

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)

КЛАСТЕРИЗАЦІЯ ЧАСОВИХ РЯДІВ У ЗАДАЧАХ АНАЛІЗУ
РЕЗУЛЬТАТІВ СПОРТСМЕНІВ ШВИДКІСНОГО
ПІДВОДНОГО ПЛАВАННЯ
(тема)

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

Маханьов Р.О.
(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 _____
(підпис)

Кобилін О.А.
(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-професійнаОсвітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« ____ » _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУстудентові Маханьову Руслану Олександровичу
(прізвище, ім'я, по батькові)1. Тема роботи Кластеризація часових рядів у задачах аналізу результатів спортсменів швидкісного підводного плавання

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека математичних і статистичних компонентів Apache Commons Math, бібліотека методів для розрахунку динамічного викривлення часу FastDTW, мова програмування Java, середовище розробки IntelliJ IDEA, середовище управління базами даних MySQL, мова програмування JavaScript, стандартизована мова розмітки HTML, стилеві таблиці CSS, шаблонний інтерфейс Thymeleaf.

4. Перелік питань, що потрібно опрацювати в роботі _____

1. Аналіз існуючих методів кластеризації часових рядів.

2. Експериментальні порівняння на різних методах підготовки даних для кластеризації часових рядів.

3. Адаптація стандартних методів до задачі кластеризації часових рядів.

4. Розробка алгоритмів заповнення пропущених значень у часових рядах.

5. Реалізація комп'ютерної моделі для кластеризації часових рядів.

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, стандартні та адаптовані методи кластеризації, вибір методу для реалізації, розробка комп'ютерної моделі, аналіз результатів спортсменів підводного швидкісного плавання.

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	11.04.23-17.04.23	
3	Аналіз літератури з досліджуваної проблеми	18.04.23-20.04.23	
4	Аналіз методів кластеризації часових рядів	21.04.23-30.04.23	
5	Розробка методів кластеризації часових рядів	01.05.23-14.05.23	
6	Програмна реалізація	15.05.23-23.05.23	
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	31.05.23	

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: 67 с., 28 рис., 35 джерел.

КЛАСТЕРИЗАЦІЯ, ЧАСОВІ РЯДИ, НОРМАЛІЗАЦІЯ, КЛАСТЕРНИЙ АНАЛІЗ, ЗВЕДЕННЯ РЯДІВ, ДИНАМІЧНЕ ЗМІЩЕННЯ ЧАСУ.

Об'єктом роботи є послідовність часових рядів, що відображають результати спортсменів підводного плавання.

Метою роботи є розробка методів кластеризації часових рядів, які дозволяють відслідковувати схожість між рядами та формувати кластери для подальшого аналізу результатів спортсменів підводного плавання.

Проведено аналіз існуючих методів кластеризації часових рядів. Проведено експериментальні порівняння на різних методах підготовки даних для кластеризації часових рядів. Розроблено алгоритм кластеризації на базі обраних методів. Розроблено алгоритм заповнення пропущених значень у часових рядах.

У результаті роботи здійснена програмна реалізація системи для кластеризації часових рядів результатів спортсменів підводного плавання, застосовано на реальних даних та проаналізувати отримані результати.

CLUSTERING, TIME SERIES, NORMALISATION, CLUSTER ANALYSIS, SERIES REDUCTION, DYNAMIC TIME WARPING.

The object of the work is a sequence of time series reflecting the results of diving athletes.

The purpose of the work is to develop methods for clustering time series that allow tracking the similarity between the series and forming clusters for further analysis of the results of diving athletes.

An analysis of existing time series clustering methods was carried out. Experimental comparisons on different methods of data preparation for time series clustering were conducted. A clustering algorithm based on the selected methods is developed. An algorithm for filling in missing values in time series is developed.

As a result of the work, a software implementation of the system for clustering time series of diving athletes' results was carried out, applied to real data, and analyzed.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	7
Вступ.....	8
1 Огляд стану проблеми і постановка задачі	9
1.1 Види часових рядів та їх властивості	9
1.1.1 Передобробка часових рядів	11
1.1.2 Типи забруднень часових рядів.....	12
1.1.3 Візуалізація часових рядів.....	13
1.2 Проблеми кластеризації та класифікації часових рядів	14
1.3 Класифікація методів кластеризації часових рядів	16
1.4 Актуальність роботи	18
1.5 Постановка задачі.....	18
2 Огляд методів	20
2.1 Алгоритм кластеризації <i>k-means</i>	20
2.2 Нечіткий алгоритм кластеризації <i>c-means</i> для часових рядів	23
2.3 Алгоритм нормалізації часових рядів	26
2.3.1 Min-Max нормалізація	27
2.3.2 Z-score стандартизація	28
2.4 Методи зведення рядів до фіксованої довжини.....	30
2.4.1 Метод заповнення пропущених значень	30
2.4.2 Інтерполяція	31
2.5 Dynamic Time Warping	31
3 Програмна реалізація	36
3.1 Обґрунтування вибору інструментів програмної реалізації	36
3.1.1 Обґрунтування вибору формату програмної реалізації.....	36
3.1.2 Обґрунтування вибору мови програмної реалізації.....	37
3.1.3 Обґрунтування вибору середовища програмної реалізації ..	39
3.1.4 Обґрунтування вибору бази даних.....	41
3.2 Програмна реалізація	43

	6
3.2.1 Реалізація завантаження даних	43
3.2.2 Реалізація графічного інтерфейсу	45
3.2.3 Реалізація серверної архітектури	46
3.3 Інструкція користувача	47
3.4 Тестування розробленої моделі	50
3.4.1 Порівняння роботи різних методів кластеризації	51
3.4.2 Аналіз впливу нормалізації на кластеризацію	53
3.4.3 Аналіз впливу DTW на кластеризацію	55
3.4.4 Аналіз впливу комбінації DTW та нормалізації	57
3.4.5 Тестування завантаження даних	58
3.4.6 Тестування візуалізації даних	59
Висновки	61
Перелік джерел посилання	63

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DTW – Dynamic Time Warping (динамічне зміщення часу)

TWE – Time Warp Edit (зміщення часу з редагуванням)

JSON – JavaScript Object Notation (мова опису об'єктів JavaScript)

SQL – Structured Query Language (мова структурованих запитів)

SVN – Subversion (централізована система керування версіями)

СУБД – система управління базами даних

РБД – реляційна база даних

REST – Representational State Transfer (представлення стану взаємодії програмного забезпечення)

API – Application Programming Interface (інтерфейс програмування)

CSV – Comma-Separated Values (формат збереження табличних даних, в якому значення розділені комами)

HTML – HyperText Markup Language (мова гіпертекстових міток)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

JAR – Java Archive

Maven – інструмент для управління проєктами та збирання програмного забезпечення у Java

JRE – Java Runtime Environment (середовище виконання Java)

URL – Uniform Resource Locator (адреса, що ідентифікує ресурс у комп'ютерній мережі)

КМС – кандидат у майстри спорту

МС – майстер спорту

БД – база даних

ВСТУП

Сучасний спорт вимагає системного та всебічного аналізу результатів спортсменів для досягнення максимальної ефективності та покращення тренувального процесу. Одним з ключових аспектів аналізу є обробка та інтерпретація часових рядів, які відображають досягнення спортсменів у різних дисциплінах підводного плавання. Зрозуміти закономірності та виявити групи схожих результатів стає важливим завданням для тренерів, науковців та аналітиків у цій галузі.

Основним завданням є виявлення схожих груп результатів, залежностей та встановлення факторів, які впливають на досягнення спортсменів. Це також включає в себе вибір та оптимізацію алгоритмів кластеризації, врахування особливостей часових рядів у спортивному контексті та проведення експериментального аналізу для оцінки ефективності та точності розроблених методів.

Актуальність роботи полягає у тому, що кластеризація часових рядів є потужним інструментом, який дозволяє групувати подібні результати спортсменів, виявляти тенденції та робити прогнози. Це допомагає зрозуміти фактори, які впливають на досягнення спортсменів і визначити оптимальні підходи до тренувань і підготовки. Крім того, використання кластеризації дозволяє зменшити обсяг даних та зробити їх більш зрозумілими та інтерпретованими. Актуальність роботи також обумовлена потребою в удосконаленні процесу аналізу результатів спортсменів підводного плавання. Традиційні методи аналізу можуть бути обмежені тим, що вони не враховують часову структуру даних та не виявляють складні залежності між результатами на різних етапах тренування. Використання кластеризації часових рядів дозволяє поділити спортсменів на групи з подібними характеристиками результатів і виявити скриті закономірності, що сприяє більш точному аналізу та встановленню факторів, що впливають на успішність спортсменів.

1 ОГЛЯД СТАНУ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧІ

В контексті інтелектуального аналізу потоків даних та динамічної обробки інформації виникає необхідність для інспекції проблеми кластеризації часових рядів та наявних методів її вирішення. Тому важливо розглянути різні метрики, що використовуються у методах кластеризації часових рядів, а також врахувати умови недостатньої кількості даних, наявність аномалій та випадки, коли моделі або методи на основі статистики не є придатними. Додатково, важливо розглянути питання ефективності та масштабованості методів кластеризації часових рядів, оскільки потоки даних можуть мати велику кількість точок, що потребує швидкого та оптимального аналізу.

Тобто, існує потреба у розробці нових науково-практичних рішень для інтелектуального аналізу даних, які дозволять ефективно кластеризувати часові ряди в умовах динамічної обробки інформації та забезпечувати якісний аналіз даних.

1.1 Види часових рядів та їх властивості

Часовий ряд – це послідовність спостережень, які відбуваються в певний часовий період, зазвичай відбиваючи динаміку певного явища чи процесу. Кожне спостереження в часовому ряді відноситься до певного моменту часу, тому часові ряди часто відображаються на графіках залежності значень спостережень від часу. Часові ряди зустрічаються в різних галузях науки і техніки, таких як економіка, фінанси, метеорологія, фізика, соціологія та інші. Важливою характеристикою часових рядів є їх стаціонарність або нестаціонарність, яка може впливати на ефективність методів їх аналізу.

Якщо для ряду час є неперервним, то це називається процесом, а якщо час змінюється дискретно, то це дискретний часовий ряд. Часовий ряд, що підлягає аналізу, може бути розглянутий як одна часткова реалізація системи, що навчається та генерується прихованим ймовірнісним або іншим механізмом. Важливо мати на увазі, що аналіз часових рядів може бути складним завданням через їхню властивість змінюватись в часі та може вимагати використання різних методів та технік для досягнення бажаного рівня точності та надійності [1–3].

Існує декілька типів часових рядів:

а) за режимом збору даних:

1) моментний часовий ряд – описує значення явищ в певний момент часу, дискретний ряд;

2) інтервальний часовий ряд – описує значення явищ за певний інтервал часу, містить у собі набір моментних рядів, які можна об'єднати в інтервальний;

3) часовий ряд з подіями – описує події, які відбуваються в певний момент часу;

б) за відношенням між значеннями:

1) абсолютний часовий ряд – вимірює значення в абсолютних одиницях виміру;

2) відносний часовий ряд – вимірює зміну відносно початкового значення;

в) за стаціонарністю:

1) стаціонарний часовий ряд – середнє значення та варіативність не залежать від часу. Це означає, що статистичні показники ряду (середнє значення, варіативність тощо) не змінюються з часом;

2) нестаціонарний часовий ряд – середнє значення та варіативність залежать від часу. Це означає, що статистичні показники ряду змінюються з часом;

г) за залежністю від інших факторів:

1) екзогенний часовий ряд – залежить від зовнішніх факторів, які не є частиною ряду;

2) ендогенний часовий ряд – залежить від внутрішніх факторів, які є частиною ряду.

1.1.1 Передобробка часових рядів

Передобробка часових рядів – це процес обробки та підготовки даних, щоб забезпечити їх ефективне використання в подальшому аналізі. Опрацювання часових рядів є важливим етапом перед їх подальшим використанням, оскільки часто дані можуть бути зашумленими, містити пропуски, відхилення та інші неточності. Опрацювання дозволяє виправити ці неточності та зробити часові ряди готовими до подальшого аналізу. Для цього можуть використовуватись різноманітні методи, такі як згладжування, інтерполяція, видалення аномалій тощо. Опрацьовані часові ряди можуть бути використані в різних задачах, наприклад, в прогнозуванні показників, виявленні аномальних поведінок, класифікації об'єктів та іншому. Також, опрацьовані часові ряди є необхідним етапом перед використанням їх у задачах кластеризації, оскільки непридатні часові ряди можуть призвести до неточних результатів кластеризації.

Зазвичай, передобробка включає в себе:

- видалення пропущених значень – це процес видалення пропущених або відсутніх даних з ряду, які можуть спотворити аналіз;

- видалення аномальних значень – це процес видалення значень, які відрізняються від інших значень ряду та можуть впливати на точність аналізу;

- вирівнювання – це процес приведення ряду до однорідної частоти та періоду, що полегшує порівняння та аналіз між рядами;

- нормалізація – це процес приведення значень ряду до одного масштабу, що полегшує порівняння та аналіз між рядами;
- згладжування – це процес зменшення шуму та випадкової варіації в ряді, щоб показати загальну тенденцію та залежність між значеннями;
- перетворення – це процес зміни форми ряду, щоб полегшити аналіз та отримання корисної інформації.

Ці процеси можуть виконуватися окремо або разом, в залежності від характеру даних та задачі аналізу. Результатом обробки є чистий, зрозумілий та підготовлений для подальшого аналізу часовий ряд.

1.1.2 Типи забруднень часових рядів

У часових рядах можна виділити наступні типи забруднень:

- пропуски даних – це відсутність даних в певний період часу. Пропуски даних можуть бути викликані технічними проблемами, помилками вводу даних, відмовою обладнання, тощо;
- аномальні значення – це значення, які відрізняються від інших значень ряду та можуть бути результатом помилок вимірювань, аномальних подій, відмов обладнання, тощо;
- шум – це випадкові коливання значень ряду, які можуть бути спричинені помилками вимірювання, відмінностями в середовищі, змінами внутрішніх чинників, тощо;
- сезонність – це періодичні коливання в значеннях ряду, які повторюються через певні періоди часу;
- тренд – це загальна тенденція зростання або спаду значень ряду з часом;
- кореляція – це залежність між значеннями ряду та іншими факторами, такими як погода, економічні умови, тощо;

– випадковість – це непередбачувані зміни в значеннях ряду, які не мають зв'язку з жодним іншим фактором.

Ці типи забруднень можуть впливати на точність та надійність аналізу часових рядів, тому перед їх аналізом важливо виконати процес передобробки даних, що включає в себе видалення аномалій та пропусків даних, а також згладжування та перетворення даних для зменшення впливу шуму та інших видів забруднень.

1.1.3 Візуалізація часових рядів

Існує два основні підходи до візуалізації часових рядів: накладені графіки та відокремлені графіки (рис. 1.1). Накладені графіки відображають кілька часових рядів на одному графіку, що дозволяє порівняти їх поведінку в часі. Однак, цей підхід може стати незручним, якщо ряди мають різний масштаб або велику амплітуду коливань. У відокремлених графіках кожен ряд відображається на окремому графіку, що дозволяє краще побачити деталі та особливості кожного ряду. Правильний вибір підходу до візуалізації часових рядів є важливим етапом в опрацюванні даних, що може значно впливати на подальші результати аналізу.

Окрім накладених та відокремлених графіків, існують інші методи відображення часових рядів, наприклад:

- графіки зміни середньої – цей тип графіку демонструє зміну середнього значення часового ряду в залежності від часу;
- графіки зміни розмаху – цей тип графіку відображає зміну розмаху значень часового ряду в залежності від часу;
- графіки зміни відсоткової зміни – цей тип графіку відображає зміну відсоткової зміни значень часового ряду в залежності від часу;
- графіки зміни відносної зміни – цей тип графіку відображає зміну відносної зміни значень часового ряду в залежності від часу;

– графіки автокореляції – цей тип графіку відображає ступінь кореляції між значеннями часового ряду в різні моменти часу.



Рисунок 1.1 – Види графіків для візуалізації часових рядів

1.2 Проблеми кластеризації та класифікації часових рядів

Зараз методи кластеризації, що використовуються для обробки часових рядів, мають деякі суттєві недоліки, такі як чутливість до змін в ряді, обробка надмірної інформації та висока вартість. Метою застосування кластеризації для часових рядів є необхідність виявлення невідомих закономірностей в ряді, оскільки ряди не є періодичними, і значення, які містяться у ряді, не можуть бути розпізнані через надмірну або недостатню кількість інформації [4]. Тому одним з основних завдань в області інтелектуального аналізу даних є кластеризація, а саме розбиття вихідних даних на однорідні групи в режимі самонавчання.

Проблеми кластеризації та класифікації часових рядів включають в себе наступні аспекти:

- вибір критеріїв для групування – для кластеризації та класифікації часових рядів потрібно визначити певні ознаки, за якими будуть групуватися ряди. Вибір цих ознак може впливати на якість результатів;

- обробка інформації про часові ряди – для класифікації та кластеризації часових рядів потрібно обробити дані, виконати їх попередню обробку та підготовку, щоб вони були придатні для аналізу;

- вибір методів кластеризації та класифікації – для кластеризації та класифікації часових рядів можна використовувати різні методи, такі як методи на основі подібності, методи на основі відстані, методи на основі статистичних моделей, тощо. Вибір методів може впливати на якість результатів;

- оцінка якості результатів – для кластеризації та класифікації часових рядів потрібно провести оцінку якості результатів, щоб переконатися, що вони є достатньо точними та надійними;

- наявність шуму та випадкових змін – часові ряди можуть містити шум та випадкові зміни, що можуть вплинути на результати кластеризації та класифікації. Для підвищення якості результатів необхідно виконати передобробку даних, що включає згладжування та видалення шуму [5, 6].

