована у вигляді таблиць. Для того щоб аналізувати дані за допомогою Pandas, необхідно розуміти, як влаштовані структури даних усередині бібліотеки. Для роботи з даними Pandas використовує DataFrame та Series. Series – це одновимірний масив. Візуально він схожий на пронумерований список, де ліворуч у колонці знаходяться індекси елементів, а праворуч – самі елементи. DataFrame – це вже двовимірний масив, схожий на таблицю або лист Excel, до речі, дані з Excel можна читати або записувати за допомогою цієї бібліотеки. Як будь-яка таблиця, DataFrame складається зі стовпців і рядків, причому стовпцями будуть вже відомі об’єкти Series. Завдяки неї можна об’єднувати дані в групи, сортувати за певною ознакою, проводити деякі обчислення [27].

3.1.3 Matlotlib

Matplotlib – це бібліотека, що призначена для візуалізації даних двовимірною та трьохвимірною графікою. Автором бібліотеки є Джон Д. Хантер, який був нейробіологом, коли написав Matplotlib під час досліджень кори головного мозку пацієнтів з епілепсією. Головною метою Matplotlib було поліпшення візуалізації та вивчення загальних закономірностей у графіках, що відображають діяльність мозку. Matplotlib був випущений в 2003 році і з того часу став однією з найбільш широко використовуваних бібліотек для побудови графіків. Бібліотека підтримує дуже багато видів графіків та діаграм [28]:

- діаграми розсіювання;
- стовпчасті діаграми та гістограми;
- кругові діаграми;
- діаграми стебло-листя;

– контурні графіки;
– поля градієнтів;
– спектральні діаграми.

3.1.4 Scikit-learn

Scikit-learn – це один з найбільш широко використовуваних пакетів Python для аналізу даних та машинного навчання. Спочатку цей пакет був розроблений під назвою scikits.learn французьким вченим у галузі даних Девідом Курнапо. Він дозволяє виконувати безліч операцій та надає дуже багато різних алгоритмів. Scikit-learn має гарну документацію про методи та функції, а також опис використовуваних алгоритмів. Scikit-learn заснований на NumPy та SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати scikit-learn. Ця бібліотека безкоштовна навіть для комерційного використання.

До складу scikit-learn входять різні алгоритми, у тому числі призначені для завдань класифікації, регресійного та кластерного аналізу даних, включаючи метод опорних векторів, метод випадкового лісу, алгоритм посилення градієнта, метод *k*-середніх та інших [29].

3.1.5 PyQT

PyQt – це бібліотека Python для створення програм з графічним інтерфейсом за допомогою інструментарію Qt. Бібліотека працює на всіх платформах, що підтримуються Qt: Linux та інші UNIX-подібні ОС, macOS та Windows. PyQt також включає Qt Designer – дизайнер графічного інтерфейсу користувача. Програма pyuic генерує Python код із файлів, створених у Qt Designer. Це робить PyQt дуже корисним інструментом швидкого

прототипування. Qt Designer є кросплатформним компонувальником макетів та форм графічного інтерфейсу користувача. Він дозволяє швидко спроєктувати віджети та діалоги, використовуючи екранні форми з використанням тих самих віджетів, які будуть використовуватися в застосунку [30].

3.2 Вхідні дані

У рамках кваліфікаційної роботи був розроблений застосунок для кластеризації часових рядів у вигляді реальних медичних даних, а саме пульс, центральний венозний тиск та внутрішньочерепний тиск. Під час післяопераційного періоду у людини беруться ці та інші показники з деяким інтервалом часу. В рамках роботи розглядаються лише ці три показники, які були виміряні після операції, через день, три, сім, чотирнадцять та на двадцять восьму добу. Після операції пацієнти зазвичай перебувають під медичним наглядом, щоб забезпечити оптимальний післяопераційний період і уникнути можливих ускладнень. Якщо показники приходять в норму, то людину виписують з лікарні, інакше за нею продовжують стежити та доглядати.

Пульс – це ритмічні коливання стінки артерії, які виникають за рахунок зміни заповнення кров’ю артерій при серцевих скороченнях. Це найважливіший показник ритмічних коливань судин, викликаних скороченням серця. Він зазвичай відповідає частоті серцевих скорочень. Пульс може змінюватися в залежності від типу та складності операції, відповідної анестезії, віку та загального стану здоров’я пацієнта.

Зазвичай після операції пульс збільшується на 10-20 ударів в хвилину в порівнянні з показниками до операції. Це пов’язано зі стресом та болем, які супроводжують операцію. Якщо пульс після операції перевищує 100-110 ударів в хвилину, це може свідчати про можливі ускладнення, такі як кровотеча або інфекція.

У деяких випадках післяопераційний пульс може бути нижче звичайного, що може бути пов’язано з анестезією, використовуваною під час операції. Низький пульс може також бути ознакою проблем з серцем, тому лікарі зазвичай звертають особливу увагу на це після операції.

Післяопераційний пульс повинен бути відстежений та звірений з нормальними значеннями пульсу, щоб забезпечити правильний післяопераційний період та уникнути можливих ускладнень. Нормальним пульсом у чоловіків прийнято вважати 60-70 ударів на хвилину, тоді як для жінок це 70-80 ударів на хвилину.

Після операції у людини постійно заміряють центральний венозний тиск (ЦВТ), оскільки це допомагає контролювати стан серцево-судинної системи та оцінювати ефективність відновлення. ЦВТ вказує на тиск в великій вені, яка прямує до серця, і допомагає визначити обсяг крові, що повертається до серця. Це важливо, оскільки при операції людина може втратити кров, що може призвести до зниження кровообігу. Замір ЦВТ може також допомогти виявити підвищений тиск вен, що може бути показником серйозних ускладнень, таких як тромбоемболія легенів або застійна серцева недостатність. Крім того, замір ЦВТ може допомогти лікареві визначити оптимальну дозу рідини, яку необхідно вводити пацієнту, щоб уникнути надмірного навантаження на серце. Оптимальний рівень ЦВТ залежить від стану пацієнта, його віку, наявності супутніх захворювань та стадії лікування. Зазвичай, оптимальний рівень ЦВТ знаходиться в діапазоні від 60 до 110 мм водного. ст. у дорослих. Отже, замір ЦВТ після операції є важливим інструментом для контролю стану серцево-судинної системи та попередження можливих ускладнень.

Відстеження внутрішньочерепного тиску (ВЧТ) є важливим елементом післяопераційного спостереження у пацієнтів. Нейрохірургічні операції можуть призвести до збільшення ВЧТ, що може викликати різноманітні ускладнення. Збільшення ВЧТ може призвести до компресії мозку, що може привести до порушення функцій мозку, зниження свідомості та навіть до смерті. Моніторинг ВЧТ дозволяє лікарям вчасно виявити збільшення тиску і

вжити відповідних заходів для його зменшення, таких як ліки або дренування. Також моніторинг ВЧТ може допомогти у визначенні ефективності лікування та процесу підновлення пацієнта. Підвищений ВЧТ може викликати головний біль, тошноту, рвоту, судоми, дезорієнтацію та інші неврологічні симптоми. Крім того, зниження ВЧТ може вказувати на кровотечу, що може бути небезпечним і вимагати негайного лікування. Моніторинг ВЧТ після операції може допомогти у визначені ризику розвитку ускладнень та прийнятті відповідних заходів для їх запобігання. У деяких випадках, низький ВЧТ може бути показником гіповолемії (втрата крові або інших рідин) і вимагати додаткового лікування. Високий ВЧТ може бути пов’язаний з ризиком розвитку інших ускладнень, таких як інфекція та набряк мозку.

3.3 Програмна реалізація

3.3.1 Розробка інтерфейсу

Інтерфейс – це візуальна частина, через яку користувач взаємодіє із програмною частиною, тому інтерфейс повинен буди інформативним та легким у розумінні. Розробити інтерфейс на Python можна завдяки фреймворку PyQt, який представляє віджети для інтерфейсу. PyQt є потужною бібліотекою, яка надає можливість розробки графічних інтерфейсів для програм на мові Python. Вона базується на бібліотеці Qt, що дозволяє створювати крос-платформові програми, які працюють на різних операційних системах. Також PyQt представляє інтерфейс графічного дизайнера Qt Designer (рис. 3.1), завдяки якому можна легко створювати віджети для застосунку. На рисунку 3.1 по центру можна побачити вікно програми, що розробляється. Це вікно можна редагувати, та доповнювати різними віджетами, добавляючи функціонал. Також завдяки Qt Designer можна задавати різні стилі віджетам, наприклад кнопці (рис. 3.2).

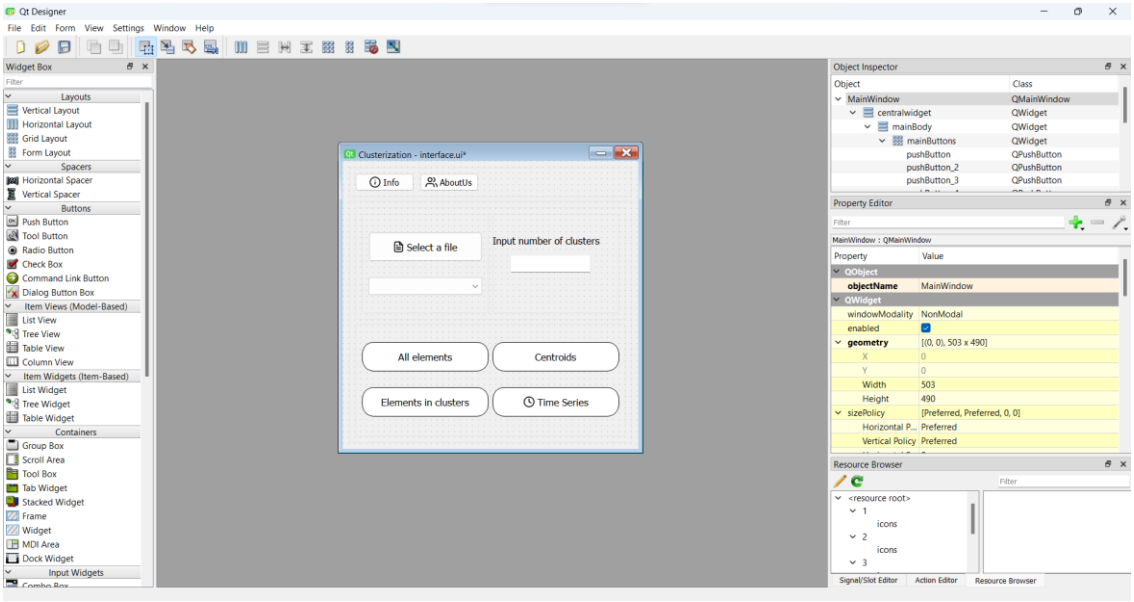

Рисунок 3.1 – Інтерфейс Qt Designer

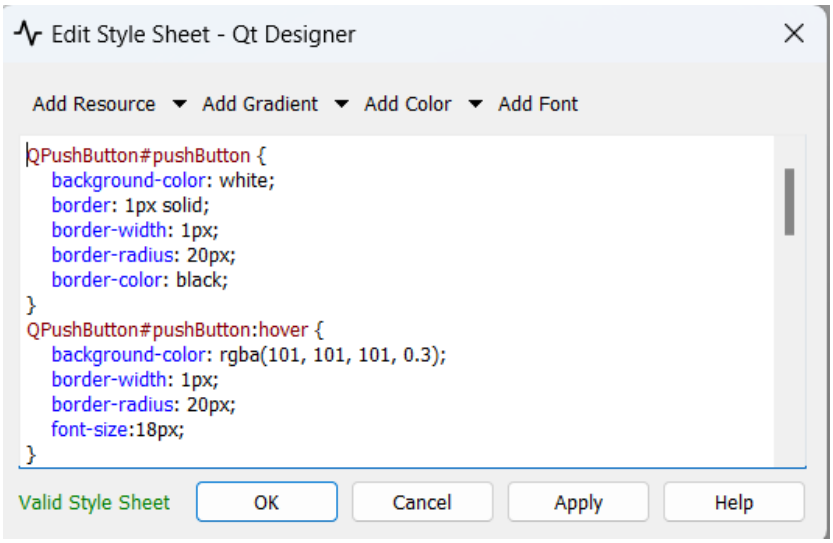

Рисунок 3.2 – Надання стилю для кнопки

Стиль можна задавати завдяки мові опису зовнішнього вигляду документу css, який також використовуються для створення інтернет сторінок. Створивши кнопку для застосунку в неї буде стандартний стиль (рис. 3.3). Щоб змінити стиль, Qt Designer надає вікно для редагування стилю (рис. 3.4), завдяки якому можна змінити положення на екрані, форми об’єкта, стиль тексту, надпис на кнопці та інші параметри. В результаті можна отримати кнопку, зображену на рисунку 3.5.

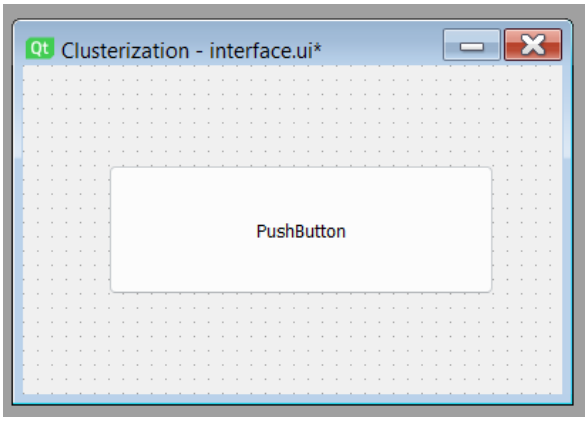

Рисунок 3.3 – Стандартний стиль для кнопки

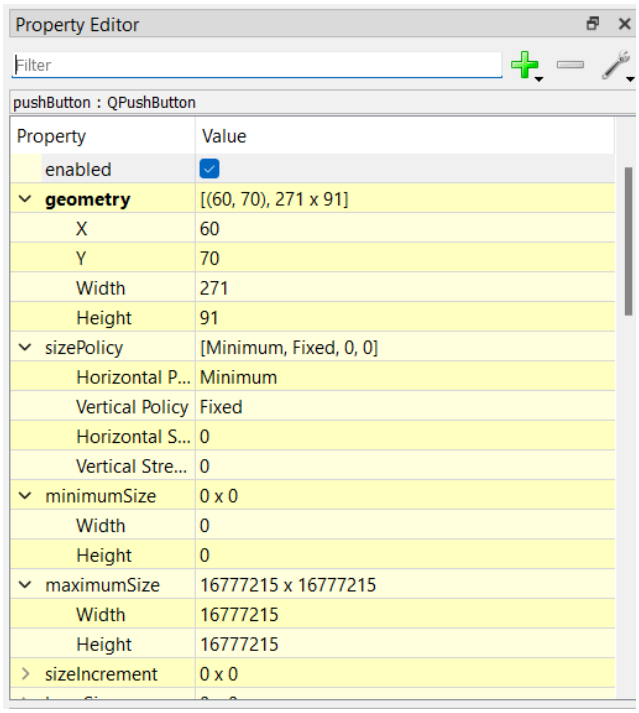

Рисунок 3.4 – Вікно для редагування стилю

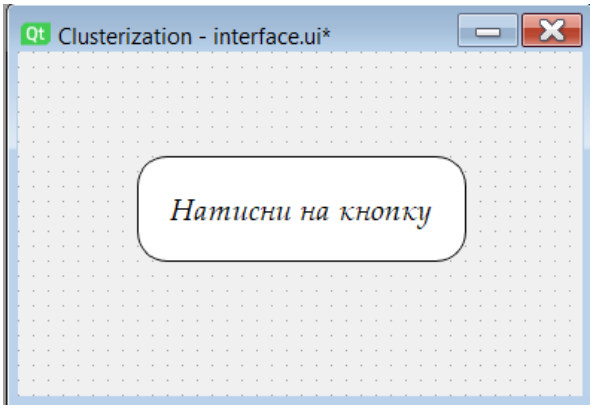

Рисунок 3.5 – Змінений стиль для кнопки

Після того, як графічний інтерфейс розроблений (рис. 3.6), його треба перенести на Python. Це можна зробити завдяки інструменту pyuic, який генерує Python код із файлів з розширеннями ui, створені завдяки Qt Designer. Для цього треба виповнити команду в терміналі, зображену на рисунку 3.7, де interface.ui – це назва файлу, створеного за допомогою Qt Designer, а generatedFile.py – це файл куди треба зберегти код.