Кластеризація є проблемою неконтрольованого навчання, що полягає у виявленні прихованої структури у сукупності даних та організації об'єктів у групи, члени яких мають певну міру подібності. Кластери можуть бути представлені як скупчення точок у просторі змінних, форма яких може бути різною (рис. 1.2). Цей процес може бути виконаний без застосування вчителя, тобто без наявності чіткої позначки класу для кожного об'єкта [7].

Кластеризація часових рядів відрізняється від кластеризації об'єктів в евклідовому просторі тим, що кластери з часовими рядами можуть мати різну довжину. Якщо всі часові ряди мають однакову довжину, то можна використовувати стандартні методи кластеризації, представивши кожен

часовий ряд у вигляді вектора та використовуючи звичайну відстань норми. Однак, такий підхід можна використовувати лише при схожості у часі, не враховуючи подібності за формою та в зміні значень.

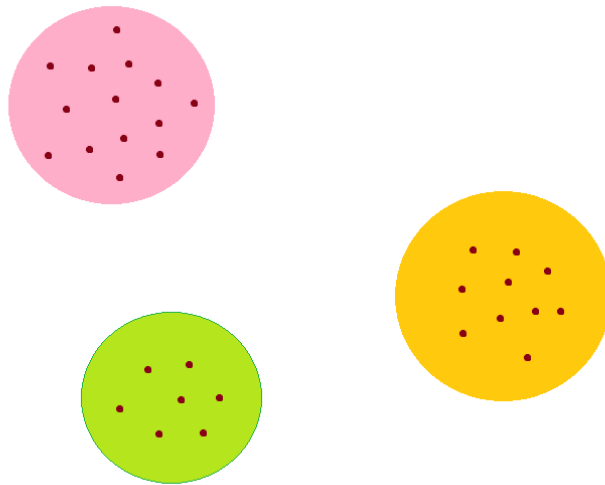


Рисунок 1.2 – Кластеризація

1.3 Класифікація методів кластеризації часових рядів

Проблема вибору ознак взаємопов'язана з проблемою визначення кластерів набору функцій [8]. Наразі існує кілька методів кластеризації (рис. 1.3), які використовують підмножини ознак, щоб максимізувати основну тенденцію кластеризації.

Кластеризація часових рядів є складною задачею через труднощі у визначенні подібності між різними рядами, які можуть бути масштабовані як у часових, так і у властивісних вимірах. Тому концепція подібності є важливою для кластеризації часових рядів. Коли міра подібності визначена для часу, ряди даних можна розглядати як абстрактні об'єкти, на яких застосовуються методи, засновані на подібності, такі як спектральні методи або методи розбиття [9–11]. Такі часові ряди дозволяють використовувати

різноманітні формулювання для процесу кластеризації в залежності від того, чи групуються ряди на основі їх онлайн-кореляцій, чи на основі їх форми.

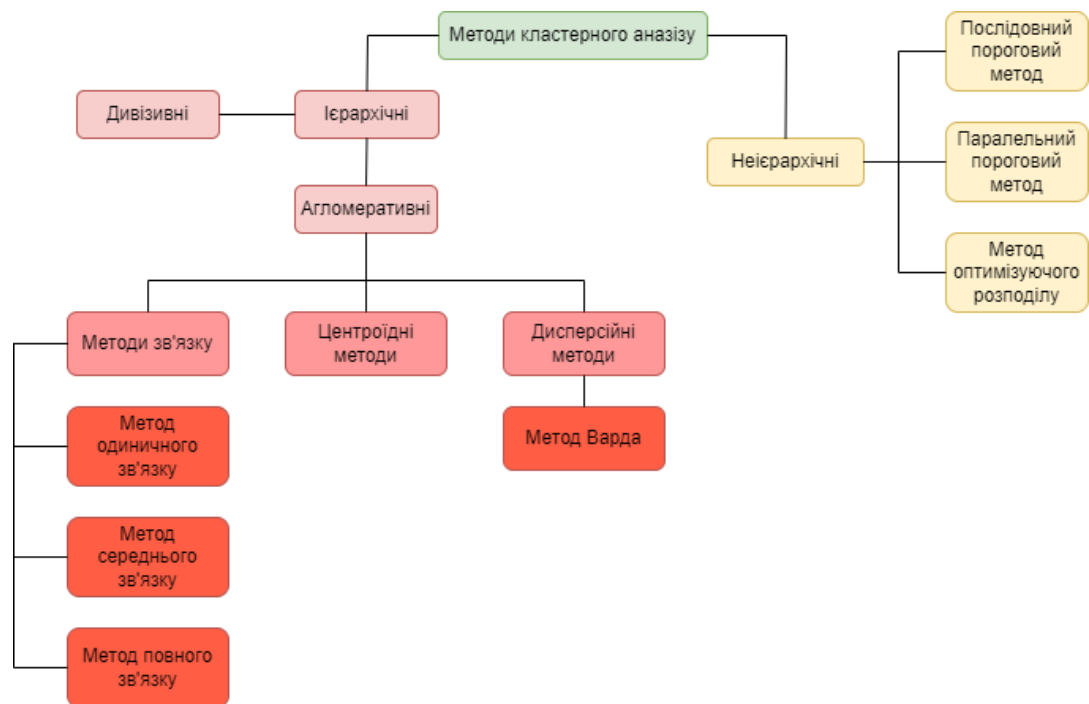


Рисунок 1.3 – Методи кластерного аналізу

Метою кластеризації часових рядів є створення кластерів, що мають високу внутрішню кластерну подібність, але низьку схожість між різними кластерами. Це важливо для всіх типів даних, але особливо висока розмірність, наявність шуму та висока кореляція можуть ускладнювати завдання кластеризації [12, 13]. Розробка ефективних алгоритмів кластеризації є важливим завданням у зв'язку з цим.

Також важливою характеристикою є необхідність виконання таких задач у режимі реального часу, оскільки потоки даних постійно змінюються. Таким чином, необхідно виявляти значущі зміни в поведінці, що дає уявлення про те, як потік даних змінюється з часом. Для досягнення цієї мети, використовують алгоритми кластеризації, що підтримують інкрементальне навчання. Адже вони здатні пристосовуватись до нових даних, додавати або видаляти кластери при надходженні нових спостережень.

1.4 Актуальність роботи

Кластеризація часових рядів у задачах аналізу результатів спортсменів підводного плавання має велику актуальність у сучасному спортивному аналізі. З одного боку, підводне плавання – це складна спортивна дисципліна, в якій дуже важливо аналізувати результати змагань та тренувань, щоб виявляти сильні та слабкі сторони спортсменів та розробляти ефективні стратегії тренувань. З іншого боку, часові ряди даних, зібраних в ході тренувань та змагань, можуть містити велику кількість інформації, що потребує аналізу та обробки.

Застосування методів кластеризації часових рядів дозволяє здійснювати швидкий та ефективний аналіз даних спортивних змагань та тренувань. Наприклад, ці методи можуть допомогти виявити зв'язки між показниками спортивної форми та результатами змагань, зробити прогнози щодо подальшого розвитку спортсменів, визначити ефективні способи тренувань та покращення результатів. Також кластеризація даних може допомогти зменшити обсяги даних та підвищити швидкість обробки інформації, що є особливо важливим у випадку великих обсягів даних.

1.5 Постановка задачі

Кластеризація часових рядів у задачах аналізу результатів спортсменів підводного плавання є важливим та актуальним напрямом у сучасному спортивному аналізі.

Об'єктом роботи є послідовність часових рядів, що відображають результати спортсменів підводного плавання.

Метою роботи є розробка методів кластеризації часових рядів, які дозволяють відслідковувати схожість між рядами та формувати кластери для подальшого аналізу результатів спортсменів підводного плавання.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз існуючих методів кластеризації часових рядів;
- зібрати і підготувати дані часових рядів з результатами спортсменів підводного плавання для подальшої кластеризації;
- провести експериментальні порівняння на різних методах підготовки даних для кластеризації часових рядів;
- розробити алгоритм кластеризації на базі обраних методів;
- розробити алгоритм заповнення пропущених значень у часових рядах;
- реалізувати комп'ютерну модель для кластеризації часових рядів та аналізу результатів спортсменів підводного плавання;
- застосувати розроблені алгоритми та програму на реальних даних результатів спортсменів підводного плавання та проаналізувати отримані результати.

2 ОГЛЯД МЕТОДІВ

Під час аналізу методів кластеризації даних було виявлено, що застосування стандартних методів не є найкращим варіантом для кластеризації часових рядів. Це зумовлено тим, що часові ряди мають свої особливості, такі як наявність трендів, сезонності, змін варіабельності та інших факторів, які не враховуються в стандартних методах.

У зв'язку з цим, для кластеризації часових рядів використовують більш спеціалізовані методи, які враховують особливості цих даних. Такі методи включають в себе алгоритми, які базуються на відстані між часовими рядами, такі як DTW та TWE, а також методи, які використовують додаткову інформацію про часові ряди, такі як графічні моделі та методи, що використовують спектральні характеристики.

Крім того, для кластеризації часових рядів можна адаптувати стандартні методи, використовуючи різноманітні інструменти, такі як метрики відстані, які враховують особливості часових рядів, функції ядра, які забезпечують нелінійність у просторі ознак, та інші техніки.

2.1 Алгоритм кластеризації *k*-means

Алгоритм кластеризації *k*-means є одним з найпоширеніших методів кластеризації даних. Це ітеративний алгоритм, який спробує розбити набір даних на *k* кластерів, де *k* – заздалегідь визначена кількість кластерів. На рисунку 2.1 наведено приклад роботи алгоритму *k*-means.

Алгоритм кластеризації *k*-means складається з таких кроків:

Крок 1. Ініціалізація: спочатку вибираємо кількість кластерів, яку хочемо створити, та розміщуємо центри цих кластерів випадковим чином в межах розглянутої області. Основна задача методу *k*-means полягає у тому,

максимально зменшити значення сумарного квадратичного відхилення від центрів до точок цих кластерів.

$$\min[\sum_{i=1}^n \sum_{x^j \in N^i} \rho(x^j, c_i)^2], \quad (2.1)$$

де n – кількість кластерів;

N^i – список елементів кластеру;

ρ – функція обчислення метрик;

c_i – центр поточного кластеру.

Базовою метрикою для обчислення відстані є Евклідова відстань. Евклідова відстань – це метрика, яка вимірює відстань між двома точками в n -вимірному просторі. У контексті кластеризації, Евклідова відстань може використовуватися для обчислення відстані між точками даних, що належать до різних кластерів [14].

$$\rho(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}. \quad (2.2)$$

Крок 2. Розподіл даних: кожен елемент даних призначається до ближчого кластеру на основі відстані між центром кластеру та елементом.

$$l_i = \arg \min \rho(x_i, c_l), \quad (2.3)$$

де ρ – евклідова відстань;

x_i – певний елемент;

c_l – центр поточного кластеру.

Крок 3. Перерахунок центрів: для кожного кластеру обчислюємо новий центр як середнє арифметичне всіх елементів кластеру за формулою:

$$c_l = \frac{1}{N_l} \sum_{x^l \in N_l} x^l, \quad (2.4)$$

де N_l – кількість елементів поточного кластеру;

x^l – елемент поточного кластеру.

Крок 4. Перевірка зупинки: перевіряємо, чи змінилися центри кластерів після Крок 2 та Крок 3. Якщо ні, то процес завершується, інакше переходимо до Крок 2.

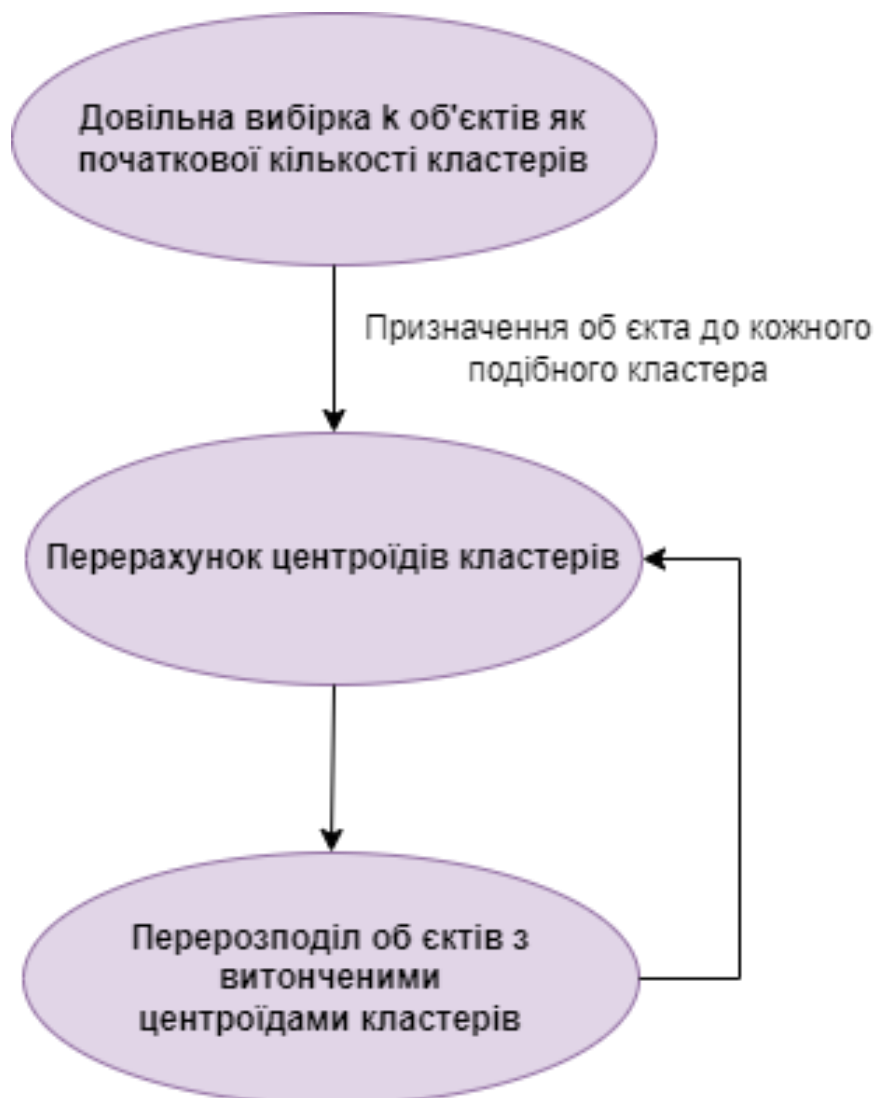


Рисунок 2.1 – Приклад роботи алгоритму *k-means*

Головним недоліком методу *k-means* є те, що матриця належності містить дискретну множину значень, що часто призводить до неправильних рішень [15–17]. Складність алгоритму *k-means* становить:

$$O(k \cdot N \cdot r \cdot D), \quad (2.5)$$

де k – кількість очікуваних кластерів;

N – кількість об'єктів для кластеризації;

r – кількість ітерацій для збіжності;

D – розмірність простору об'єктів.

Неадекватний вибір початкового числа кластерів k може призвести до неправильних результатів [18, 19]. Тому перед застосуванням методу *k-means* важливо визначити оптимальну кількість кластерів для даного набору даних.

2.2 Нечіткий алгоритм кластеризації *c-means* для часових рядів

Алгоритм нечіткої кластеризації *c-means* для часових рядів – це метод кластеризації, який використовується для групування часових рядів на основі схожості їх форми і динаміки. На рисунку 2.2 наведено приклад роботи алгоритму *c-means*.

Алгоритм складається з наступних кроків:

Крок 1. Вибір кількості кластерів, яку необхідно створити.

Крок 2. Випадкове ініціалізування центрів кластерів.

Крок 3. Обчислення ступеня відповідності кожного часового ряду до кожного кластеру, що базується на подібності їх форми та динаміки. Для цього використовується функція належності, яка визначає, наскільки даний часовий ряд належить до кожного кластеру.

$$r_{ij} = \frac{N(d(x_i, c_j) | \mu = 0, \sigma)}{\sum_j^k N(d(x_i, c_j) | \mu = 0, \sigma)}, \quad (2.6)$$

де c_j – центр поточного кластеру;

x_i – певний елемент;

$d(x_i, c_j)$ – відстань між точками x_i і c_j ;

N – щільність ймовірності нормального розподілу у точці $d(x_i, c_j)$.

Крок 4. Обчислення нових центроїдів кластерів на основі належності кожного часового ряду до кожного кластеру.

$$c_j \leftarrow \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}, \quad (2.7)$$

де c_j – центр поточного кластеру;

x_i – певний елемент;

r_{ij} – матриця належності елементів до кластерів r .

Крок 5. Розрахувати функцію втрат.

$$J = \sum_j^k \sum_i^N d(x_i, c_j)^2 r_{ij}, \quad (2.8)$$

де c_j – центр поточного кластеру;

x_i – певний елемент;

$d(x_i, c_j)$ – відстань між точками x_i і c_j ;

r_{ij} – матриця належності елементів до кластерів r .

Якщо значення функції втрат зменшується, то необхідно повторити цикл з Крок 2.

Крок 6. Кінцеве призначення кожного часового ряду до одного з кластерів на основі його найбільшої належності до цього кластеру.

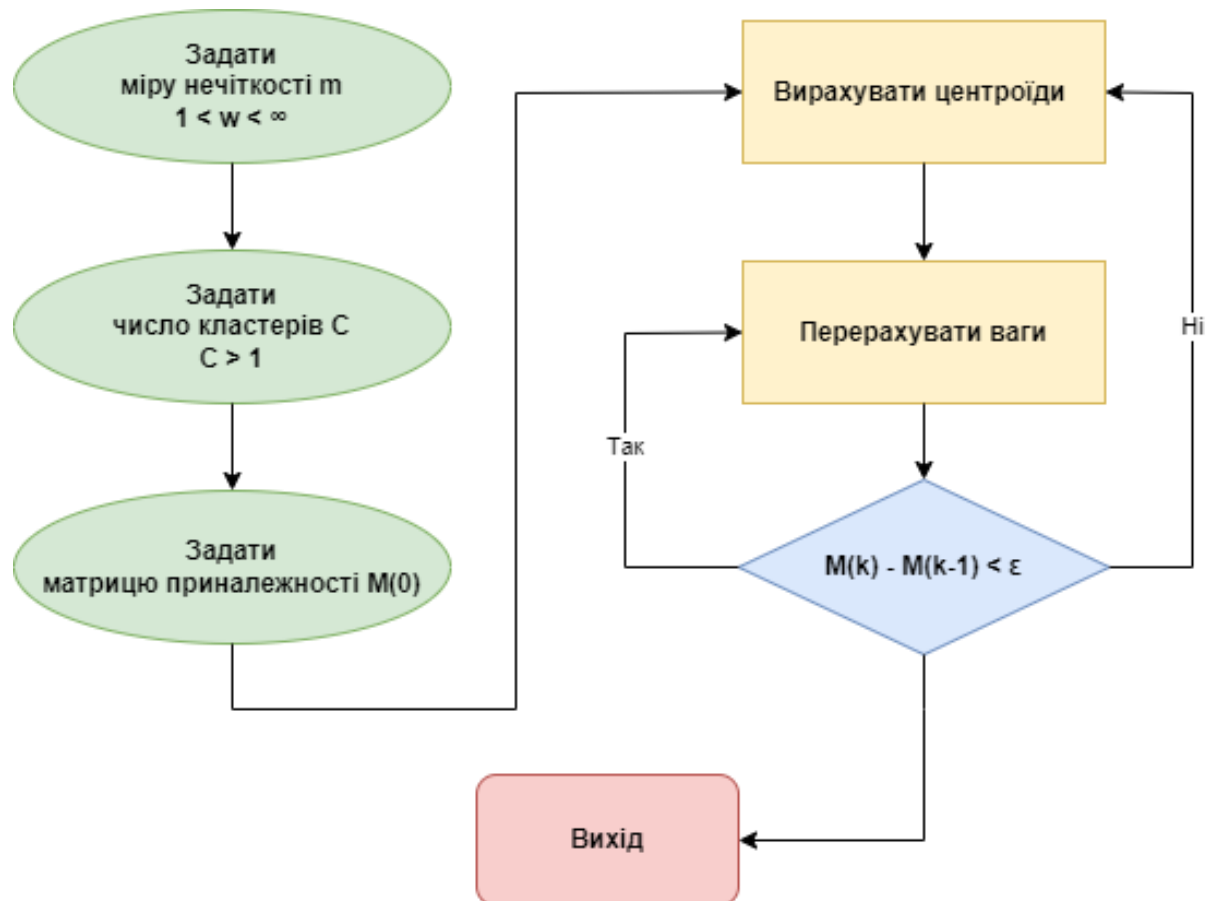


Рисунок 2.2 – Приклад роботи алгоритму c-means

Алгоритм c-means має квадратичну складність від кількості об'єктів та кубічну складність від кількості кластерів. Таким чином, загальна складність алгоритму c-means складає:

$$O(N^2k^3), \quad (2.9)$$

де N – кількість об'єктів;

k – кількість кластерів.

В основі алгоритму лежить використання нечітких множин, що дозволяє розподілити кожен часовий ряд між кластерами з різною вагою залежно від ступеня подібності до цих кластерів. Крім того, використання функції належності дозволяє враховувати невизначеність даних та зменшити вплив шумів [20–22].

Алгоритм нечіткої кластеризації *c-means* для часових рядів є ефективним методом кластеризації, який дозволяє групувати часові ряди за їх формою та динамікою з урахуванням невизначеності даних та шумів [23].

Взаємозв'язок між кластерним аналізом та теорією нечітких множин полягає в тому, що вони можуть бути застосовані для структуризації складних систем, де більшість об'єктів мають неточну природу. Така неточність полягає у тому, що перехід від належності до неналежності елементів в даних класах може бути поступовим. У випадку, коли використовується міра схожості у формі відстані, величина належності об'єкта до кластера залежить від його схожості з прототипом цього кластера, причому вона обернено пропорційна відстані об'єкта до центроїда кластера.

2.3 Алгоритм нормалізації часових рядів

Нормалізація часових рядів – це процес перетворення значень часового ряду в межі від 0 до 1, що дозволяє знизити вплив різних масштабів даних на алгоритми аналізу часових рядів.

Використання нормалізованих часових рядів дає можливість порівнювати різні ряди, що мають різні масштаби, та зробити їх більш зрозумілими для аналізу. Наприклад, якщо порівнювати досягнення спортсменів з різних змагань, то можуть бути використані різні шкали для позначення рівня складності цих змагань. Нормалізація дозволяє привести значення до єдиного масштабу та зробити їх зрозумілішими для порівняння. Крім того, нормалізація даних може покращити роботу алгоритмів

машинного навчання та кластеризації, оскільки деякі алгоритми можуть бути чутливими до масштабу даних та відносних значень між атрибутами [24–26].

Для нормалізації часових рядів існують метод нормалізації Min-Max або метод стандартизації Z-score. Обидва методи мають свої переваги та недоліки, але обидва можуть бути корисними в залежності від ситуації.

Проведення нормалізації є важливим етапом перед кластеризацією даних, включаючи часові ряди. Це необхідно для того, щоб уникнути проблеми, коли деякі атрибути мають значення, які великою мірою відрізняються від інших, і це може впливати на результати кластеризації.

2.3.1 Min-Max нормалізація

Min-Max нормалізація – це простий метод масштабування даних, який приводить всі значення до відрізка між 0 та 1. Цей метод корисний, коли необхідно перетворити різні функції, що мають різні діапазони значень, на спільний масштаб для спрощення порівняння або обробки. Використання Min-Max нормалізації може покращити результати деяких алгоритмів машинного навчання, особливо тих, що чутливі до масштабування вхідних даних.

Min-Max нормалізація здійснюється за наступною схемою:

Крок 1. Визначаємо мінімальне ($x_{\min}$) та максимальне ($x_{\max}$) значення даних.

Крок 2. За формулою Min-Max нормалізації знайдемо нормалізоване значення для кожного x у наборі даних:

$$x_{\text{norm}} = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}, \quad (2.10)$$

де x_{norm} – нормалізоване значення;

x – поточне значення;

$x_{\min}$ – мінімальне значення в часовому ряді;

$x_{\max}$ – максимальне значення в часовому ряді.

Недолік Min-Max нормалізації полягає в тому, що цей метод чутливий до викидів (outliers) у даних. Викиди можуть зміщувати мінімальні та максимальні значення, внаслідок чого інші значення стискаються у вузькому діапазоні. Це може призвести до згладжування особливостей даних та втрати інформації. У таких випадках можна використовувати інші методи нормалізації, такі як Z-score стандартизація.

2.3.2 Z-score стандартизація

Z-score стандартизація, також відома як нормалізація за допомогою середнього та стандартного відхилення, перетворює дані таким чином, що середнє значення стає 0, а стандартне відхилення – 1. Це дозволяє перетворити різні функції на спільний масштаб, що полегшує порівняння та обробку даних. Z-score стандартизація корисна для алгоритмів машинного навчання, що чутливі до масштабування вхідних даних, а також цей метод менш чутливий до викидів, порівняно з Min-Max нормалізацією.

Аби провести Z-score стандартизацію, для початку необхідно обчислити середнє значення та стандартне відхилення набору даних. Стандартне відхилення є мірою розподілення значень у часовому ряді або наборі даних. Воно допомагає визначити, наскільки індивідуальні значення відхиляються від середнього значення.

Z-score стандартизація здійснюється за наступною схемою:

Крок 1. Розрахувати середнє значення значень, що містяться в часовому ряді.

$$\mu = \frac{\sum x}{N}, \quad (2.11)$$

де $\sum x$ – сума всіх значень у часовому ряду;

N – кількість значень у часовому ряді.

Крок 2. Обчислити стандартне відхилення часового ряду.

$$\sigma = \sqrt{\frac{\sum (x - \mu)^2}{N}}, \quad (2.12)$$

де x – значення у часовому ряді;

μ – середнє значення часового ряду;

N – кількість значень у часовому ряді.

Крок 3. За формулою Z-score стандартизації знайдемо стандартизоване значення для кожного x у наборі даних:

$$x_{std} = \frac{(x - \mu)}{\sigma}, \quad (2.13)$$

де x_{std} – стандартизоване значення;

x – поточне значення;

μ – середнє значення часового ряду;

σ – стандартне відхилення часового ряду.

Недоліками Z-score стандартизації є те, що цей метод не гарантує, що дані будуть обмежені певним діапазоном, як у випадку з Min-Max нормалізацією. Це може створювати проблеми для деяких алгоритмів, які очікують обмежені вхідні дані. Також, якщо розподіл даних дуже скошений або має тяжкі хвости, стандартизація може не бути достатньо ефективною для перетворення даних до більш рівномірного масштабу.

2.4 Методи зведення рядів до фіксованої довжини

Якщо часові ряди мають різну довжину, тоді можна використовувати методи зведення рядів до фіксованої довжини. Це є необхідним умовою для використання стандартних методів кластеризації, оскільки більшість алгоритмів припускає, що всі вектори мають однакову довжину [27, 28].

Існує декілька методів зведення рядів до фіксованої довжини, таких як метод заповнення пропущених значень і інтерполяція.

2.4.1 Метод заповнення пропущених значень

Метод заповнення пропущених значень є простим і ефективним способом зведення ряду до фіксованої довжини. Він полягає в тому, щоб заповнити відсутні значення в ряді значеннями, отриманими за допомогою попередніх і наступних значень. Цей метод особливо корисний для даних, що мають пропущені значення через відсутність даних або помилки вимірювання [29].

$$x_i = \frac{x_{i-1} + x_{i+1}}{2}, \quad (2.14)$$

де x_i – пропущене значення;

x_{i-1} – попереднє значення;

x_{i+1} – наступне значення.

Заповнення пропущених значень дозволяє зберегти більш повну інформацію з ряду і покращити аналіз результатів. Проте варто зазначити, що заповнення пропущених значень може мати вплив на статистичні властивості ряду та результати аналізу.

2.4.2 Інтерполяція

Інтерполяція – це більш складний метод зведення рядів до фіксованої довжини. Цей метод полягає в тому, щоб побудувати математичну модель, яка зможе описати ряд і заповнити пропущені значення, використовуючи наявні дані. Інтерполяція може бути лінійною або нелінійною, залежно від математичної моделі.

Вибір методу інтерполяції для заповнення пропущених значень в часових рядах залежить від властивостей самого ряду та конкретної задачі. Не існує універсального методу, який підійде для всіх типів рядів та випадків.

Для стаціонарних часових рядів зі змінною дисперсією часто використовують лінійну інтерполяцію або метод заповнення пропущених значень значеннями з попереднього або наступного часового кроку. Формула для лінійної інтерполяції між двома точками (x_1, x_2) та (y_1, y_2) має вигляд:

$$y = y_1 + (y_2 - y_1) \cdot \frac{(x - x_1)}{(x_2 - x_1)}, \quad (2.15)$$

де x – точка, для якої потрібно знайти значення y .

Для нестационарних рядів можуть використовуватися більш складні методи, такі як кубічна інтерполяція або метод згладжування.

2.5 Dynamic Time Warping

Dynamic Time Warping – це алгоритм, який вимірює схожість між двома часовими рядами, навіть якщо вони мають різну довжину або швидкість. DTW шукає оптимальний шлях відповідності між двома часовими рядами, мінімізуючи сумарну відстань між відповідними точками. Цей

алгоритм особливо ефективний коли існують деформації у часі або коли часові ряди мають різну довжину.

У методах кластеризації, що використовують евклідову відстань, відстань між двома точками розраховується як квадратний корінь з суми квадратів різниць між відповідними значеннями на кожному часовому кроці. У випадку з DTW, замість цього використовується алгоритм динамічного програмування, який знаходить оптимальний шлях відповідності між двома рядами за допомогою локальних зміщень у часі [30, 31].

Таким чином, для адаптації методів кластеризації до використання DTW потрібно замінити значення евклідової відстані на значення DTW у формулах, які визначають відстань між кластерами, а також у формулах, які визначають центри кластерів.

Розрахунок DTW здійснюється за наступною схемою:

Крок 1. Задати два часових ряди x та y довжини n та m відповідно.

Крок 2. Ініціалізувати матрицю відстаней D розміром $(n + 1) \cdot (m + 1)$.

$$\begin{aligned} D[0,0] &= 0 \\ D[i,0] &= D[0,j] = \infty, \end{aligned} \tag{2.16}$$

де $i = 1..n$;

$j = 1..m$.

Крок 3. Оцінити відстань між двома точками за допомогою евклідової метрики.

$$d(x[i-1], y[j-1]) = (x[i-1] - y[j-1])^2. \tag{2.17}$$

Крок 4. Обчислити матрицю відстаней, тобто знайти відстані між кожною парою часових рядів.

$$D[i, j] = d(x[i-1], y[j-1]) + \min(D[i-1, j], D[i, j-1], D[i-1, j-1]), \quad (2.18)$$

де $i = 1..n$;

$j = 1..m$;

$d(x[i-1], y[j-1])$ – відстань між точками $x[i-1]$ та $y[j-1]$.

Крок 5. Отримати відстань DTW між часовими рядами x та y , яка дорівнює значенню у нижньому правому куті матриці відстаней.

$$DTW(x, y) = \sqrt{D[n, m]}. \quad (2.19)$$

Варіанти алгоритму DTW можуть включати додаткові обмеження, такі як шаблони вікон або локальні обмеження шляху, для прискорення обчислень та покращення якості відповідності.

Для часових рядів DTW може бути кращою метрикою, ніж Евклідова відстань, оскільки DTW дозволяє враховувати різницю у швидкості та часовому зміщенні між двома рядами, тоді як Евклідова відстань не бере ці фактори до уваги. Зважаючи на це, DTW може бути кращою метрикою для часових рядів, оскільки вона враховує зміщення часу та може допомогти вирішити проблему зі швидкістю та різними часовими шкалами між двома рядами. Окрім гнучкості до зсувів та масштабування, DTW має ще кілька переваг над Евклідовою відстанню в контексті аналізу часових рядів:

- DTW враховує нелінійні залежності між точками часового ряду, що дає змогу знаходити більш складні відносини між даними;
- DTW може працювати з часовими рядами різної довжини, в той час як Евклідова відстань потребує, щоб обидва ряди були однакової довжини;
- DTW може працювати з часовими рядами, що містять пропуски або шум, що не є можливим при використанні Евклідової відстані;

- DTW може бути використаний для кластеризації та класифікації часових рядів, оскільки він дає змогу порівнювати ряди, які мають різні форми та траєкторії;
- DTW може бути використаний для прогнозування часових рядів, оскільки він дає змогу знаходити схожість між часовими рядами та давати підказки щодо подальшого руху ряду.

Якщо порівняти візуально графіки Евклідової відстані і DTW, то можна побачити, що графік Евклідової відстані (рис. 2.3) буде досить гладким, з деякими різкими збільшеннями відстані на певних ділянках, що відповідає великій різниці між значеннями на цих ділянках. Тоді як графік DTW (рис. 2.4), навпаки, буде нерівномірним і зміщеним, що відображає те, що динаміка часового ряду може відрізнятись на деяких ділянках, але все ж залишатися схожою в цілому.