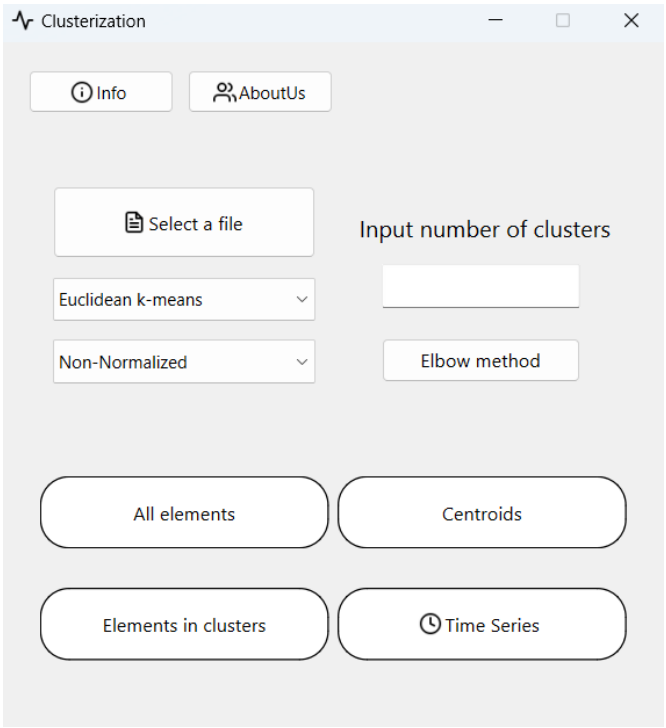

Рисунок 3.6 – Інтерфейс застосунку

```
C:\arturk>pyuic5 -x interface.ui -o generatedFile.py
```

Рисунок 3.7 – Генерація Python коду для інтерфейсу

На рисунку 3.8 зображено приклад згенерованого коду для кнопки. Ця кнопка буде відображатися на екрані, як зображено на рисунку 3.5, але якщо на неї натиснути, то нічого не буде відбуватиметься. Для того щоб при натисканні на кнопку відбувалась якась дія, треба додати їй функцію, завдяки

команді connect, зображеній на рисунку 3.9, де TimeSeries – це назва функції, яка буде викликатись при натисканні на кнопку.

```
self.btnTimeSeries = QtWidgets.QPushButton(self)
self.btnTimeSeries.setText("Time series")
self.btnTimeSeries.setFont(font)
self.btnTimeSeries.setGeometry(QtCore.QRect(215, 270, 140, 45))
self.btnTimeSeries.show()
```

Рисунок 3.8 – Згенерований код для кнопки

```
self.btnTimeSeries.clicked.connect(self.TimeSeries)
```

Рисунок 3.9 – Додавання функції для кнопки

3.3.2 Зчитування даних

Застосунок працює з даними, тому наступним кроком треба отримати ці дані. Кожні окремі набори даних для пульсу, центрального венозного тиску та внутрішньочерепного тиску зберігаються в окремих excel файлах (рис. 3.10). Кожен файл має сім стовпців, перший для ідентифікатора людини, інші шість для даних, які брались з деяким інтервалом часу. У ролі ідентифікатора може бути як прізвище з ім’ям, так і порядковий номер або щось інше.

	A	B	C	D	E	F	G
1	Man	Begin	1day	3day	7day	14day	28day
2	1	70	65	65	65	25	25
3	2	75	75	75	75	35	35
4	3	80	65	65	65	20	20
5	4	80	80	80	35	35	35
6	5	80	60	60	60	25	25
7	6	85	85	85	45	45	45
8	7	85	85	85	50	35	35
9	8	110	60	60	35	35	35
10	9	95	65	65	30	30	30
11	10	85	60	60	60	35	35
12	11	95	95	95	35	35	35

Рисунок 3.10 – Приклад вхідних даних

Для того, щоб загрузити дані застосунок має спеціальну кнопку (рис. 3.11), яка дає можливість вибрати файл (рис. 3.12), який треба прочитати.

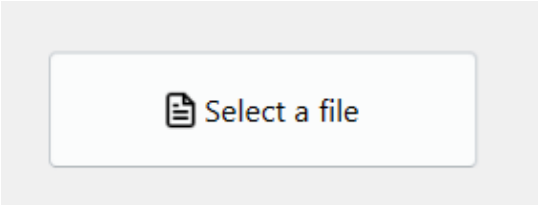

Рисунок 3.11 – Кнопка, щоб вибрати файл, який треба прочитати

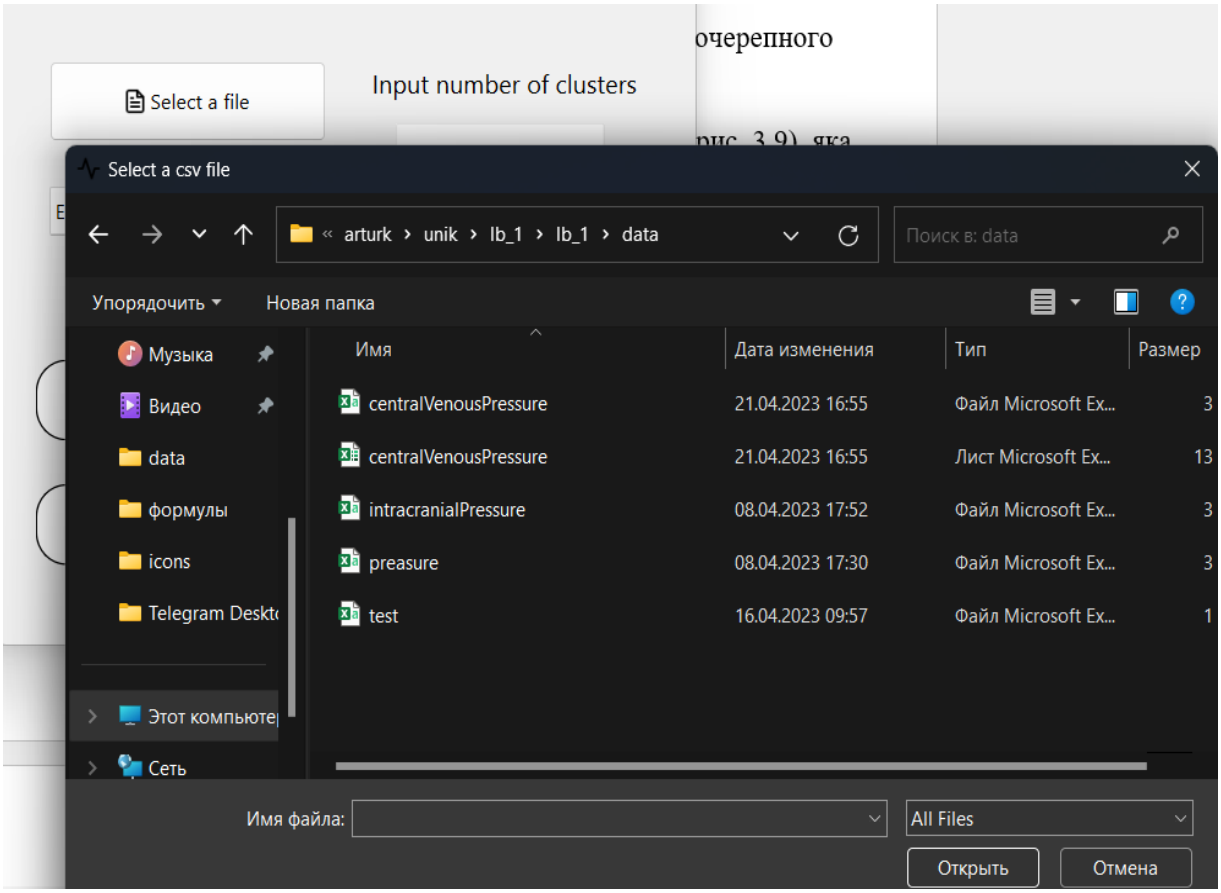

Рисунок 3.12 – Вікно для вибору файлу

Це можна зробити завдяки бібліотеці pandas, яка має спеціальну функцію read_excel(), для того щоб конвертувати excel файли у DataFrame, який складається із стовпців та рядків, так звану двовимірну матрицю, щоб працювати із цими даними.

3.3.3 Відображення усіх елементів з файлу

Застосунок має можливість виводити усі елементи, які були прочитані з файлу, завдяки спеціальній кнопки (рис. 3.13).

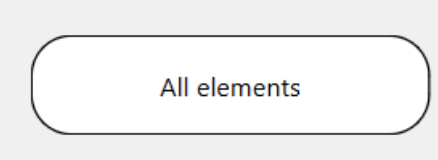

Рисунок 3.13 – Кнопка для виводу усіх елементів

Для цієї кнопки реалізований спеціальний метод, який отримує дані, які були зчитані на попередньому кроці, та будує графік для кожного рядка з файлу завдяки бібліотеці matplotlib. На осі *X* відображаються назви колонок з файлу, а на осі *Y* – значення показників (рис. 3.14).

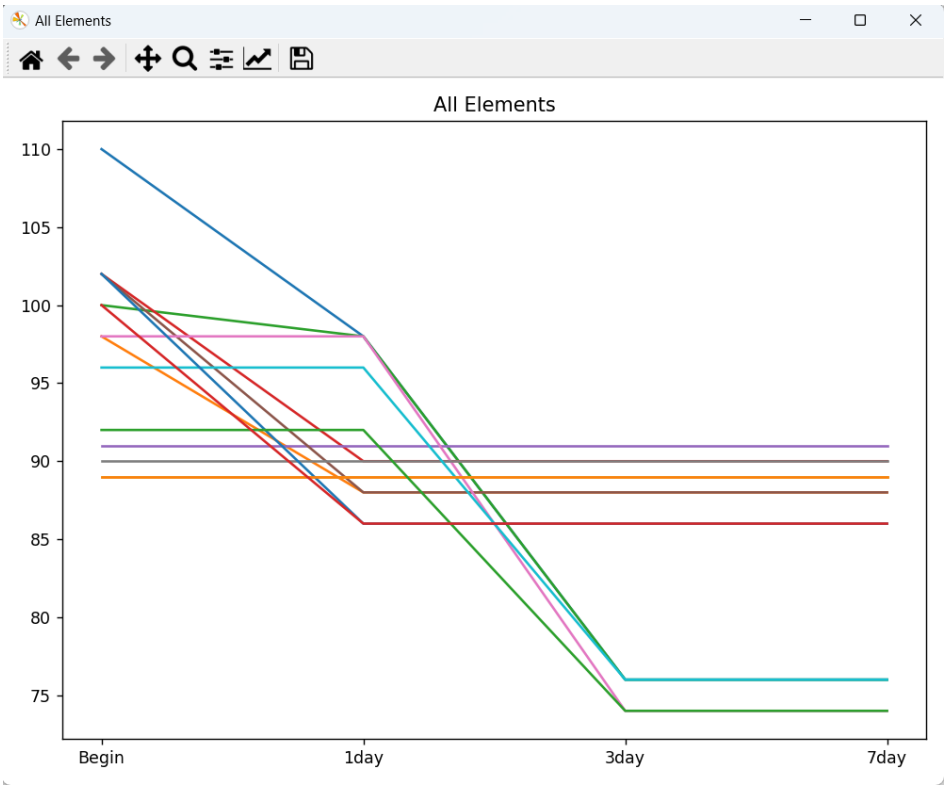

Рисунок 3.14 – Відображення усіх елементів з файлу на графіку

3.3.4 Кластеризація часового ряду

Після того, як дані були зчитані, їх можна відкластеризувати. Для цього треба задати кількість кластерів (рис. 3.15) та вибрати метод кластеризації (рис. 3.16).

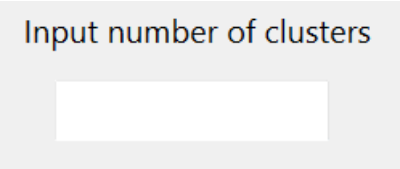

Рисунок 3.15 – Поле для вводу кількості кластерів

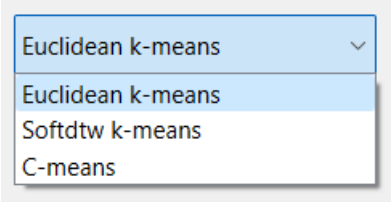

Рисунок 3.16 – Поле для вибору кластеризації

QLineEdit (рис. 3.15) віджет бібліотеки PyQt дає можливість вводити текст у вікно, після чого в програмній реалізації можна дістати цей текст завдяки методу text(). На рисунку 3.17 можна побачити приклад як працювати із цим віджетом, де NumberOfClusterBox – це вікно для вводу текста, а NumberOfClusters – це змінна, яка зберігає значення, введене в цьому вікну.

```
try:
    if self.NumberOfClustersBox.text().isdigit():
        self.NumberOfClusters = int(self.NumberOfClustersBox.text())
    else:
        raise Exception("NumberOfClusters contains not digital value")
except Exception as e:
```

Рисунок 3.17 – Отримання значення з QLineEdit віджета

QComboBox (рис. 3.16) віджет бібліотеки PyQt дає можливість зробити список із можливими варіантами вибору. Для того, щоб добавити в цей список

елементи, треба використати метод addItems(). Завдяки методу currentText() у програмі можна взяти варіант, який вибрав користувач (рис. 3.18).

```
self.algorithm = QtWidgets.QComboBox(parent=self.widget_3)
self.algorithm.addItems(["Euclidean k-means", "Softdtw k-means", "C-means"])
```

Рисунок 3.18 – Добавлення елементів у випадаюче вікно

Після того, як користувач ввів кількість кластерів та обрав метод кластеризації, дані можна кластеризувати. Кластеризації відбувається завдяки методам бібліотеки scikit-learn, яка має підпакет clustering. Її результатом є одновимірний масив, довжина якого дорівнює кількості часових рядів, де кожен осередок зберігає інформацію до якого кластера відноситься відповідний часовий ряд (рис. 3.19).

	0	1	2	3	4	5	6	7
0	1	1	1	1	0	2	2	0

Рисунок 3.19 – Приклад відкластеризованих даних

Після того, як часові ряди були віднесені до певних кластерів, формуються графіки (рис. 3.20), зображені на рисунках 3.21 та 3.22. На одному графіку відображаються усі часові ряди, які відносяться до певного кластеру, а також сама центроїда красним кольором. Також після кластеризації створюється новий файл з префіксом out, куди додається нова колонка cluster з інформацією про те, до якого кластера відноситься відповідний часовий ряд. Для того щоб зберегти excel файл використовується to_excel() метод із пакету pandas (рис. 3.23).

```
for xx in X_data[y_pred == yi]:
    plt.plot(default_y_ticks, xx.ravel(), "k-", alpha=.2)
    plt.xticks(default_y_ticks, y)
plt.plot(km.cluster_centers_[yi].ravel())
```