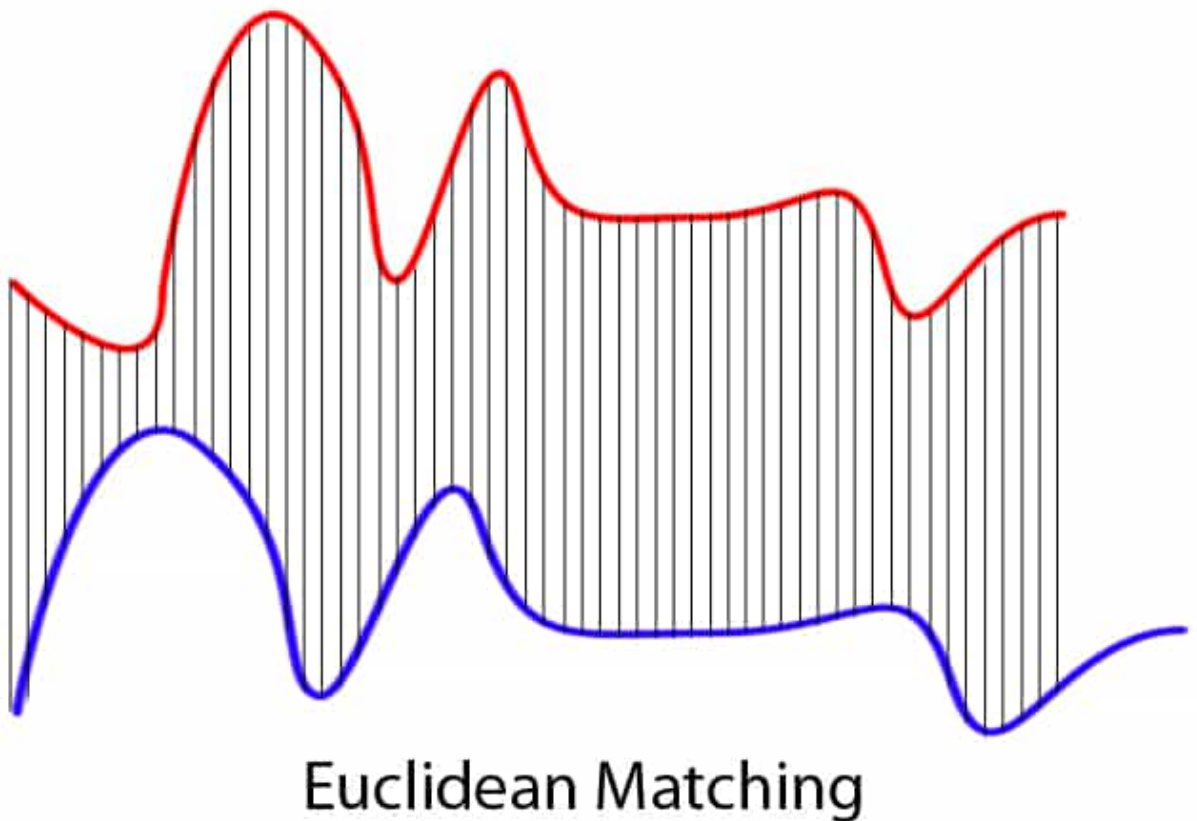
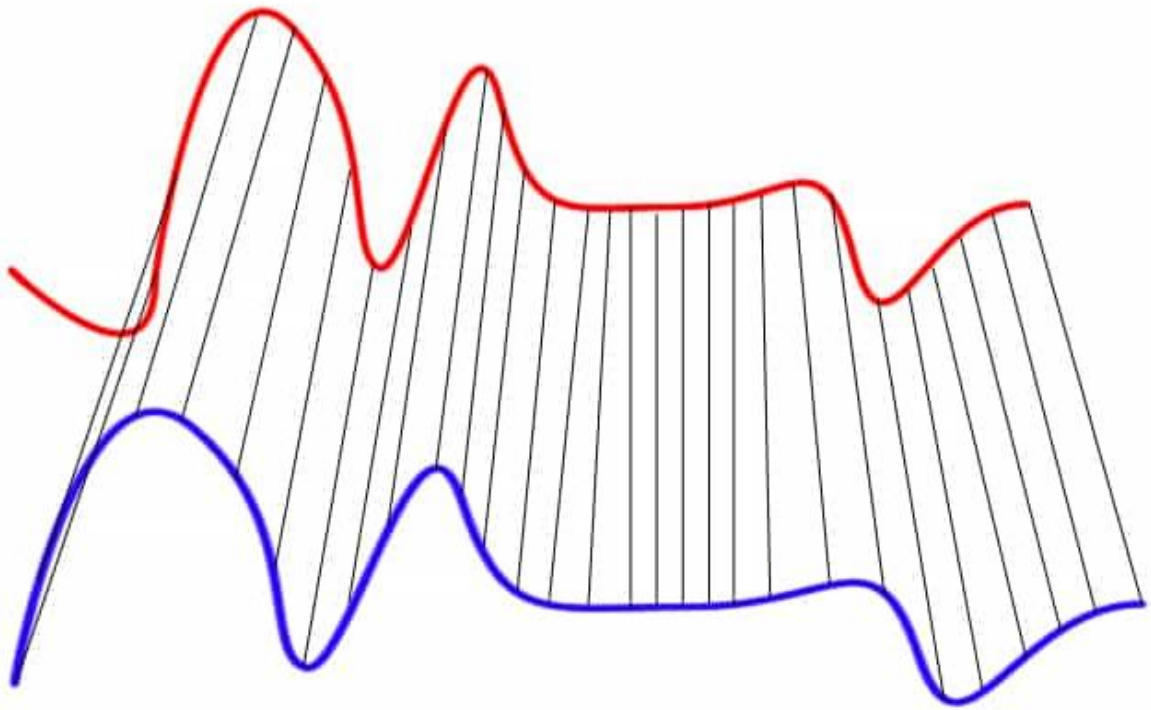


Рисунок 2.3 – Графік Евклідової відстані



Dynamic Time Warping Matching

Рисунок 2.4 – Графік DTW

Отже, метод динамічного зміщення часу є корисним і ефективним підходом до порівняння та аналізу часових рядів. Він має переваги над традиційною Евклідовою відстанню, оскільки враховує зміщення часу, швидкість і різні часові шкали між рядами. DTW здатний виявляти нелінійні залежності, працювати з рядами різної довжини, включати пропущені значення та шум, і дозволяє класифікувати, кластеризувати та прогнозувати часові ряди.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Обґрунтування вибору інструментів програмної реалізації

У рамках кваліфікаційної роботи був розроблений застосунок, що дозволяє тренеру завантажувати, оновлювати та кластеризувати дані для подальшого аналізу та графічної візуалізації, що полегшить взаємодію з цими даними.

3.1.1 Обґрунтування вибору формату програмної реалізації

Аби зробити застосунок максимально доступним було вирішено зробити його у формфакторі вебзастосунку.

Існує кілька переваг використання вебзастосунків порівняно з десктопними застосунками. Ось основні з них:

- вебзастосунки можна відкривати з будь-якого пристрою з підключенням до Інтернету, що робить їх доступними з будь-якої точки світу. У десктопних застосунках не завжди є можливість дистанційного доступу;
- оновлення вебзастосунків відбуваються автоматично на серверах, що робить їх більш безпечними та оновленими. У десктопних застосунках потрібно вручну встановлювати оновлення;
- вебзастосунки можна масштабувати без встановлення додаткового обладнання або програмного забезпечення. У десктопних застосунках зазвичай потрібно встановлювати додаткове обладнання, щоб масштабувати функціональність;
- вебзастосунки можуть працювати на будь-якому операційному середовищі, що робить їх більш сумісними з різними пристроями та

операційними системами. У десктопних застосунках не завжди можна забезпечити сумісність з різними операційними системами;

- вебзастосунки зазвичай потребують менше інвестицій у розробку та підтримку порівняно з десктопними застосунками;

- вебзастосунки можуть легко інтегруватись з іншими вебсервісами, що робить їх більш функціональними та корисними. У десктопних застосунках інтеграція може бути більш складною та вимагати більшої кількості ресурсів.

3.1.2 Обґрунтування вибору мови програмної реалізації

Java є однією з найбільш популярних мов програмування для реалізації вебзастосунків. Перша причина, чому використання Java є вигідним для розробки вебзастосунків це платформна незалежність. Java є мовою програмування, яка працює на будь-якій платформі, що підтримує віртуальну машину Java. Це означає, що вебзастосунок, написаний на Java, може працювати на будь-якій операційній системі без необхідності виконувати додаткові налаштування. Наступна причина – це багата стандартна бібліотека. Java має велику стандартну бібліотеку, що дозволяє розробникам створювати вебзастосунки з високою продуктивністю та надійністю. Бібліотека містить багато інструментів для маніпулювання рядками, роботи з мережами, обробки даних та іншого.

Ще одним плюсом є об'єктно-орієнтована парадигма. Java є мовою програмування, яка базується на об'єктно-орієнтованій парадигмі, що дозволяє розробникам створювати більш структуровані та гнучкі вебзастосунки. Об'єктно-орієнтований підхід дозволяє розробникам створювати модульні та легко розширювані застосунки.

Не варто забувати про високу продуктивність. Java є мовою програмування, яка дозволяє розробникам створювати високопродуктивні

вебзастосунки. Java використовує віртуальну машину, що дозволяє забезпечити високу швидкість виконання коду та ефективне використання ресурсів.

Також, розробнику доступний великий вибір фреймворків. Java має великий вибір фреймворків, таких як Spring, Struts, Hibernate та інші, що спрощують розробку вебзастосунків та дозволяють швидко розгорнути функціональність. І останнє, що є не менш важливим – це велика спільнота розробників. Java має велику спільноту розробників, які допомагають один одному вирішувати проблеми та надають широкий вибір різних інструментів, бібліотек та фреймворків.

Для зручного використання Java проєкту, саме у форматі вебзастосунку, краще за все використовувати фреймворк – Spring Boot Web. Цей фреймворк є одним з найбільш популярних фреймворків для розробки та розгортання вебзастосунків на Java, що має кілька переваг порівняно з іншими фреймворками. Серед головних переваг можна виділити швидкість розробки та простий синтаксис, який дозволяє розробникам швидко розгорнути та налаштувати вебзастосунок. Spring Boot Web також має вбудовану функціональність, таку як кешування, збереження сесій, підтримка формату JSON та інші, що дозволяє розробникам швидко додавати функціональність до свого застосунок.

Фреймворк підтримує середовище Docker, що дозволяє розробникам розгортати свій вебзастосунок на будь-якому сервері, де є встановлений Docker, та має великий вибір різноманітних бібліотек та плагінів, що дозволяє розробникам швидко додавати до свого застосунку різноманітну функціональність. Для фреймворку також існує добра документація та активна спільнота розробників, що допомагає вирішувати проблеми та швидко отримувати допомогу. Нарешті, Spring Boot Web має високу продуктивність та швидкість виконання завдань, що дозволяє розробляти вебзастосунки, які працюють ефективно та забезпечують високу продуктивність.

3.1.3 Обґрунтування вибору середовища програмної реалізації

Для реалізації програмного застосунку з урахуванням всіх технологій, що потрібно використовувати, у якості середовища розробки було обрано IntelliJ IDEA (рис. 3.1).

IntelliJ IDEA – це інтегроване середовище розробки програмного забезпечення для різних мов програмування, таких як Java, Kotlin, Python, JavaScript та інші. Вона має кілька переваг, що роблять її одним з найпопулярніших інструментів для розробки програмного забезпечення.

Основні переваги IntelliJ IDEA такі:

- підтримка багатьох мов програмування: IntelliJ IDEA підтримує більше десятка мов програмування, включаючи Java, Kotlin, Python, JavaScript, TypeScript, SQL та інші, що дозволяє розробникам працювати з різними мовами у межах одного проєкту;
- функціональність рефакторингу: IntelliJ IDEA надає широкий набір функцій для рефакторингу коду, що дозволяє розробникам швидко та безпечно вносити зміни у свій код;
- автодоповнення коду: IntelliJ IDEA має дуже потужну систему автодоповнення коду, що дозволяє розробникам швидко вводити код та зменшувати кількість помилок;
- підтримка систем контролю версій: IntelliJ IDEA підтримує різні системи контролю версій, такі як Git, SVN та Mercurial, що дозволяє розробникам працювати зі своїм кодом та контролювати його версіонність;
- інтеграція зі засобами розробки: IntelliJ IDEA має інтеграцію з іншими засобами розробки, такими як Maven, Gradle та Ant, що дозволяє розробникам швидко налаштувати свій проєкт;
- підтримка тестування: IntelliJ IDEA має широкий набір функцій для тестування коду, що дозволяє розробникам швидко писати та запускати тести для свого коду;

– вбудована СУБД: IntelliJ IDEA має вбудовану СУБД, яка називається IntelliJ IDEA Database (рис. 3.2). Це легка та швидка вбудована база даних, яка дозволяє зберігати дані без необхідності встановлення додаткового програмного забезпечення. За допомогою IntelliJ IDEA Database, розробники можуть створювати, редагувати, та запитувати бази даних прямо в середовищі розробки без необхідності використання сторонніх програм. Окрім того, IntelliJ IDEA також підтримує підключення до зовнішніх баз даних, таких як MySQL, Oracle, та PostgreSQL.

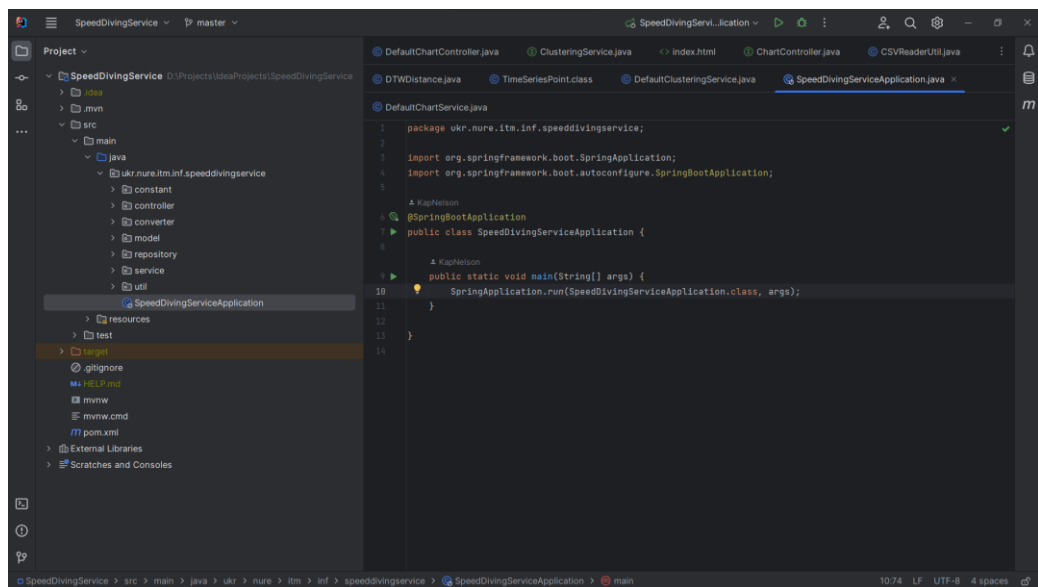


Рисунок 3.1 – Приклад інтерфейсу IntelliJ IDEA

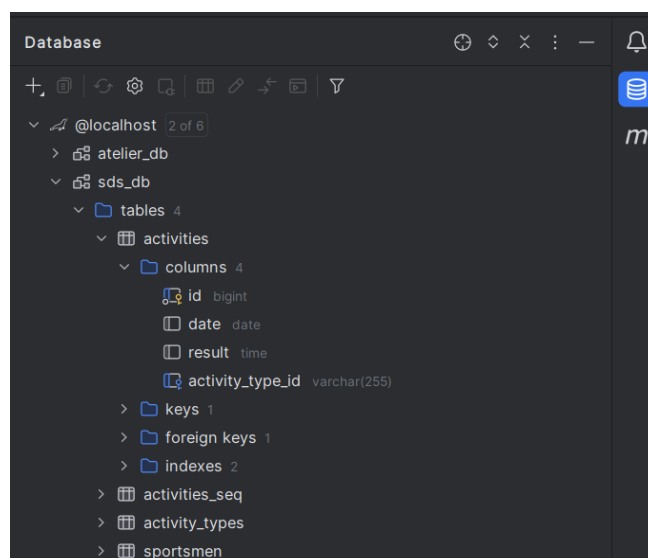


Рисунок 3.2 – Приклад інтерфейсу IntelliJ IDEA Database

3.1.4 Обґрунтування вибору бази даних

Використання баз даних в проєкті має багато переваг. По-перше, вони дозволяють ефективно зберігати та організовувати великі обсяги даних. Це зменшує час на пошук потрібної інформації та покращує продуктивність застосування. База даних забезпечує також можливість одночасної роботи з даними для багатьох користувачів, що зробить проєкт більш ефективним та зручним у використанні. Крім того, ще одним плюсом є те, що вона дозволяє забезпечити високий рівень безпеки даних, так як зазвичай присутній захист у вигляді паролів та прав доступу, що зменшує ризик втрати даних. Також бази даних дають можливість відновити дані у разі їх втрати чи пошкодження. Іншою важливою перевагою є можливість проведення аналітики даних та отримання звітів за допомогою спеціальних запитів до бази даних. Це дозволяє швидко та ефективно аналізувати потрібну інформацію та приймати обґрунтовані рішення. У загальному, використання баз даних дозволяє ефективно зберігати, організовувати та аналізувати великі обсяги даних, зменшує ризик втрати даних та забезпечує високий рівень безпеки даних.

У якості СУБД для застосування було обрано MySQL, тому що вона є одною з найпопулярніших та надійних реляційних систем управління базами даних. Її використання має багато переваг у проєкті, спрямованому на кластеризацію. По-перше, MySQL дозволяє легко взаємодіяти з багатьма серверами баз даних, що є ключовим фактором для проєкту з кластеризацією. Він може бути налаштований для роботи з різними реплікаційними та шардовими стратегіями, що дозволяє збільшити продуктивність та забезпечити високий рівень доступності даних. Крім того, MySQL надає багато інструментів для оптимізації запитів та зменшення навантаження на базу даних, що дозволяє отримувати швидкі та ефективні результати. Її можливості по оптимізації запитів дозволяють створювати складні запити, які вимагають обробки великої кількості даних, збільшуючи продуктивність

та ефективність в процесі роботи з проектом. Також доволі важливо що, MySQL є відкритою системою, що дає можливість розширювати її можливості та функціонал за допомогою різноманітних застосунків та розширень. Це забезпечує гнучкість та можливість налаштування системи під потреби проекту з кластеризацією. У загальному, MySQL є потужною та надійною системою управління базами даних, що забезпечує широкі можливості для кластеризації та оптимізації роботи з даними. Використання MySQL для проекту з кластеризацією дозволяє отримати швидкі та ефективні результати, збільшити продуктивність та забезпечити високий рівень доступності.