Рисунок 3.20 – Цикл для побудови графіка для часових рядів

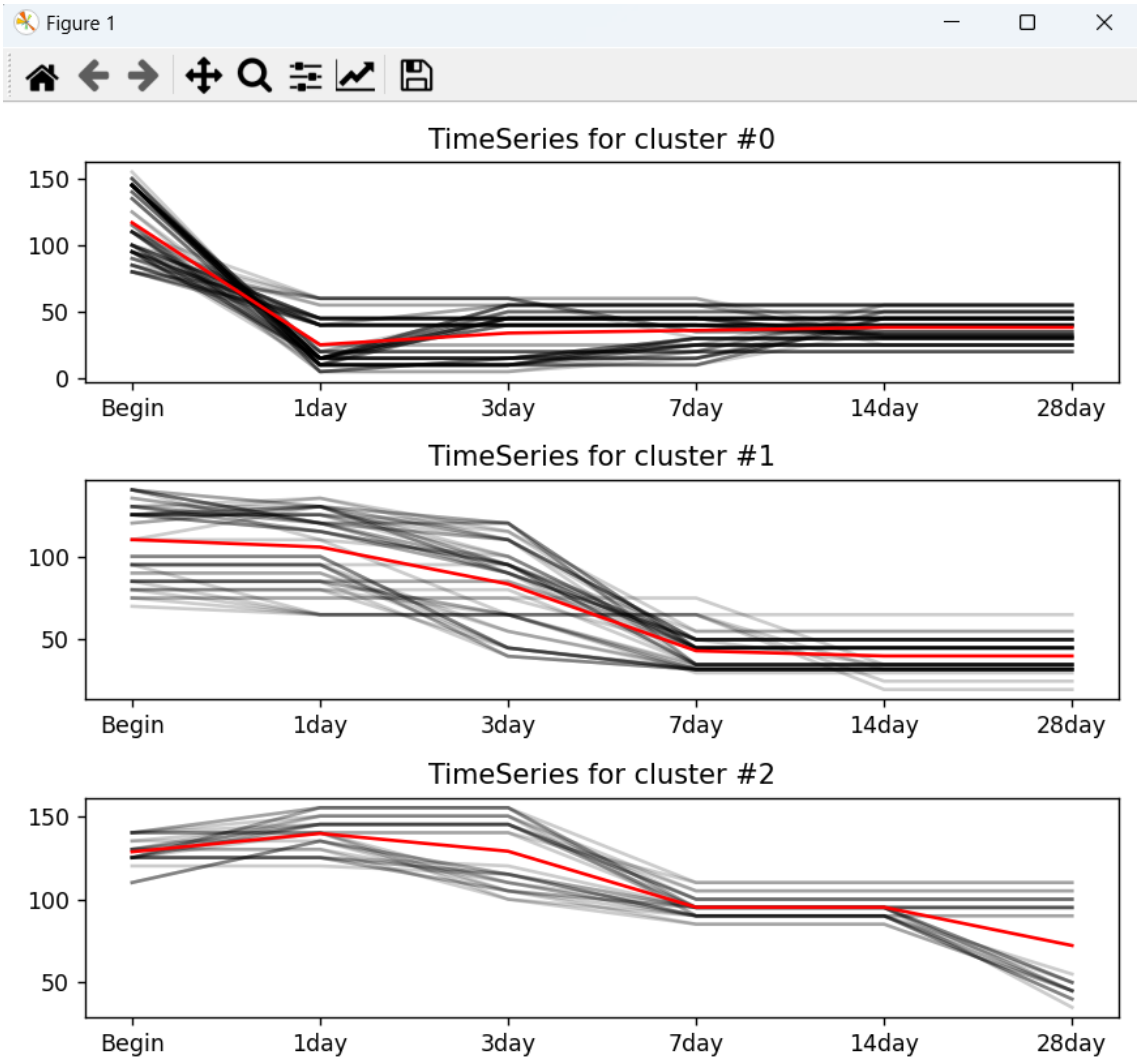

Рисунок 3.21 – Відкластеризовані часові ряди на три кластери

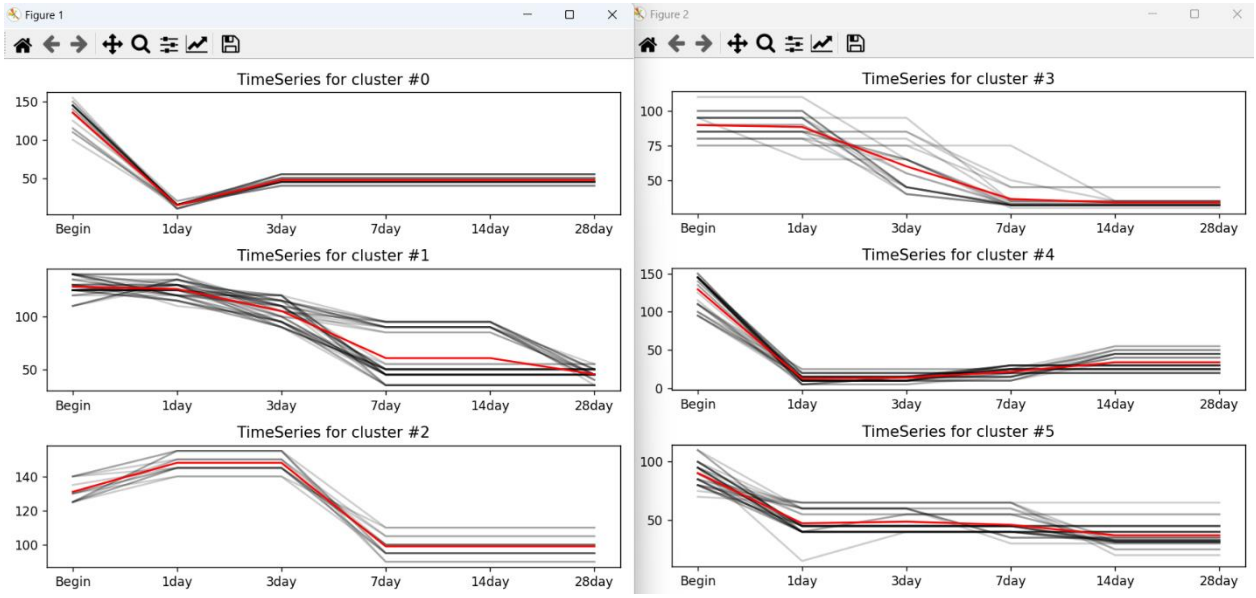

Рисунок 3.22 – Відкластеризовані часові ряди на шість кластерів

	A	B	C	D	E	F	G	H	I
1		Man	Begin	1day	3day	7day	14day	28day	Cluster
2	0	1	70	65	65	65	25	25	5
3	1	2	75	75	75	75	35	35	3
4	2	3	80	65	65	65	20	20	5
5	3	4	80	80	80	35	35	35	3
6	4	5	80	60	60	60	25	25	5
7	5	6	85	85	85	45	45	45	3
8	6	7	85	85	85	50	35	35	3
9	7	8	110	60	60	35	35	35	5
10	8	9	95	65	65	30	30	30	3

Рисунок 3.23 – Створений новий excel файл після кластеризації

3.3.5 Метод «ліктя»

Для того щоб зазначити оптимальну кількість кластерів при кластеризації застосунок надає можливість скористатися методом «ліктя» (рис. 3.24).

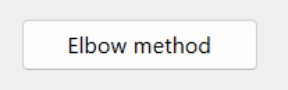

Рисунок 3.24 – Кнопка, для використання методу «ліктя»

Застосунок кластеризує дані у циклі, використовуючи кількість кластерів від нуля до двадцяти та на кожному кроці виводить на графік критерій кластеризації (рис. 3.25).

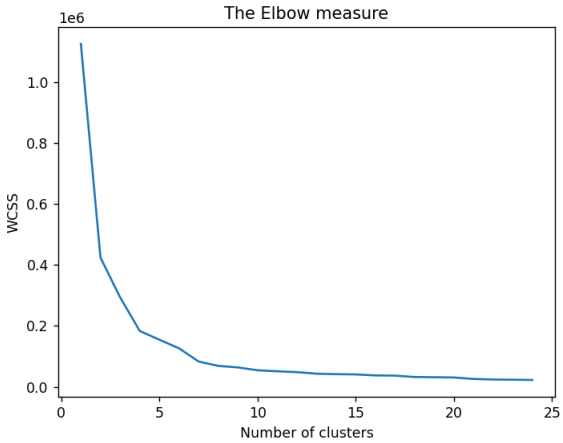

Рисунок 3.25 – Результат методу «ліктя»

3.3.6 Нормалізація даних

Також застосунок дає можливість нормалізувати дані перед кластеризацією, за допомогою метода при якому дані перетворюються таким чином, щоб їхнє середнє значення дорівнювало 0, а стандартне відхилення дорівнювало 1. Це доступно, завдяки спеціальному вікну (рис. 3.26). Нормалізація даних потрібна для покращення кластеризації. Після кластеризації застосунок виводить графік, побудований для нормалізованих часових рядів (рис. 3.27), а також для звичайних (рис. 3.28), але застосунок відносить їх до певних кластерів не за допомогою кластеризації, а за допомогою вже нормалізованих аналогів, які вже були відкластеризовані. Наприклад, якщо якийсь нормалізований часовий ряд належить до першого кластеру, тоді цей же часовий ряд, але до нормалізації також належить до першого кластеру. Це потрібно для перевірки результатів.