Так як важливо забезпечити ефективну роботу з даними, зберегти точність та забезпечити цілісність даних, а також передбачити легке управління та масштабування застосунку, було вирішено використовувати саме реляційну базу даних. Реляційні бази даних є одним з найбільш популярних видів баз даних та мають кілька переваг для застосунків, спрямованих на кластеризацію. Ось основні з них:

- стандартизований формат даних: РБД використовують стандартний формат зберігання даних, що дозволяє легко обмінюватися даними між різними застосунками та системами;
- ефективна робота зі зв'язками між даними: в РБД даних зв'язки між різними таблицями встановлюються за допомогою ключів, що робить роботу зі зв'язками між даними більш ефективною та простішою;
- забезпечення цілісності даних: РБД мають механізми для забезпечення цілісності даних, що дозволяє зберігати та забезпечувати точність даних;
- легкість управління даними: РБД мають декілька інструментів для управління даними, такі як мова SQL, що дозволяє швидко та ефективно виконувати операції з даними, такі як вставка, оновлення та видалення даних;

– масштабованість: РБД дозволяють легко масштабувати застосунок та оброблювати великі обсяги даних, що дозволяє зберігати та аналізувати великі обсяги даних, що потрібні для кластеризації.

3.2 Програмна реалізація

У данному розділі буде описано програмну реалізацію системи кластеризації часових рядів. На першому етапі було створено концептуальну модель даних та виконано проєктування бази даних з використанням MySQL. Наступним етапом було створення backend частини за допомогою Spring Framework, а саме розробка REST API та його інтеграція з базою даних. Frontend частина була реалізована за допомогою JavaScript та бібліотеки Chart.js для візуалізації даних та взаємодії з користувачем. Крім того, було застосовано розподілену архітектуру, яка дозволяє запускати застосунок на різних пристроях та забезпечувати швидку обробку даних. В даному розділі будуть розглянуті деталі реалізації кожної з компонент системи.

3.2.1 Реалізація завантаження даних

Першим етапом програмної реалізації було правильне розбиття, аналіз, адаптація та збереження у реляційну базу даних CSV файлу (рис. 3.3) з даними спортсменів. Для цього було створено сервіс, який зчитував дані з CSV файлу та перевіряв їх на відповідність формату. Далі, отримані дані були збережені у відповідні таблиці бази даних (рис. 3.4), де кожен стовпець таблиці відповідав конкретному параметру спортсмена. Таким чином, було забезпечено зручний та швидкий доступ до даних для подальшої обробки та аналізу.

79	Перший етап Кубку України м. Київ 2011	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	30.01.2011	0:16:50	##	21	3	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:18:20
80	Фінал Кубку України з плавання в ластах м. Київ	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	26.04.2008	0:16:50	##	22	5	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:17
81	Чемпіонат України з плавання в ластах та марафону 01-05.06.2014	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	01.06.2014	0:17:00	##	14	6	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:03
82	Фінал Кубку України з плавання в ластах м. Київ 2009	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	26.04.2009	0:17:20	##	15	6	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:18:24
83	Відкритий Чемпіонат міста Києва серед 4-х вікових категорій із плавання в ластах	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	24.11.2013	0:17:20	##	18	4	B	МС	"АкваЛіде"	"АкваЛіде"	0:17:25
84	Фінал Кубку України з плавання в ластах 13-15.04.2014	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	13.04.2014	0:17:25	##	19	6	A	МС	"АкваЛіде"	"АкваЛіде"	0:17:53
85	Фінал Кубку України з плавання в ластах 22-24.02.2015	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	22.02.2015	0:17:28	##	13	2	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:10
86	Перший етап Кубку України м. Київ 20-23.01.2008	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	22.01.2008	0:17:30	##	10	4	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:41
87	Другий етап Кубку України з плавання в ластах м. Київ 16-18.03.2008	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	16.03.2008	0:17:30	##	18	5	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:13
88	Всукраїнське змагання з плавання в ластах 25-27.01.2015	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	25.01.2015	0:17:30	##	19	5	A	МС	"АкваЛіде"	"АкваЛіде"	0:17:59
89	Чемпіонат України м. Київ 2009	50 м плавання в ластах	Жін.	Бартак	Поліна	1999	08.06.2009	0:17:00	##	0	0	A	МС	"АкваЛіде"	"АкваЛіде"	0:00:00
90	Чемпіонат України з плавання в ластах 25-29.05.2013	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	25.05.2013	0:17:00	##	12	7	A	МС	"АкваЛіде"	"АкваЛіде"	0:17:18
91	Фінал Кубку України з плавання в ластах м. Київ 2009	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	26.04.2009	0:17:00	##	15	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:25
92	Чемпіонат України м. Київ	50 м плавання в ластах	Жін.	Бартак	Поліна	1999	08.06.2008	0:17:00	##	15	6	A	МС	"АкваЛіде"	"АкваЛіде"	0:00:00
93	Другий етап Кубку України з плавання в ластах м. Київ 2009	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	16.03.2009	0:17:00	##	17	3	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:18:04
94	Перший етап Кубку України м. Київ 2009	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	24.01.2009	0:17:00	##	20	6	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:15
95	Чемпіонат України серед юніорів з плавання в ластах 20-23.04.2013	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	21.04.2013	0:17:00	##	12	4	B	МС	"АкваЛіде"	"АкваЛіде"	0:17:41
96	Всукраїнське змагання серед юніорів з плавання в ластах по 3-х вікових групах 19-22.01.2013	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	20.01.2013	0:17:00	##	21	5	B	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:49
97	Фінал Кубку України з плавання в ластах м. Київ	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	26.04.2008	0:17:00	##	22	6	B	МС	"АкваЛіде"	"АкваЛіде"	0:18:11
98	Міжнародні змагання з плавання в ластах м. Харків 18-19.12.2014	50 м плавання в ластах	Жін.	Іщук	Ніна	2004	18.12.2014	0:17:07	##	10	5	A	МС	Пер. уніс Пер. уніс	Пер. уніс	0:18:51
99	Чемпіонат України м. Київ 2009	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	08.06.2009	0:17:10	##	0	0	A	МС	"АкваЛіде"	"АкваЛіде"	0:00:00
100	Чемпіонат України м. Київ	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	08.06.2008	0:17:10	##	15	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:00:00
101	Фінал Кубку України 17-19.02.2013	50 м плавання в ластах	Жін.	Коледна	Ангеліна	1998	17.02.2013	0:17:10	##	12	4	B	МСМК	"АкваЛіде"	"АкваЛіде"	0:18:01
102	Другий етап Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	21.03.2010	0:17:20	##	16	7	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:18:13
103	Перший етап Кубку України м. Київ 2010	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	24.01.2010	0:17:20	##	20	6	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:45
104	Другий етап Кубку України з плавання в ластах м. Київ 2011	50 м плавання в ластах	Жін.	Ельцова	Ганна	2004	03.04.2011	0:17:25	##	17	7	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:47
105	Чемпіонат України м. Київ 2010	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	09.06.2010	0:17:30	##	14	7	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:16
106	Чемпіонат України м. Київ 2010	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	09.06.2010	0:17:50	##	13	3	A	МС	"АкваЛіде"	"АкваЛіде"	99-99-99
107	Першість м. Києва серед 4 вікових груп	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	12.12.2009	0:17:50	##	15	3	A	МС	"АкваЛіде"	"АкваЛіде"	99-99-99
108	Першість м. Києва серед 4 вікових груп	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	12.12.2009	0:17:50	##	15	5	A	МСМК	"АкваЛіде"	"АкваЛіде"	99-99-99
109	Другий етап Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	21.03.2010	0:17:50	##	16	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:40
110	Фінал Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	25.04.2010	0:17:50	##	17	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:08
111	Фінал Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Бернадська	Олександра	1993	25.04.2010	0:17:50	##	17	2	A	МСМК	"АкваЛіде"	"АкваЛіде"	0:17:23
112	Другий етап Кубку України з плавання в ластах м. Київ 2011	50 м плавання в ластах	Жін.	Бойкова	Катерина	1995	03.04.2011	0:17:50	##	17	8	A	МСМК	Летіон - Д Летіон - Д	Летіон - Д	0:18:28
113	Перший етап Кубку України м. Київ 2010	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	24.01.2010	0:17:50	##	20	2	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:04
114	Перший етап Кубку України м. Київ 2011	50 м плавання в ластах	Жін.	Бойкова	Катерина	1995	30.01.2011	0:17:50	##	20	3	A	МСМК	Летіон - Д Летіон - Д	Летіон - Д	0:17:52
115	Чемпіонат Харківської області з плавання в ластах 27-28.03.2014	50 м плавання в ластах	Жін.	Іщук	Ніна	2004	27.03.2014	0:17:59	##	10	5	A	МС	Пер. уніс Пер. уніс	Пер. уніс	0:18:20
116	Перший етап Кубку України м. Київ 2009	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	24.01.2009	0:18:10	##	20	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:57
117	Фінал Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Бойкова	Дар'я	1993	25.04.2010	0:18:10	##	16	5	B	МС	Летіон - Д Летіон - Д	Летіон - Д	0:19:15
118	Фінал Кубку України з плавання в ластах м. Київ 2009	50 м плавання в ластах	Жін.	Бартак	Поліна	1999	26.04.2009	0:18:20	##	14	3	A	МС	"АкваЛіде"	"АкваЛіде"	88-88-88
119	Фінал Кубку України з плавання в ластах м. Київ 2011	50 м плавання в ластах	Жін.	Біндер	Катерина	1995	01.05.2011	0:18:20	##	17	1	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:37
120	Фінал Кубку України з плавання в ластах м. Київ 2011	50 м плавання в ластах	Жін.	Бойкова	Катерина	1995	01.05.2011	0:18:20	##	17	8	A	МСМК	Летіон - Д Летіон - Д	Летіон - Д	88-88-88
121	Фінал Кубку України з плавання в ластах м. Київ 2010	50 м плавання в ластах	Жін.	Біндер	Катерина	1995	25.04.2010	0:18:20	##	17	8	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:18
122	Другий етап Кубку України з плавання в ластах м. Київ 2009	50 м плавання в ластах	Жін.	Бартик	Богдана	1994	16.03.2009	0:18:20	##	17	8	A	МС	"АкваЛіде"	"АкваЛіде"	0:18:55
123	Перший етап Кубку України м. Київ 2011	50 м плавання в ластах	Жін.	Бойкова	Дар'я	1993	30.01.2011	0:18:20	##	20	6	A	МС	Летіон - Д Летіон - Д	Летіон - Д	0:18:52

Рисунок 3.3 – Приклад формату вхідного CSV файлу з даними

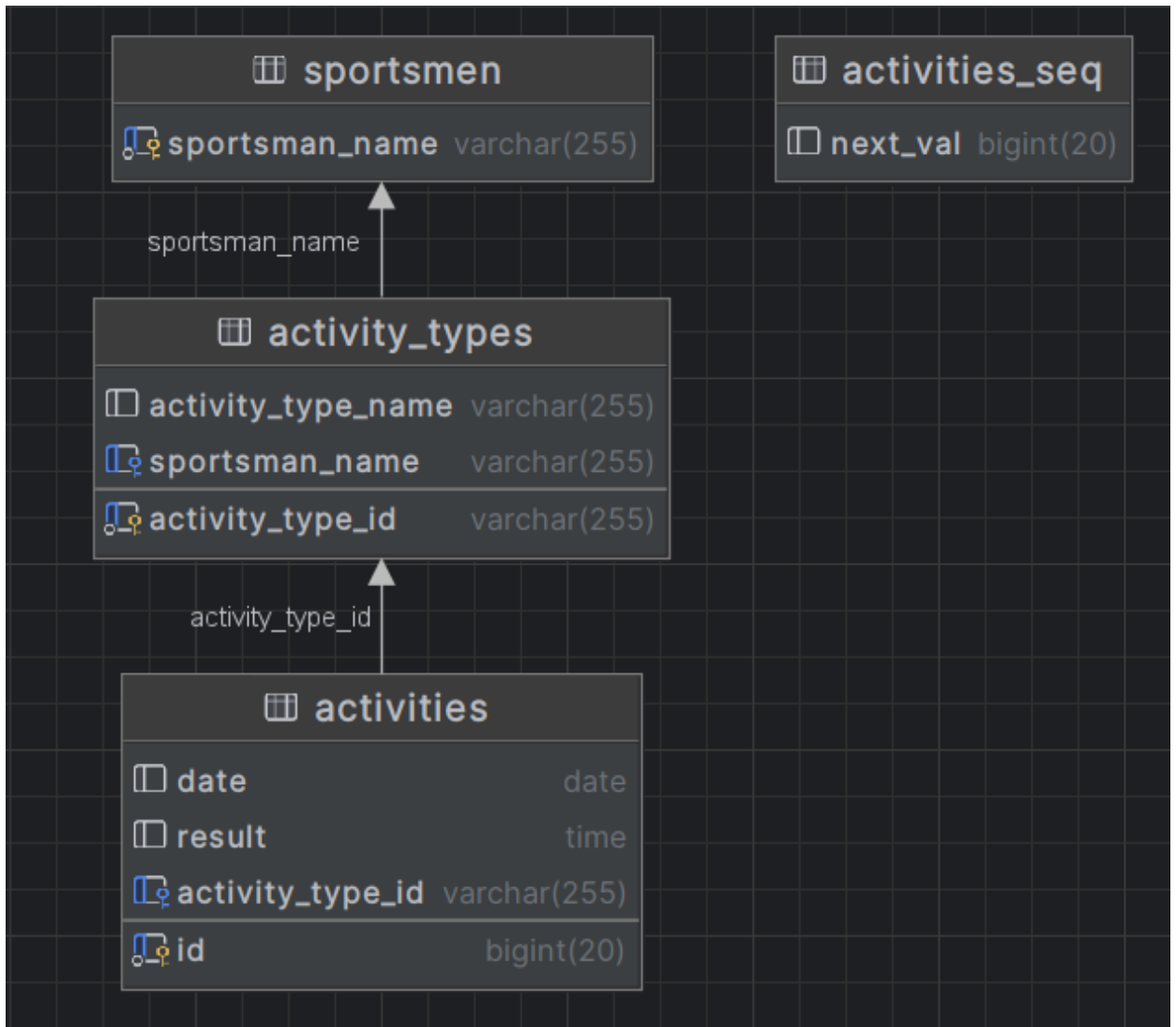


Рисунок 3.4 – Структура бази даних

3.2.2 Реалізація графічного інтерфейсу

Другим етапом програмної реалізації було створення графічного інтерфейсу (рис. 3.5) користувача для зручної взаємодії з застосунком. Графічний інтерфейс було розроблено на основі сучасних технологій, таких як HTML, CSS та JavaScript, щоб забезпечити зручний та інтуїтивно зрозумілий доступ до функціоналу застосунку. За допомогою цього користувач може зручно вибирати параметри для аналізу даних та переглядати результати кластеризації у вигляді графіків.

На сторінці є заголовок сайту та основний блок, в якому містяться розділи з випадаючим списком у якому знаходяться форми, які дозволяють завантажувати CSV файли з даними про досягнення спортсменів, будувати графіки для окремих спортсменів чи дисциплін та виконувати кластеризацію даних. Кожен розділ містить форму з відповідними полями для введення параметрів, що передаються на сервер для обробки.

The screenshot shows a web application titled "Кластеризація часових рядів" (Time Series Clustering). It features a sidebar with four menu items: "Про сайт" (About the site), "Завантаження даних" (Data loading), "Побудова графіків" (Building graphs), and "Кластеризація" (Clustering). The "Кластеризація" section is active and contains a form with the following elements: a text input for "Дисципліна" (Discipline) with the value "100 м плавання в біластах"; a dropdown menu for "Метод кластеризації" (Clustering method) set to "K-Means"; two checkboxes for "Нормалізувати" (Normalize) and "DTW"; and a large black button labeled "Кластеризувати" (Cluster).

Рисунок 3.5 – Графічний інтерфейс застосунку

У даному проєкті всі обчислення та кластеризація були виконані на стороні Java серверу, що дозволило зменшити навантаження на клієнтський

браузер та забезпечити швидке відображення результатів. Однак, для відображення графіків та взаємодії з користувачем використовувався JavaScript, що забезпечило зручну та динамічну візуалізацію результатів. Поєднання функціональності Java та JavaScript дало можливість забезпечити ефективну та зручну роботу з проектом.

3.2.3 Реалізація серверної архітектури

Застосунок для кластеризації часових рядів був написаний з дотриманням архітектурних принципів, що дозволило зберегти його структурованість та зручність для розширення. Всі обчислення та операції з базою даних були реалізовані на стороні Java серверу, що дозволило досягнути максимальної продуктивності. Логіка проекту розбита на окремі класи (рис. 3.6), такі як контролери, сервіси, моделі, репозиторії та інші, що спрощує його подальше розширення та підтримку. Також в проекті присутні інтерфейси та класи, які їх імplementують, що дозволяє легко замінювати реалізацію окремих частин без зміни всього застосунку в цілому. Загалом, дотримання архітектурних принципів дозволило зберегти в проекті чистоту коду та зробити його більш підтримуваним та розширюваним у майбутньому.

Один з плюсів такого підходу полягає у тому, що він дозволяє забезпечити високу модульність проекту та розподілити відповідальність між окремими компонентами. Це дозволяє забезпечити більш просте тестування і підтримку коду, оскільки кожен компонент може бути розроблений та перевірений окремо, і вони можуть бути замінені або модифіковані без впливу на решту системи. Крім того, такий підхід сприяє більшій стійкості до змін та розширенню проекту, оскільки компоненти можуть бути додані або замінені без необхідності переписувати весь код. В результаті, застосунок може бути більш гнучким та більш легко розвиватися в

майбутньому. Також, такий підхід дозволяє підтримувати високу якість коду та покращити читабельність, оскільки логіка проєкту розбита на окремі компоненти зі своїм визначеним призначенням. Це полегшує розуміння та редагування коду розробниками.

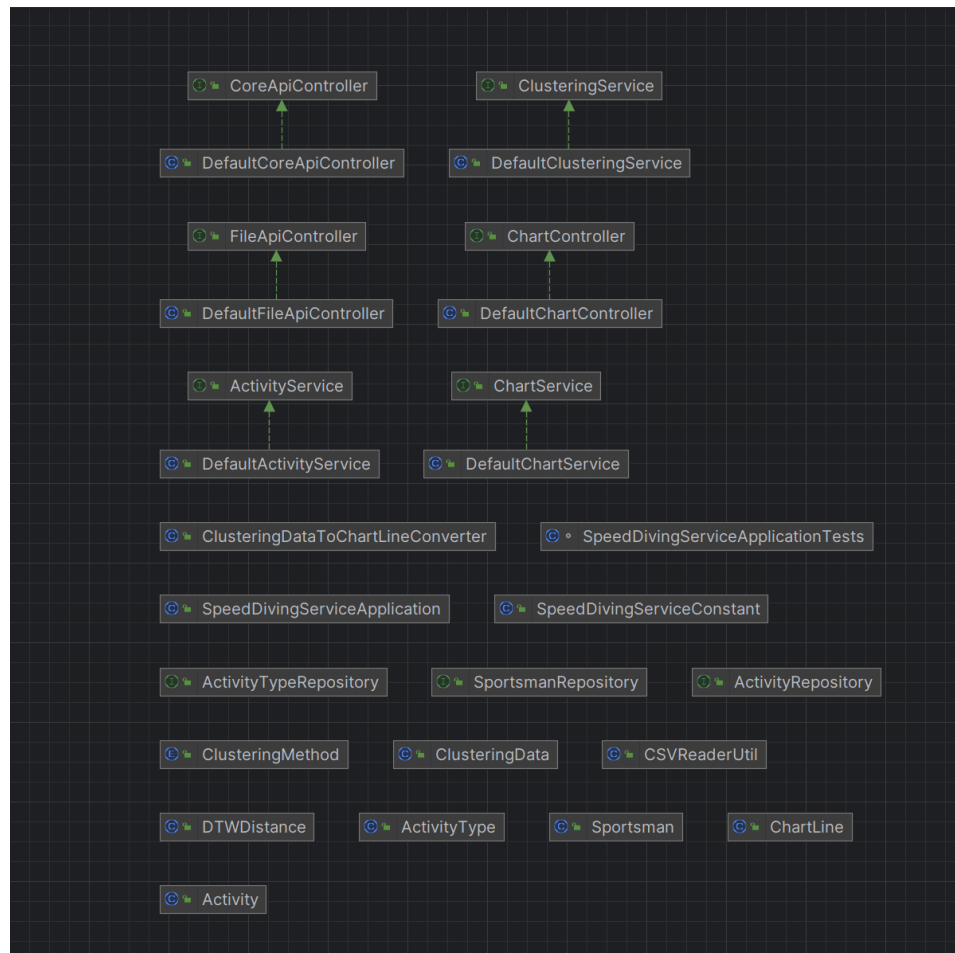


Рисунок 3.6 – Діаграма класів

3.3 Інструкція користувача

На даний момент скористатись сервісом можна запустивши Spring застосунок за допомогою JAR файлу, який було створено, з консолі. Для створення JAR файлу було використано Maven, який дозволяє автоматизувати процес збірки та управління залежностями проєкту. Maven генерує JAR файл, який містить всі залежності та класи проєкту, що необхідні для його роботи. Це робить процес розгортання застосунку значно

простішим, оскільки не потрібно вручну копіювати всі файли та налаштовувати середовище. Всі необхідні бібліотеки та залежності додаються автоматично, що зменшує ризик помилок та полегшує розгортання застосунку. Цей підхід може бути корисним для розробників, які хочуть перевірити роботу застосунку на локальному комп'ютері до його розгортання на сервері або хостингу. Дотримуючись інструкції, можливо доволі легко запустити застосунок та перевірити його роботу на своєму комп'ютері.

Інструкція користувача для запуску Spring застосунку з консолі за допомогою JAR файлу включає наступні кроки:

Крок 1. Переконатися, що на комп'ютері встановлений JRE, якщо його немає, необхідно завантажити та встановити його з офіційного сайту Java.

Крок 2. Завантажити JAR файл застосунку на комп'ютер і зберегти його в потрібному місці.

Крок 3. Відкрити командний рядок (консоль) операційної системи.

Крок 4. Далі перейти в ту директорію, де збережено JAR файл застосунку за допомогою команди `cd`, наприклад: `cd C:\myproject\` (рис. 3.7).

Крок 5. Ввести команду для запуску застосунку: `java -jar <назва_файлу>.jar`, де `<назва_файлу>` – назва JAR файлу застосунку.

Крок 6. Натиснути клавішу Enter, щоб запустити застосунок.

Крок 7. Після успішного запуску застосунку з'явиться повідомлення про запуск сервера, а також URL адреса застосунку. Відкривши браузер та перейшовши за адресою необхідно перевірити роботу застосунку (рис. 3.8).

Важливо, що для успішного старту застосунку повинна бути присутня та налаштована база даних. Перед запуском застосунку необхідно переконатися, що база даних встановлена та працює на комп'ютері або сервері, а також що в конфігураційному файлі проєкту вказані правильні налаштування для з'єднання з базою даних. Якщо база даних не запущена, застосунок не зможе працювати з даними та будуть виникати помилки. Для

Крок 4. Після того, як застосунок буде зупинено, консоль повинна повернути контроль. За потреби можна закрити консоль.

Важливо, що зупинення застосунку може зайняти деякий час, залежно від його обробки та поточної завантаженості.

```
2023-05-08T15:25:42.958+03:00 INFO 18708 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for
persistence unit 'default'
2023-05-08T15:25:42.960+03:00 INFO 18708 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2023-05-08T15:25:42.964+03:00 INFO 18708 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
D:\Projects\IdeaProjects\SpeedDivingService\target>
```

Рисунок 3.9 – Зупинка застосунку

У майбутньому, для звичайного користувача запуск застосунку з консолі може бути не потрібним, оскільки можна буде розмістити сервіс на хостингу та звертатися до нього за допомогою URL адреси в інтернеті. Таким чином, користувачі зможуть легко отримувати доступ до функціональності застосунку, не займаючись установкою та налаштуванням його на своїх комп'ютерах. Це зробить використання застосунку більш зручним та доступним для користувачів з різних місць та пристроїв.

3.4 Тестування розробленої моделі

В даному пункті буде розглянуто тестування розробленої моделі кластеризації часових рядів. Метою цього тестування є перевірка роботи програми та порівняння результатів різних методів кластеризації. Крім того, буде проаналізовано вплив використання нормалізації та DTW на якість кластеризації часових рядів. В цьому розділі будуть наведені графіки, що ілюструють результати кластеризації, а також будуть проаналізовані отримані результати. Також буде розглянуто вплив зведення рядів до фіксованої довжини на якість кластеризації. Аналіз цього аспекту дозволить з'ясувати, чи є зведення рядів до фіксованої довжини корисним і ефективним підходом у контексті кластеризації часових рядів.

У даному випадку для кластеризації було обрано саме 8 кластерів. Однією з причин такого вибору є відповідність цього числа ранговій системі спортивних звань, таких як КМС, МС і тд.

Для тестування розробленої моделі було обрано конкретну дисципліну та дистанцію, щоб забезпечити більш точну та чітку порівняльну аналітику між різними методами. Відбір декількох дисциплін або дистанцій може призвести до нерівних умов порівняння, які можуть вплинути на результати.

На графіках, що представлені в цьому розділі, кожен часовий ряд позначений випадковим окремим кольором, який відповідає кластеру, до якого він був призначений. Крім того, червоним кольором позначено часовий ряд, який є центроїдом кластеру.

3.4.1 Порівняння роботи різних методів кластеризації

Якщо порівнювати графіки *k-means* та *s-means* без нормалізації та DTW, то можна зазначити, що графіки *k-means* (рис. 3.10) та *s-means* (рис. 3.11) без нормалізації виглядають досить схоже. Обидва методи кластеризації розділили дані на 8 кластерів, проте можна помітити деякі відмінності в розташуванні окремих точок у кожному кластері. Також, графік *k-means* має дещо більш рівномірний розподіл точок у кластерах порівняно з графіком *s-means*, де деякі кластери мають більш випадкове розподілення точок [32, 33].

Можна явно побачити, як зведення рядів до фіксованої довжини вплинуло на вид графіків. Після зведення рядів до фіксованої довжини, кластери стали більш чітко розділені, з меншою кількістю випадкових значень. Зокрема, на графіку *k-means* можна побачити, що кожен кластер має чітко виражені границі та більш концентрований розподіл точок [34]. Графік *s-means* також відображає краще розділення кластерів, з меншою кількістю точок, що випадково потрапили до неправильного кластеру. Таким чином,

зведення рядів до фіксованої довжини вплинуло на поліпшення результатів кластеризації та зробило графіки більш зрозумілими і чіткими.

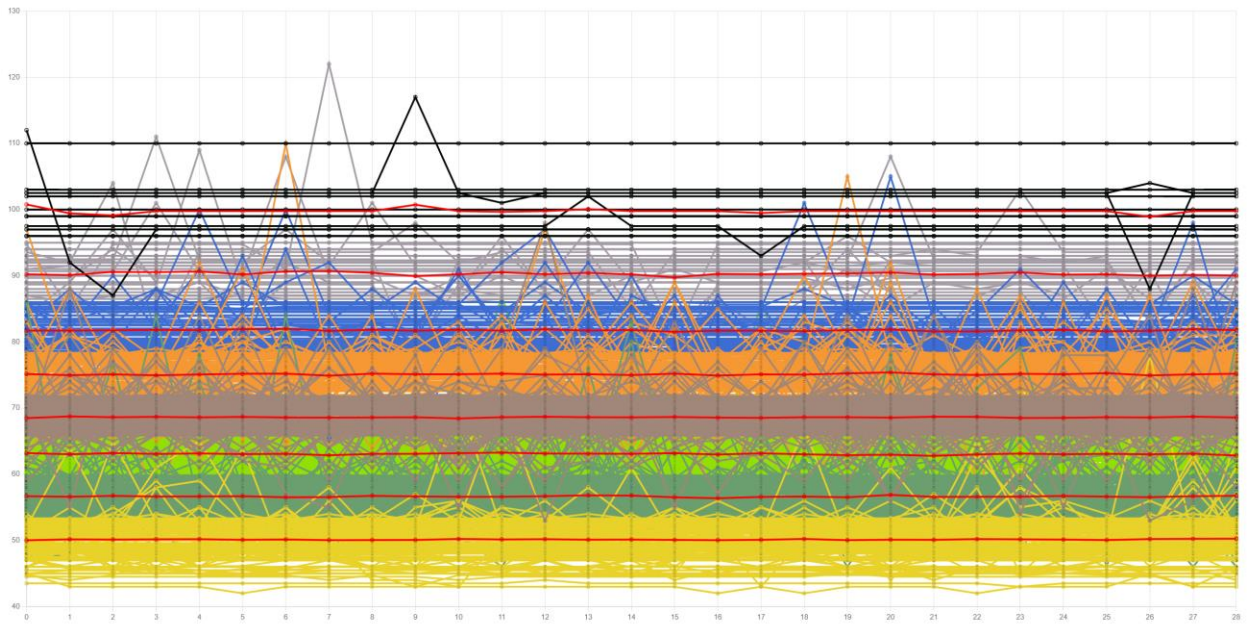


Рисунок 3.10 – Графічне представлення кластеризації методом k -means без використання нормалізації та DTW

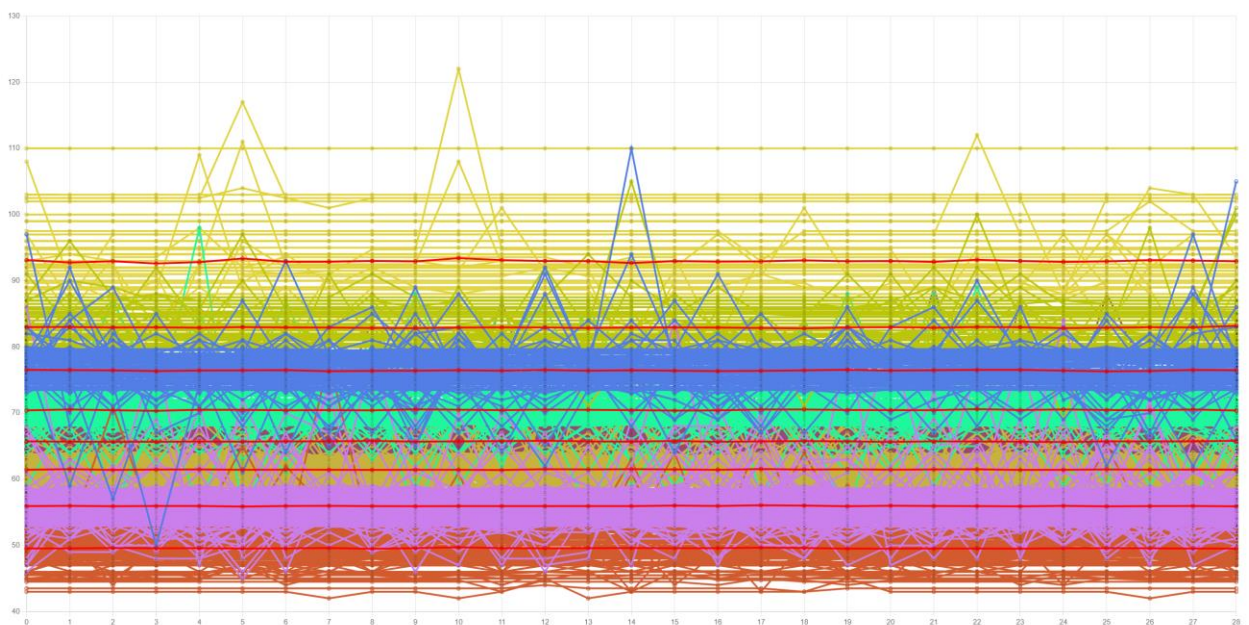


Рисунок 3.11 – Графічне представлення кластеризації методом c -means без використання нормалізації та DTW

Проте, зведення рядів до фіксованої довжини може мати деякі негативні аспекти. Одним з них є втрата деталей часових рядів. Коли довжина рядів зменшується, це може призвести до втрати важливих даних, які були враховані в оригінальному ряді. Це може спричинити помилкові результати при кластеризації та іншому аналізі даних. Крім того, зведення рядів до фіксованої довжини може привести до порушення властивостей часових рядів, таких як стаціонарність, автокореляція тощо. Це може ускладнити аналіз та кластеризацію даних. Також важливо враховувати, що зведення рядів до фіксованої довжини може бути неефективним у випадку, коли вхідні дані мають дуже високу варіативність. У такому випадку зведення до фіксованої довжини може призвести до втрати важливих даних та зменшення якості кластеризації. Але не зважаючи на це, у рамках даної задачі, зведення рядів до фіксованої довжини є необхідним кроком для застосування стандартних методів кластеризації, таких як *k-means* та *c-means*. Це дозволяє отримати результати кластеризації, які можна порівняти з результатами, отриманими в інших дослідженнях. Хоча зведення рядів до фіксованої довжини може мати негативний вплив на точність кластеризації, він є прийнятним компромісом між точністю та можливістю застосування стандартних методів.

Також на обох графіках можна помітити кілька аномальних значень, які відхиляються від загальної закономірності розподілу точок у кластерах. Це може свідчити про наявність відхилень у деяких часових рядах, які потребують додаткового аналізу та уваги при подальшій обробці даних.

3.4.2 Аналіз впливу нормалізації на кластеризацію

Після додавання нормалізації до методів *k-means* та *c-means* можна помітити значні зміни в графіках кластеризації. На графіку *k-means* (рис. 3.12) можна помітити більш однорідний розподіл точок у кластерах, а

також меншу кількість аномальних значень. Графік *c-means* (рис. 3.13) має аналогічні покращення. Загалом, можна стверджувати, що додавання нормалізації покращило результати кластеризації в обох методах.