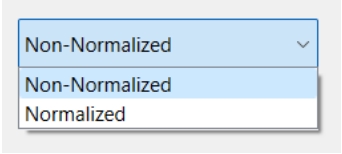

Рисунок 3.26 – Вікно для вибору нормалізації даних

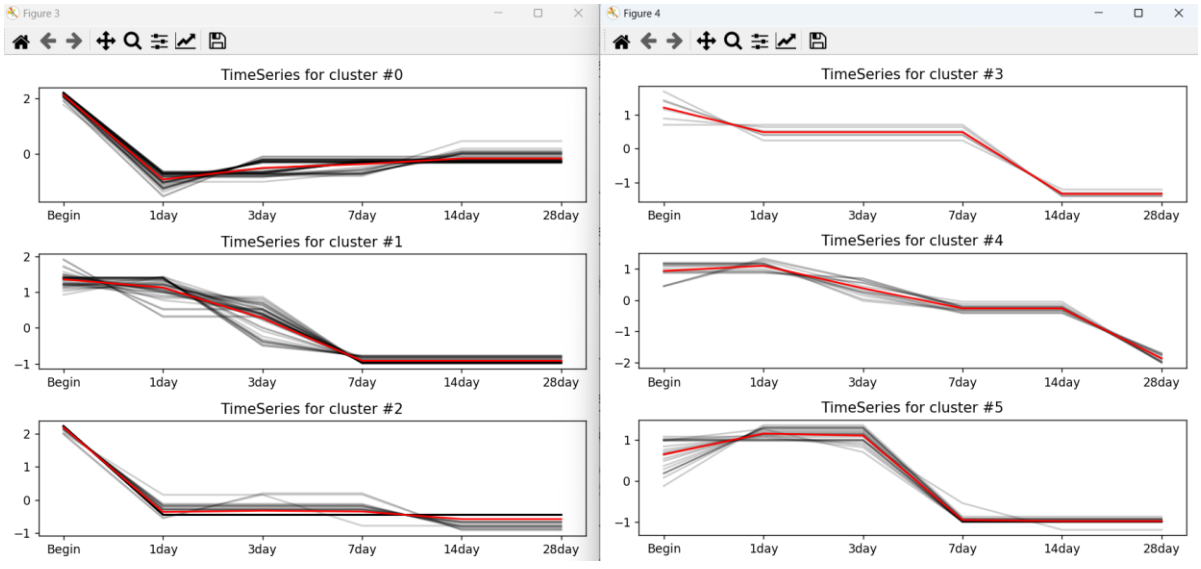

Рисунок 3.27 – Відкластеризовані часові ряди на 6 кластерів після нормалізації

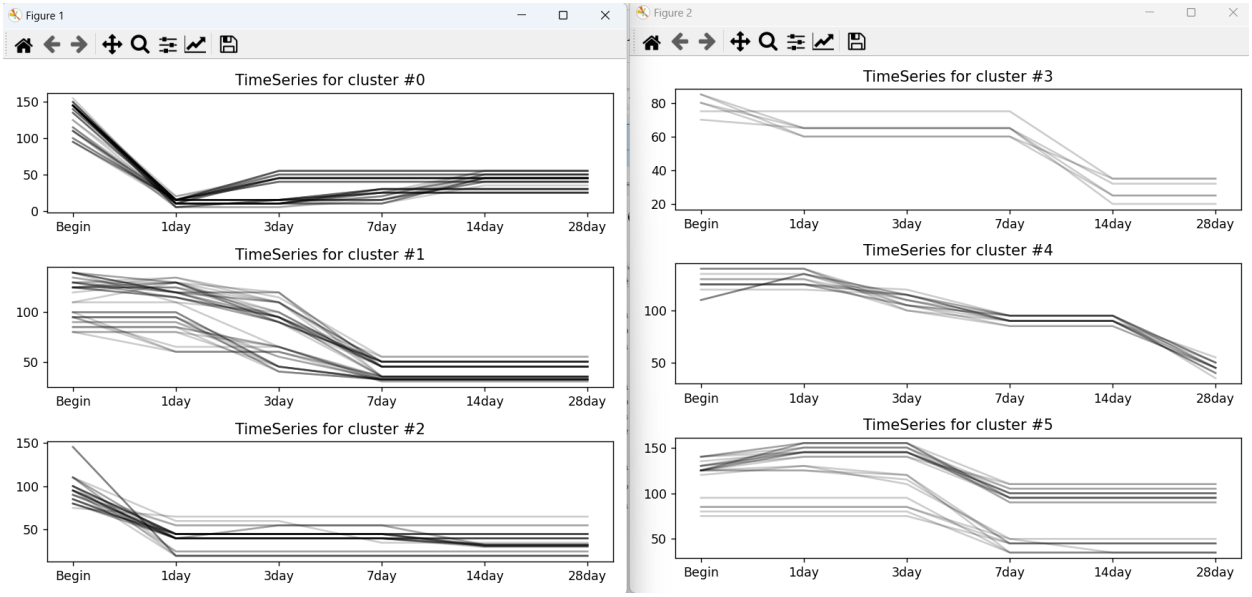

Рисунок 3.28 – Відкластеризовані часові ряди на 6 кластерів після нормалізації (оригінальний вид часових рядів)

3.3.7 Відображення центроїдів

Розроблений застосунок має додатковий функціонал завдяки спеціальній для перегляду центроїдів для кожного кластеру. На рисунку 3.29 зображені центроїди кластерів після кластеризації завдяки euclidian *k*-means методу, а на рисунках 3.30 та 3.31 – завдяки методам dtw *k*-means та fuzzy *c*-means відповідно.

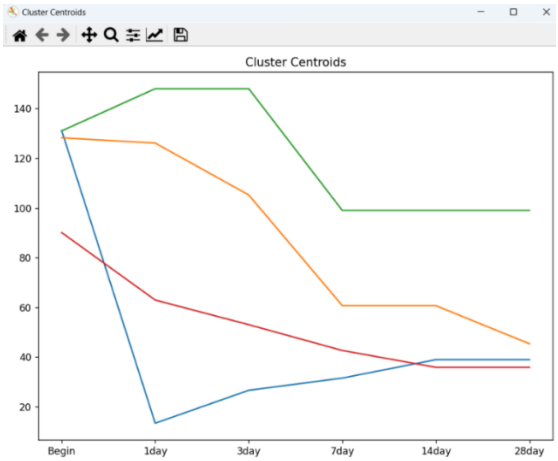

Рисунок 3.29 – Центроїди для чотирьох кластерів, користуючись euclidian *k*-means методом

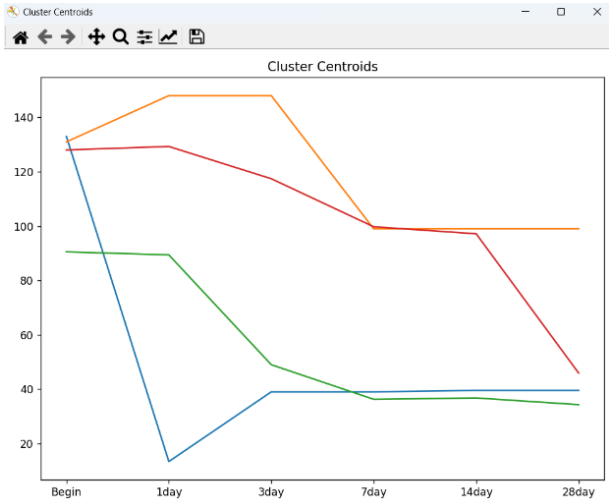

Рисунок 3.30 – Центроїди для чотирьох кластерів, користуючись softdtw *k*-means методом

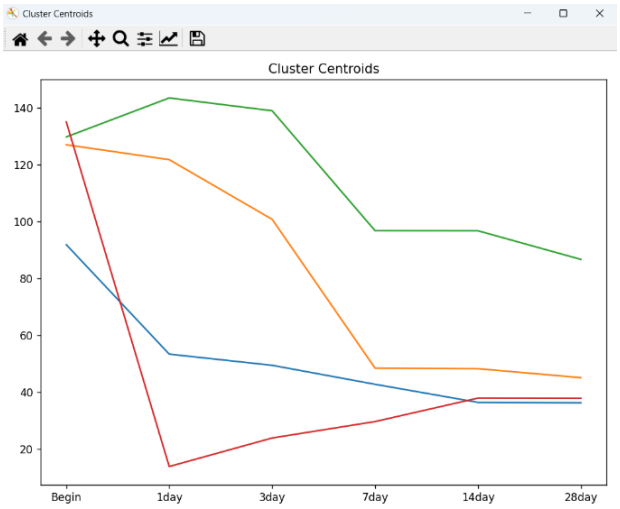

Рисунок 3.31 – Центроїди для чотирьох кластерів, користуючись fuzzy *c*-means методом

3.3.8 Елементи в кластерах

Також в застосунку є можливість виводити перші чотири елементи з кожного кластеру, для цього є спеціальна кнопка (рис. 3.32). Після кластеризації з кожного кластеру беруться 4 елементи, які відображаються на окремих графіках (рис. 3.33-3.35).