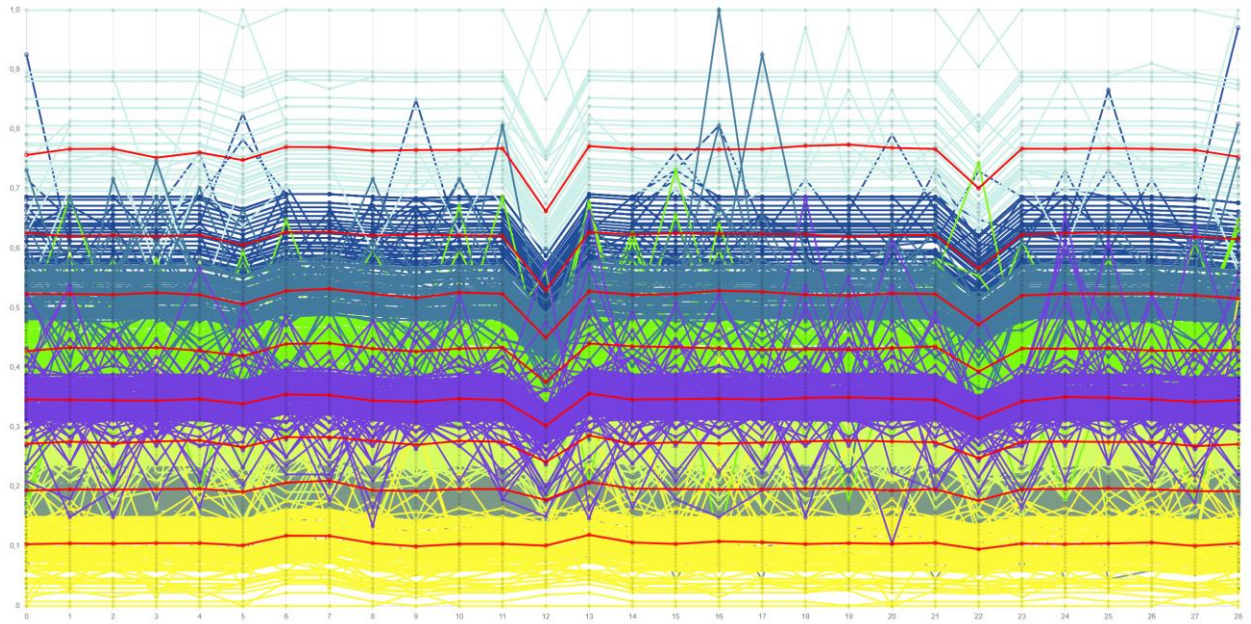


Рисунок 3.12 – Графічне представлення кластеризації методом *k-means* з нормалізацією

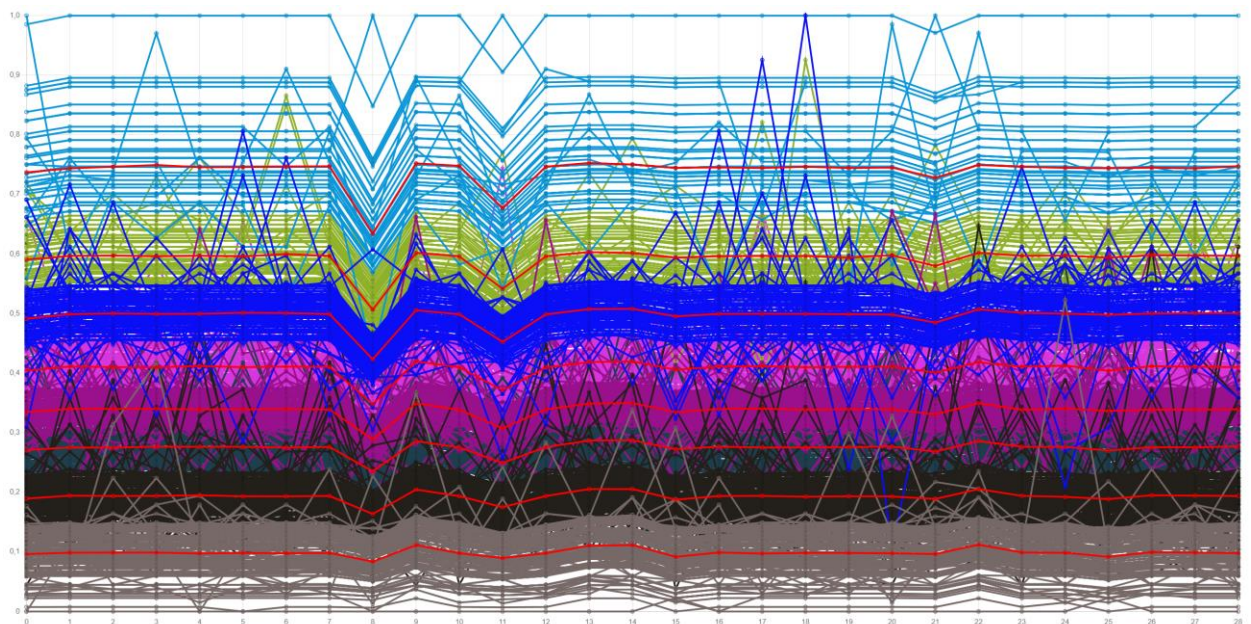


Рисунок 3.13 – Графічне представлення кластеризації методом *c-means* з нормалізацією

Після додавання нормалізації при деяких значеннях по осі X , що уявляє собою конкретний результат, бачимо, що значення на осі Y , що уявляє собою значення у часі, за який спортсмен проплив, є меншими. Це може бути пов'язано з тим, що при нормалізації дані можуть бути перетворені так, що втрачається частинка інформації. Наприклад, якщо відносні значення у часі пропливу спортсменів дуже близькі одне до одного, то після нормалізації вони можуть бути вирівняні і втратити свою відмінність.

Також, виникнення таких зсувів на графіку, може свідчити про те, що у цих моментах спортсмени показали гірші результати порівняно зі своїм середнім рівнем досягнень. Наприклад, можливо, у цей період вони перебували в стадії відновлення після тренувань або мали певні проблеми зі здоров'ям, що вплинули на їхню виступність. Також можливо, що у ці моменти спортсмени зустрілися зі змагальними складними обставинами, такими як погані умови для плавання або технічні проблеми з обладнанням. В цьому випадку, менші значення на осі Y можуть вказувати на те, що спортсмени не могли показати свій повний потенціал у цих моментах.

3.4.3 Аналіз впливу DTW на кластеризацію

Після використання DTW замість евклідової відстані в алгоритмах k -means та c -means, графіки кластеризації значно змінилися. Це пов'язано з тим, що DTW є метрикою, яка враховує не тільки значення на кожен момент часу, але й форму графіка. Це означає, що графіки можуть бути близькими за DTW, навіть якщо значення на окремих моментах часу досить різні. Як результат, кластеризація з використанням DTW виглядає більш розумною, оскільки вона кластеризує графіки, які мають схожі форми.

Однак, це також може призвести до того, що кластеризація може бути менш точною, якщо значення на окремих моментах часу досить відрізняються в межах кластера. Що і помітно на рисунках 3.14 та 3.15.

Кластери стали менш чіткими, помітна наявність вузьких смуг різних кольорів всередині одного кластера, що вказує на те, що деякі частини часового ряду мають значення, які відрізняються від значень інших частин.

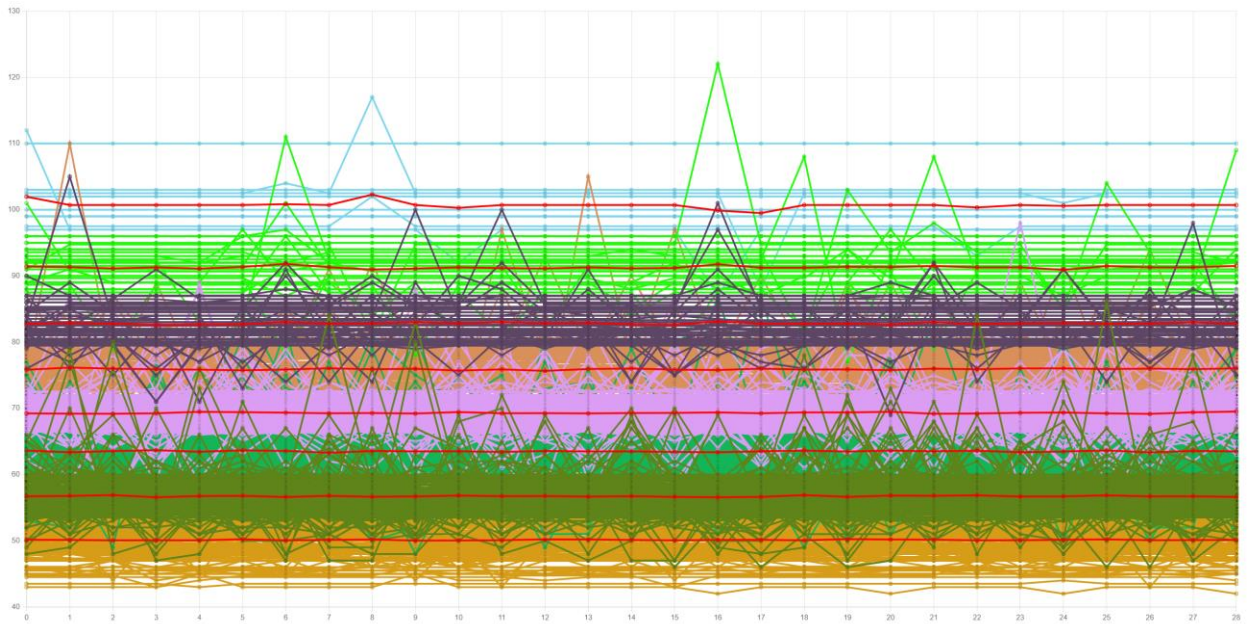


Рисунок 3.14 – Графічне представлення кластеризації методом *k-means* з використанням DTW

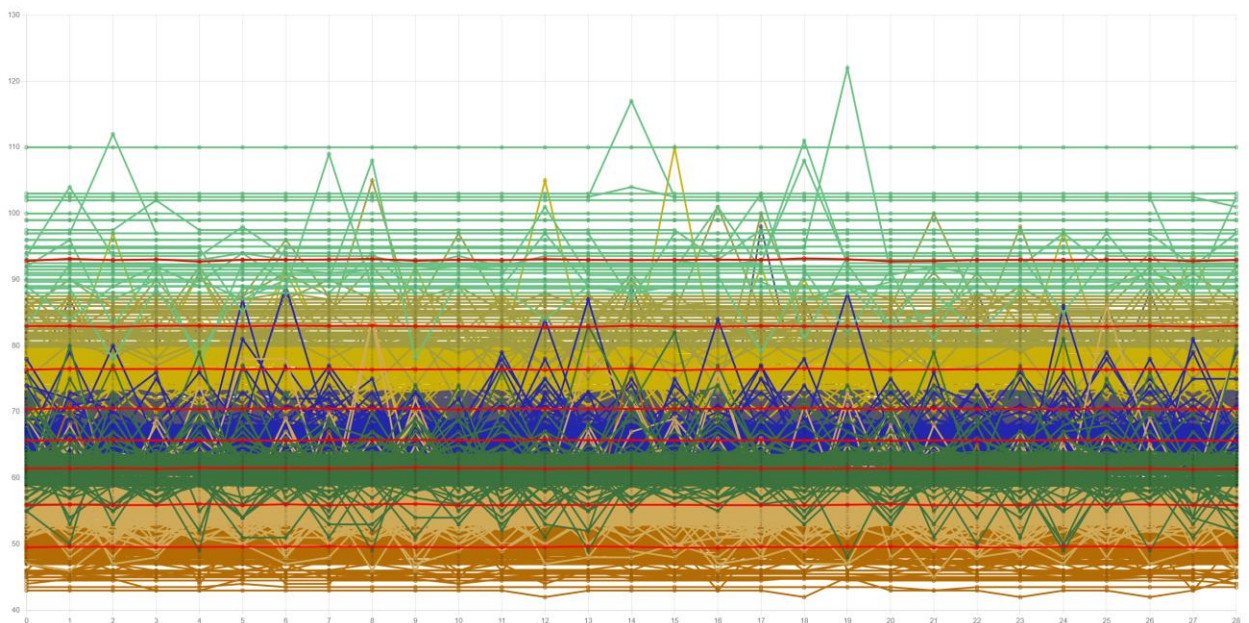


Рисунок 3.15 – Графічне представлення кластеризації методом *c-means* з використанням DTW

3.4.4 Аналіз впливу комбінації DTW та нормалізації

Застосування DTW та нормалізації разом, призвело до більш явного групування точок в кластерах на графіках кластеризації, як видно з рисунків 3.16 та 3.17. Кожен кластер має більш однорідні значення відносно відстані та часу. Це призводить до покращення точності кластеризації та зростання якості результуючих кластерів. Також, застосування DTW та нормалізації разом допомогло уникнути проблеми з поганою кластеризацією на окремих ділянках часу, які можуть відрізнятися в межах одного кластера.

Тобто, використання DTW для знаходження відстані між кластерами дозволило зменшити вплив відхилень від нормального розподілу на кінцеві результати кластеризації. Нормалізація ж, у свою чергу, збільшила чутливість кластеризації до змін в даних, тобто забезпечила більш точну класифікацію, що відображає дійсну структуру даних. Разом вони допомогли покращити результати кластеризації, зменшивши вплив випадкових відхилень та забезпечивши більш точну та надійну класифікацію.

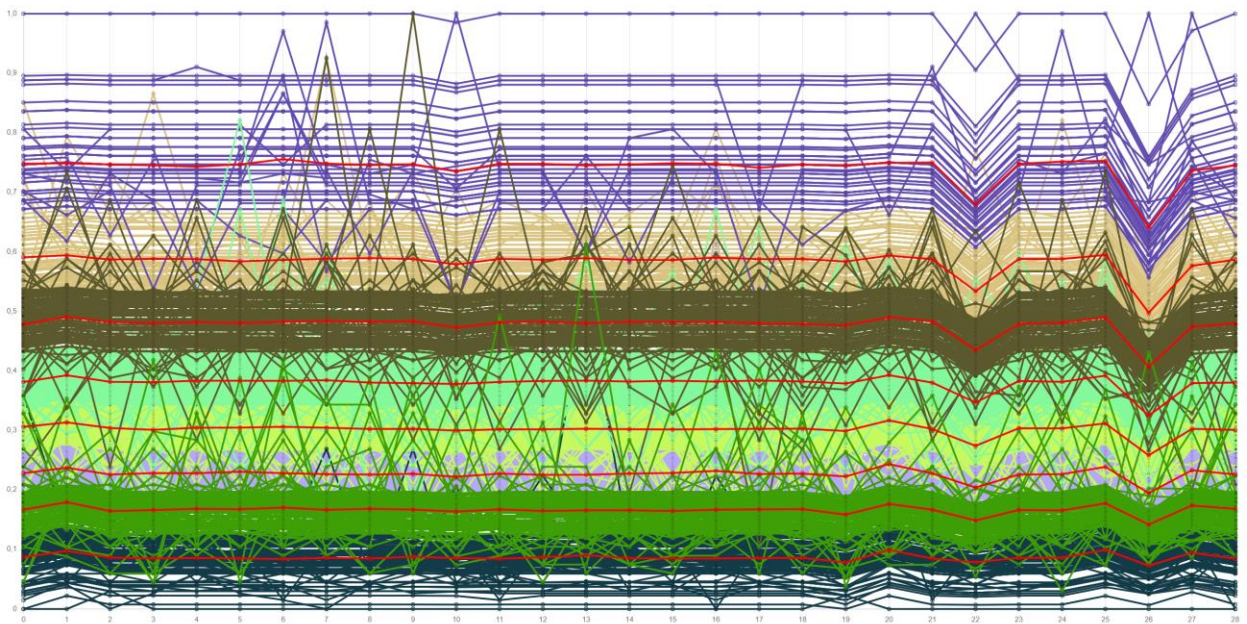


Рисунок 3.16 – Графічне представлення кластеризації методом *k*-means з використанням нормалізації та DTW

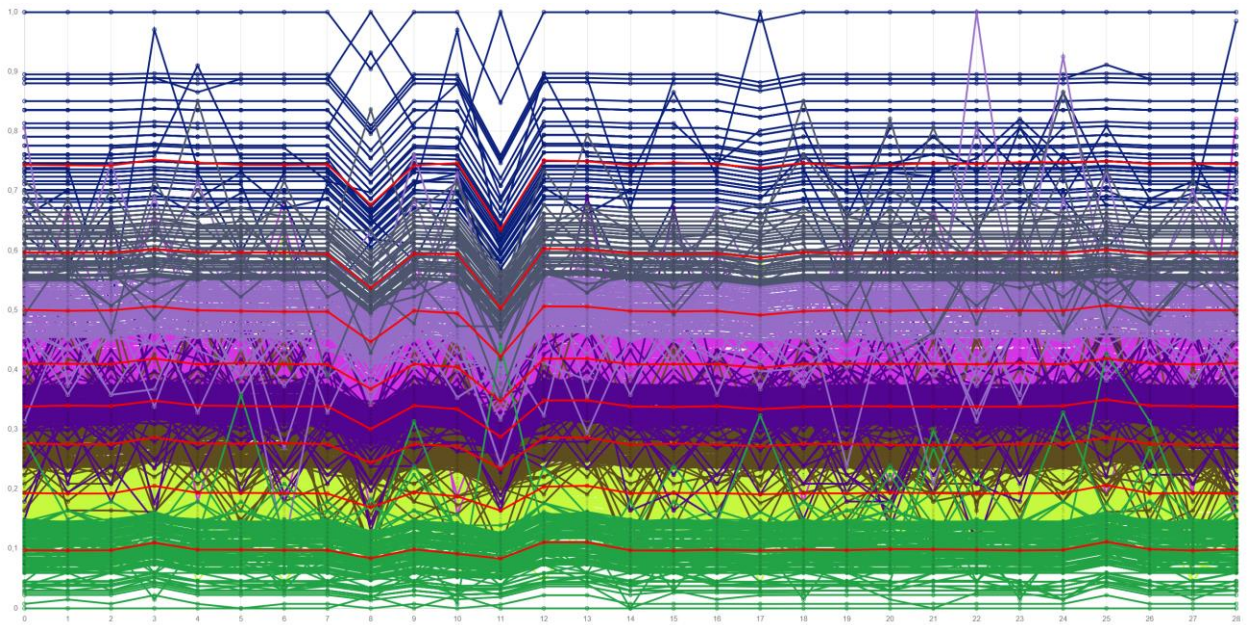


Рисунок 3.17 – Графічне представлення кластеризації методом *c-means* з використанням нормалізації та DTW

3.4.5 Тестування завантаження даних

У цьому розділі проведено тестування завантаження даних з файлу CSV у систему. Почавши імпорт CSV файлу правильного формату до системи у консолі з'являється повідомлення про успішний старт та завершення завантаження даних (рис. 3.18). Після цього можна перевірити результат у таблицях БД, де відображаються збережені дані з файлу (рис. 3.19). Можливість завантаження та подальшого збереження, підвищує ефективність та точність роботи системи, оскільки можна додавати нові дані у систему та використовувати їх для покращення аналітики та доповнення існуючих результатів.

```
2023-05-14T16:07:15.673+03:00 INFO 17516 --- [nio-8060-exec-3] u.n.i.i.s.s.impl.DefaultActivityService : ~~~~~Parsing has started~~~~~
2023-05-14T16:07:15.936+03:00 INFO 17516 --- [nio-8060-exec-3] u.n.i.i.s.s.impl.DefaultActivityService : ~~~~~Parsing completed successfully~~~~~
```

Рисунок 3.18 – Повідомлення про старт за завершення імпорту

	id	date	result	activity_type_id
7694	7694	2013-05-25	00:00:21	50 м плавання в ластвах_Олексій Зімін
7695	7695	2013-05-25	00:00:28	50 м плавання в ластвах_Іван Соловей
7696	7696	2013-05-25	00:00:24	50 м плавання в ластвах_Юрій Марков
7697	7697	2013-05-25	00:00:25	50 м плавання в ластвах_Андрій Матяш
7698	7698	2019-05-15	00:03:15	50 м плавання в ластвах_Едуард Ляшук
7699	7699	2016-04-09	00:01:02	100 м плавання в біластвах_Анастасія Мовчан
7700	7700	2016-04-09	00:00:50	100 м плавання в біластвах_Анастасія Левченко
7701	7701	2016-04-09	00:00:54	100 м плавання в біластвах_Катерина Склярова
7702	7702	2016-04-09	00:00:51	100 м плавання в біластвах_Ксенія Семикіна
7703	7703	2016-04-09	00:00:55	100 м плавання в біластвах_Анна Позій
7704	7704	2016-04-09	00:01:07	100 м плавання в біластвах_Олександра Мостовенко
7705	7705	2016-04-09	00:01:00	100 м плавання в біластвах_Юлія Черних
7706	7706	2016-04-09	00:00:59	100 м плавання в біластвах_Вероніка Романцова
7707	7707	2016-04-09	00:01:09	100 м плавання в біластвах_Аліна Черних
7708	7708	2016-04-09	00:00:55	100 м плавання в біластвах_Сніжана Строкіна
7709	7709	2016-04-09	00:00:51	100 м плавання в біластвах_Поліна Ружицька
7710	7710	2016-04-09	00:00:48	100 м плавання в біластвах_Світлана Жукова
7711	7711	2016-04-09	00:01:02	100 м плавання в біластвах_Марія Руденко
7712	7712	2016-04-09	00:00:58	100 м плавання в біластвах_Юлія Савонюк
7713	7713	2016-04-09	00:00:54	100 м плавання в біластвах_Тетяна Гудкова
7714	7714	2016-04-09	00:01:02	100 м плавання в біластвах_Аліса Соломіна

Рисунок 3.19 – Збережені дані у БД

3.4.6 Тестування візуалізації даних

Для поліпшення взаємодії з даними було реалізовано їх графічну візуалізацію. На графіку відображені часові ряди досягнень всіх спортсменів у конкретній дисципліні (рис. 3.20). При наведенні курсора на точку на графіку, можна побачити інформацію про спортсмена, який у цю дату показав певні результати (рис. 3.21). Це дозволяє з легкістю зрозуміти і проаналізувати залежності та тенденції результатів спортсменів від часу.

Кожен часовий ряд спортсмену на графіку відображається окремим кольором, що дозволяє легко відрізнити досягнення одного спортсмена від досягнень інших. Якщо між двома точками часового ряду відсутнє з'єднання, то це означає, що за цей період спортсмен не має результатів. Це важлива інформація для аналізу даних, оскільки дозволяє враховувати перерви у тренуванні або можливі травми, які впливають на досягнення спортсмена.

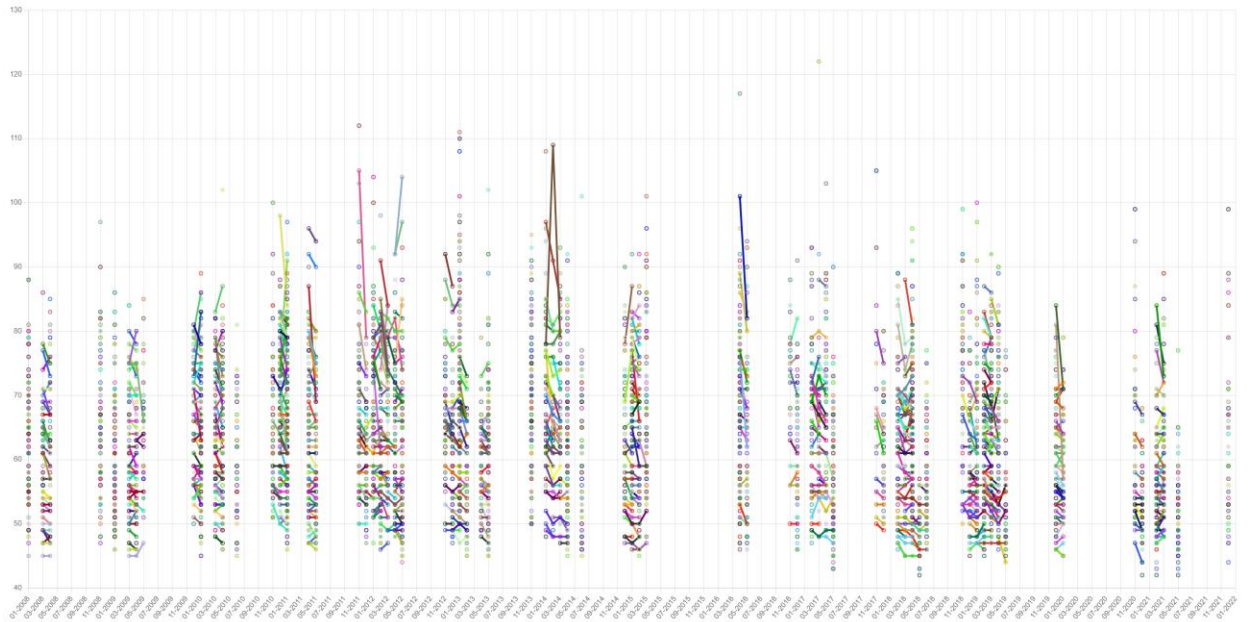


Рисунок 3.20 – Графічна візуалізація даних

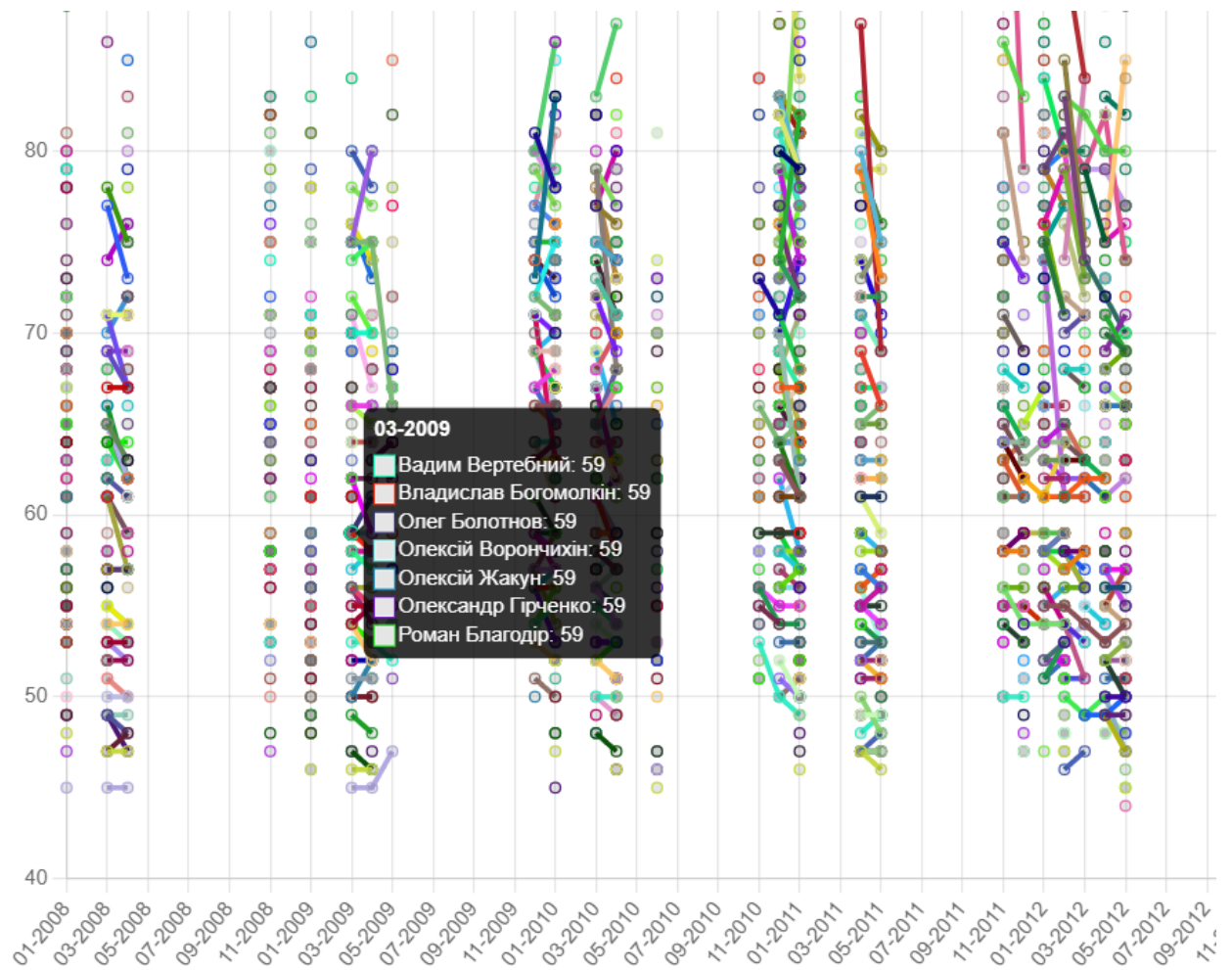


Рисунок 3.21 – Відображення детальної інформації

ВИСНОВКИ

У рамках кваліфікаційної роботи було проведено аналіз важливої проблеми кластеризації часових рядів у контексті оцінки результатів спортсменів підводного плавання. Результатом роботи було розроблення ефективної та точної моделі кластеризації, яка забезпечує групування спортсменів зі схожими результатами.

У процесі роботи були проаналізовані різні алгоритми кластеризації для часових рядів, зокрема був використаний алгоритм DTW. Використання DTW дозволило ефективно врахувати схожість між часовими рядами, уникнути проблеми зі зсувом та масштабуванням даних. Для забезпечення точності кластеризації була використана нормалізація даних, що дозволяє зрівняти значення на різних шкалах.

У ході роботи були проаналізовані різні алгоритми кластеризації для часових рядів, а особлива увага була приділена алгоритму DTW. Використання DTW дозволило ефективно враховувати схожість між часовими рядами, уникнути проблем зі зсувом та масштабуванням даних. Нормалізація даних також була використана для забезпечення точності кластеризації, дозволяючи порівнювати значення на різних шкалах.

Однак, одним з викликів у роботі з часовими рядами є їх різна довжина. Для цього була розроблена математична модель, яка дозволяє фіксувати довжину рядів. Це дозволило однаково обробляти ряди різної довжини і уникнути втрати важливої інформації. Після реалізації моделі та проведення відповідних експериментів було проведено аналіз результатів кластеризації, що підтвердило ефективність розробленої моделі.

Окремо було розроблено вебзастосунок на мові Java для реалізації та візуалізації кластеризації часових рядів. Цей застосунок дозволяє зручно завантажувати дані, проводити кластеризацію та відображати результати на графіках. Після реалізації було проведено тестування системи, що підтвердило її працездатність та надійність. Вебзастосунок належно виконує

свої функції, забезпечуючи користувачам зручний інтерфейс для взаємодії з кластеризацією часових рядів. Результати кластеризації надають важливі відомості про спортсменів підводного плавання, дозволяючи виявляти схожість між ними та отримувати корисну інформацію для подальшого аналізу.

Отже, оцінка показала, що розроблені моделі та методи кластеризації часових рядів є ефективними і можуть бути успішно використані для аналізу результатів спортсменів підводного плавання. Результати роботи відкривають широкі можливості для подальшого вдосконалення та застосування у реальних сценаріях.

Окрім того, робота вносить важливий внесок у галузь аналізу часових рядів та спортивних даних. Вона показує потенціал застосування кластерного аналізу у спорті та може сприяти поліпшенню тренувальних програм, стратегій та прийняттю рішень в галузі спорту.

В цілому, результати кваліфікаційної роботи підтверджують її важливість та актуальність у контексті аналізу результатів спортсменів підводного плавання за допомогою кластеризації часових рядів.

Результати роботи апробовано у вигляді тез доповідей під час V Міжнародної науково-теоретичної конференції «Theory and practice of modern science» [35].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aggarwal, C. C. (2015). Data mining: the textbook (Vol. 1). New York: springer.
2. Hamilton, J. D. (2020). Time series analysis. Princeton university press.
3. Chatfield, C., & Xing, H. (2019). The analysis of time series: an introduction with R. CRC press.
4. Бодянський, Є. В., Винокурова, Є. А., Пелешко, Д. Д., Кобилін, І.О., & Кобилін, О. А. (2017). Нечітка кластеризація часових рядів з нерівномірними та асинхронними тактами квантування. Системи обробки інформації, (5), 47-54.
5. Athiyarath, S., Paul, M., & Krishnaswamy, S. (2020). A comparative study and analysis of time series forecasting techniques. SN Computer Science, 1(3), 175.
6. Beard, E., Marsden, J., Brown, J., Tombor, I., Stapleton, J., Michie, S., & West, R. (2019). Understanding and using time series analyses in addiction research. Addiction, 114(10), 1866-1884.
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. 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.
9. 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 Radioelectronics, Telecommunications and Computer Engineering (TCSET) (pp. 930-935). IEEE.

10. 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.
11. Setlak, G., Bodyanskiy, Y., Pliss, I., Vynokurova, O., Peleshko, D., & Kobylin, I. (2018). Adaptive fuzzy clustering of multivariate short time series with unevenly distributed observations based on matrix neuro-fuzzy self-organizing network. In *Advances in Fuzzy Logic and Technology 2017: Proceedings of: EUSFLAT-2017–The 10th Conference of the European Society for Fuzzy Logic and Technology*, September 11-15, 2017, Warsaw, Poland IWIFSGN'2017–The Sixteenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets, September 13-15, 2017, Warsaw, Poland, Volume 3 10 (pp. 308-315). Springer International Publishing.
12. 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.
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. Giannella, C. R. (2021). Instability results for Euclidean distance, nearest neighbor search on high dimensional Gaussian data. *Information Processing Letters*, 169, 106115.
15. 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.
16. 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.

17. Shafronenko, A., Bodyanskiy, Y. V., Klymova, I., & Holovin, O. (2020, May). Online credibilistic fuzzy clustering of data using membership functions of special type. In CMIS (pp. 744-753).

18. 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.

19. 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.

20. Bodyanskiy, Y. V., Shafronenko, A., & Klymova, I. (2021, April). Adaptive Recovery of Distorted Data Based on Credibilistic Fuzzy Clustering Approach. In COLINS (pp. 6-15).

21. 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.

22. 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.

23. 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.

24. Winkler, R., Klawonn, F., & Kruse, R. (2011). Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications (IJFSA)*, 1(1), 1-16.
25. Yu, S. S., Chu, S. W., Wang, C. M., Chan, Y. K., & Chang, T. C. (2018). Two improved k-means algorithms. *Applied Soft Computing*, 68, 747-755.
26. Borlea, I. D., Precup, R. E., Borlea, A. B., & Iercan, D. (2021). A unified form of fuzzy C-means and K-means algorithms and its partitional implementation. *Knowledge-Based Systems*, 214, 106731.
27. Ghazal, T. M., Hussain, M. Z., Said, R. A., Nadeem, A., Hasan, M. K., Ahmad, M., ... & Naseem, M. T. (2021). Performances of K-Means Clustering Algorithm with Different Distance Metrics. *Intelligent Automation & Soft Computing*, 30(2).
28. Arora, J., Khatter, K., & Tushir, M. (2019). Fuzzy c-means clustering strategies: A review of distance measures. *Software Engineering: Proceedings of CSI 2015*, 153-162.
29. Askari, S. (2021). Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. *Expert Systems with Applications*, 165, 113856.
30. Okawa, M. (2019). Template matching using time-series averaging and DTW with dependent warping for online signature verification. *IEEE Access*, 7, 81010-81019.
31. Alaei, S., Mercer, R., Kamgar, K., & Keogh, E. (2021). Time series motifs discovery under DTW allows more robust discovery of conserved structure. *Data Mining and Knowledge Discovery*, 35, 863-910.
32. 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.

33. 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.

34. 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.

35. Makhanov, R., (2023). Theory and practice of modern science: collection of scientific papers «SCIENTIA» with Proceedings of the V International Scientific and Theoretical Conference, May 12, 2023. Kraków, Republic of Poland: European Scientific Platform.