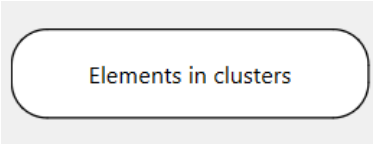

Рисунок 3.32 – Кнопка для виводу перших чотирьох елементів з кластера

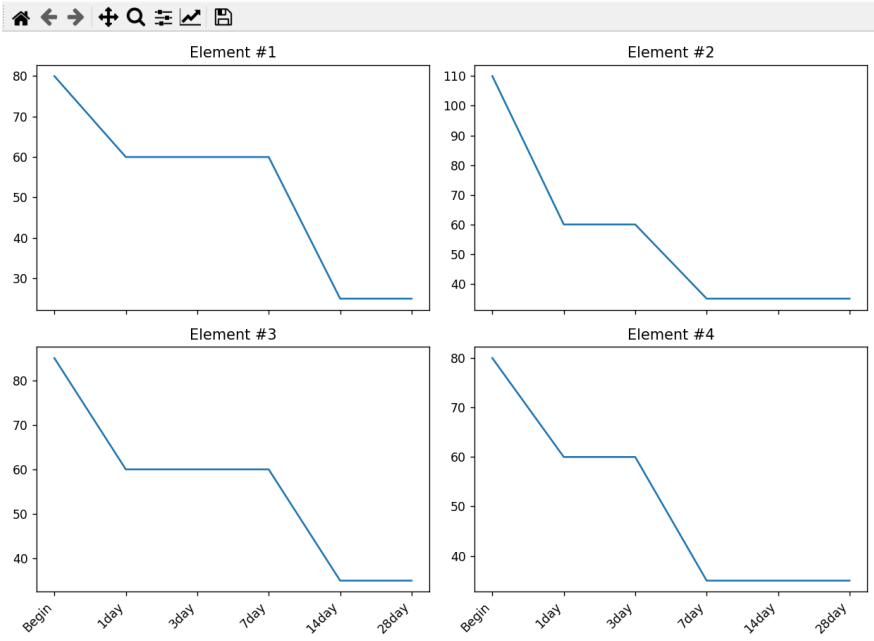

Рисунок 3.33 – Елементи з першого кластера

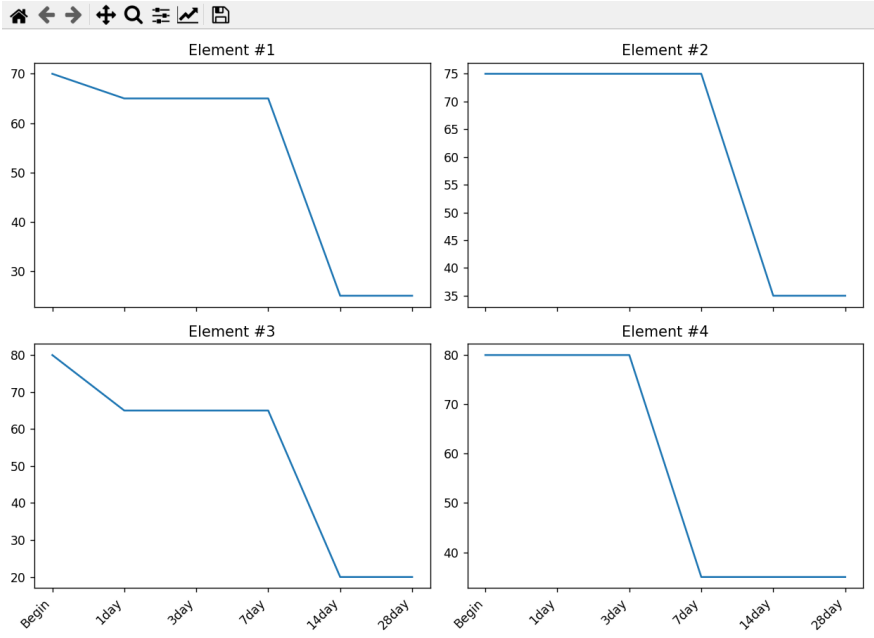

Рисунок 3.34 – Елементи з другого кластера

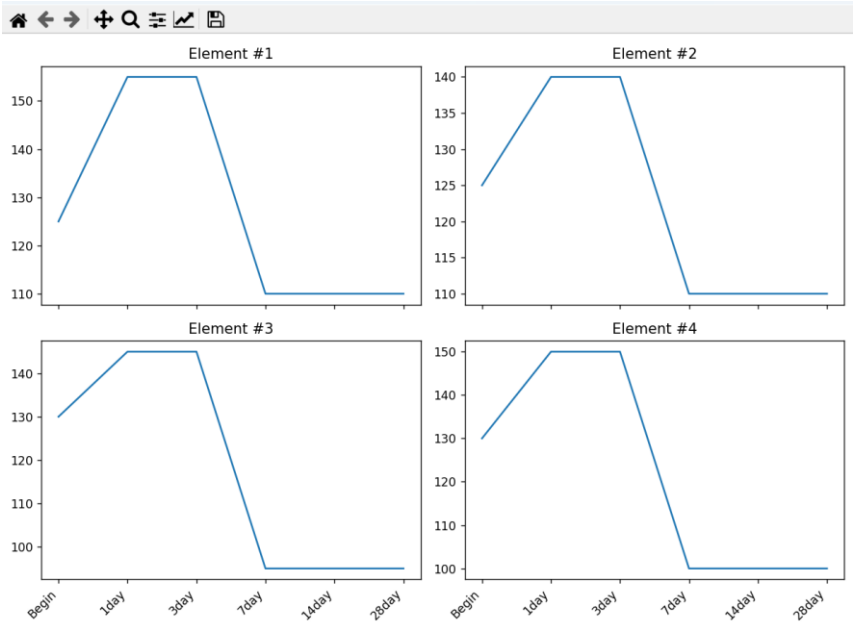

Рисунок 3.35 – Елементи з третього кластера

3.4 Тестування розробленої моделі

Після розробки застосунку проводиться тестування, щодо його коректної роботи. Під час тестування було виявлено декілька помилок. Перша пов’язана з полем для вводу кількості кластерів. Якщо це поле не заповнити, або вести не числове значення, програма вимикалась з помилкою. Для того, щоб вирішити цю проблему, була додана перевірка цього поля на введення користувачем лише числових значень (рис. 3.36). В випадках, коли користувач ввів некоректне значення, програма виводить повідомлення про те, щоб користувач ввів правильне значення (рис. 3.37).

```
try:
    if self.NumberOfClustersBox.text().isdigit():
        self.NumberOfClusters = int(self.NumberOfClustersBox.text())
    else:
        raise Exception("NumberOfClusters contains not digital value")
```

Рисунок 3.36 – Перевірка, що поле складається
лише з цифрових значень

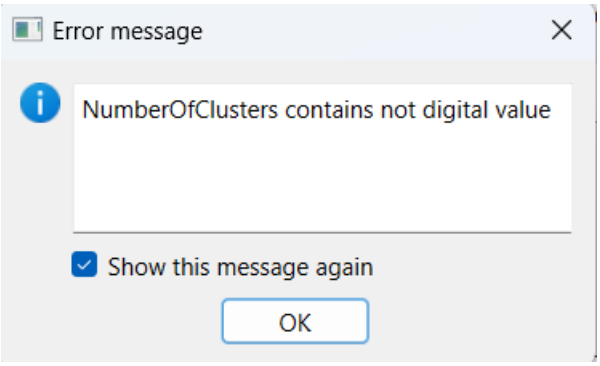

Рисунок 3.37 – Повідомлення, що значення кількості кластерів некоректне

Друга проблема з'явилась, коли було вирішено спробувавати відкластеризувати дані, не вибравши файл із даними. Після чого застосунок закінчився з помилкою. Для вирішення цього питання, треба перед початком усієї операції перевірити чи вибрав користувач excel файл (рис. 3.38). Якщо він не зробив цього, то отримує повідомлення (рис. 3.39).

```
try:
    data_csv = pd.read_excel(self.filePath)
except Exception as e:
    self.statusLabel.showMessage("Please select a right csv file")
    return
```

Рисунок 3.38 – Перевірка, що користувач вибрав excel файл

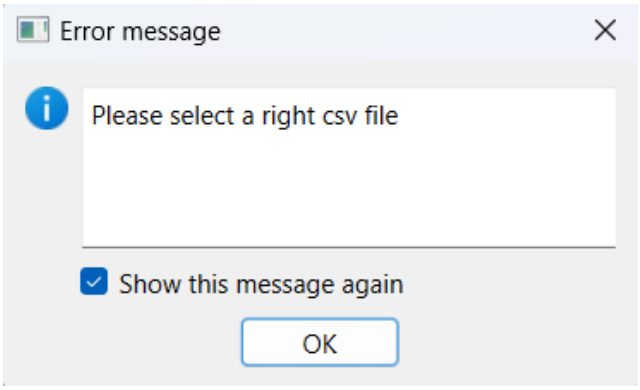

Рисунок 3.39 – Повідомлення, що користувач не вибрав файл

Також якщо під час кластеризації щось пішло не так, наприклад користувач загрузив excel файл, але з неправильними даними, то замість того щоб програма просто вимкнулась, користувач отримує повідомлення та зможе продовжити користуватися застосунком, щоб замінити файл для кластеризації (рис. 3.40).

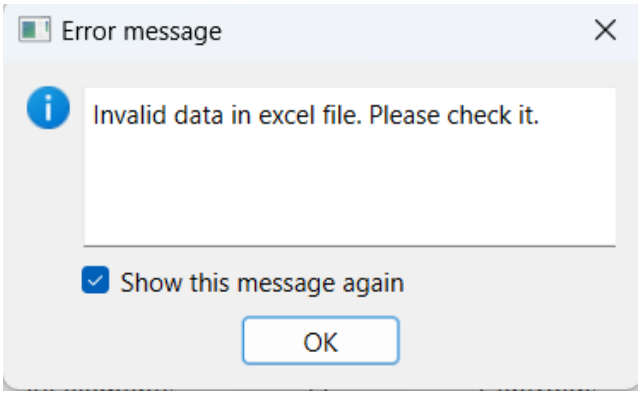

Рисунок 3.40 – Повідомлення про помилку під час кластеризації

ВИСНОВКИ

У рамках кваліфікаційної роботи було досліджено застосування алгоритмів кластеризації для аналізу часових рядів у вигляді медичних даних, а також розробка застосунку для кластеризації цих даних.

У процесі роботи було проаналізовано особливості медичних часових рядів, були запропоновані та застосовані методи попередньої обробки даних, що включають нормалізацію для підвищення якості аналізу.

Далі було проведено кластеризацію часових рядів з використанням алгоритмів *k*-середні, *c*-середні та DTW алгоритм. Застосування цих методів було реалізоване з використанням середовища PyCharm, і всі алгоритми були реалізовані на мові програмування Python. Під час проведення експериментальних досліджень було встановлено, що жоден з алгоритмів не має виразних переваг у всіх параметрах порівняно з іншими алгоритмами. Кожен з них має своє переваги та недоліки.

Крім того, проведений аналіз кластерів дозволив виявити подібність та відмінності між медичними часовими рядами, що може бути корисним для класифікації та прогнозування захворювань, а також прийняття медичних рішень.

Дана кваліфікаційна робота внесла значний внесок у область кластеризації медичних часових рядів. Розроблена методика являє собою ефективний та перспективний інструмент для аналізу медичних даних, здатний допомогти лікарям та дослідникам у виявленні патернів, класифікації захворювань та прийнятті рішень у галузі охорони здоров’я [31-34].

Результати роботи апробовано у вигляді тез доповідей у збірнику XVIII Міжнародної науково-практичної конференції «Developing an understanding of the specificities and functions of scientific language» [35].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Женчак, А. (2020). Прогнозування та аналіз часових рядів.

2. Lyashenko, V., Kobylin, O., & Selevko, O. (2020). Wavelet analysis and contrast modification in the study of cell structures images.

3. Бідюк, П. І., Романенко, В. Д., & Тимощук, О. Л. (2010). Аналіз часових рядів.

4. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

5. Bodyanskiy, Y., Vynokurova, O., Szymański, Z., Kobylin, I., & Kobylin, O. (2016, August). Adaptive robust models for identification of nonstationary systems in data stream mining tasks. In *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)* (pp. 263-268). IEEE.

6. Kobylin, O. A., Vyskrebentseva, S. O., & Petrova, R. V. (2019). Обробка даних, що містять пропуски в задачах кластеризації. *Системи управління, навігації та зв'язку. Збірник наукових праць*, *5*(57), 45-50.

7. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative based clustering of long multivariate sequences with different lengths. In *2018 IEEE second international conference on Data Stream Mining & Processing (DSMP)* (pp. 545-548). IEEE.

8. Shafronenko, A. Y., Bodyanskiy, Y. V., & Pliss, I. P. (2019, September). The Fast Modification of Evolutionary Bioinspired Cat Swarm Optimization Method. In *2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL)* (pp. 548-552). IEEE.

9. Understanding *K*-means Clustering in Machine Learning. URL: https://www.analyticsvidhya.com/blog/2021/11/understanding-k-means-clustering-in-machine-learningwith-examples/ (дата звернення 26.04.2023).

10. Moshkovitz, M., Dasgupta, S., Rashtchian, C., & Frost, N. (2020, November). Explainable k-means and k-medians clustering. In *International conference on machine learning* (pp. 7055-7065). PMLR.

11. Abbas, S. A., Aslam, A., Rehman, A. U., Abbasi, W. A., Arif, S., & Kazmi, S. Z. H. (2020). K-means and k-medoids: Cluster analysis on birth data collected in city Muzaffarabad, Kashmir. *IEEE Access*, *8*, 151847-151855.

12. Lyashenko, V., Matarneh, R., Kobylin, O., & Putyatin, Y. (2016). Contour detection and allocation for cytological images using Wavelet analysis methodology.

13. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Irina, K. (2021, September). Online Credibilistic Fuzzy Clustering Method Based on Cauchy Density Distribution Function. In *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)* (pp. 704-707). IEEE.

14. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2020). Online robust fuzzy clustering of data with omissions using similarity measure of special type. In *Lecture Notes in Computational Intelligence and Decision Making: Proceedings of the XV International Scientific Conference "Intellectual Systems of Decision Making and Problems of Computational Intelligence"(ISDMCI'2019), Ukraine, May 21–25, 2019 15* (pp. 637-646). Springer International Publishing.

15. Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Shafronenko, A. (2019, July). Fuzzy clustering of incomplete data by means of similarity measures. In *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (pp. 957-960). IEEE.

16. Bodyanskiy, Y., Kobylin, I., Rashkevych, Y., Vynokurova, O., & Peleshko, D. (2018, February). Hybrid fuzzy-clustering algorithm of unevenly and asynchronously spaced time series in computer engineering. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 930-935). IEEE.

17. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations

in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

18. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video clustering via multidimensional time-series analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63).

19. Shafronenko, A., Dolotov, A., Bodyanskiy, Y., & Setlak, G. (2018, August). Fuzzy clustering of distorted observations based on optimal expansion using partial distances. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* (pp. 327-330). IEEE.

20. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Patlan, K. (2019, June). Fuzzy Clusterization of Distorted by Missing Observations Data Sets Using Evolutionary Optimization. In *2019 9th International Conference on Advanced Computer Information Technologies (ACIT)* (pp. 217-220). IEEE.

21. An introduction to Dynamic Time Warping. URL: https://rtavenar.github.io/blog/dtw.html (дата звернення 30.04.2023).

22. ІЩо таке Python та де він використовується. URL: https://dan-it.com.ua/blog/python-chto-jeto-za-jazyk-programmirovanija-i-gde-ego-ispolzujut/ (дата звернення 26.04.2023).

23. Sahoo, K., Samal, A. K., Pramanik, J., & Pani, S. K. (2019). Exploratory data analysis using Python. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(12), 2019.

24. Heydt, M. (2017). Learning pandas. Packt Publishing Ltd.

25. Уроки Python. URL: https://itproger.com/ua/course/python (дата звернення 26.04.2023).

26. Numpy Introduction. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp (дата звернення 26.04.2023).

27. Pandas Tutoria. URL: https://www.w3schools.com/python/pandas/pandas_dataframes.asp (дата звернення 26.04.2023).

28. Introduction to Plotting with Matplotlib in Python. URL: https://www.datacamp.com/tutorial/matplotlib-tutorial-python (дата звернення 26.04.2023).

29. Scikit Learn Tutorial. URL: https://www.tutorialspoint.com/scikit_learn/index.htm (дата звернення 26.04.2023).

30. Python and PyQt: Building a GUI Desktop Calculator. URL: https://www.tutorialspoint.com/pyqt/index.htm (дата звернення 26.04.2023).

31. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

32. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

33. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

34. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.

35. Developing an understanding of the specificities and functions of scientific language. URL: https://intersci.eu/wp-content/uploads/2023/04/Developing-an-understanding-of-the-specificities-and-functions-of-scientific-language.pdf (дата звернення 28.04.2023).