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 другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ КЛАСТЕРИЗАЦІЇ ЧАСОВИХ РЯДІВ ЗА ЗНАЧЕННЯМИ ПАРАМЕТРІВ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-20-1
Зінченко Г.В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Кіріченко Л.О.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Зінченку Глібу В'ячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів кластеризації часових рядів за значеннями параметрів
затверджена наказом по університету від «22» 10 2021 року №1574Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 29 листопада 2021 р.
3. Вихідні дані до роботи вибірка даних про бронювання готелів, вибірка даних про акції компаній, перелік використовуваних програмних засобів, теоретичні відомості про методи кластеризації часових рядів.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів кластеризації часових рядів.
2. Порівняння існуючих методів кластеризації часових рядів.
3. Застосування алгоритмів на реальних прикладах.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми кластеризації часових рядів, постановка задачі, тестові зображення.

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 Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.10.2021	
2	Аналіз завдання, підбір літератури	22.10.21-24.10.21	
3	Аналіз літератури з досліджуваної проблеми	24.10.21-30.10.21	
4	Аналіз технічних засобів	01.11.21-05.11.21	
5	Реалізація алгоритмів	05.11.21-13.11.21	
6	Програмна реалізація	13.11.21-17.11.21	
7	Оформлення пояснювальної записки	17.11.21-23.11.21	
8	Перевірка на плагіат	23.11.2021	
9	Рецензування	23.11.2021	
10	Підготовка презентації та доповіді	25.11.2021	
11	Занесення роботи в електронний архів	26.11.2021	
12	Попередній захист кваліфікаційної роботи	13.12.2021	

Дата видачі завдання __20___ __жовтня__ 2021 р.

Студент _____
(підпис)

Керівник роботи _____ ____проф. Кіріченко Л.О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 56 с., 29 рис., 43 джерела.

КЛАСТЕРИЗАЦІЯ ЧАСОВИХ РЯДІВ, *K*-MEANS, DBSCAN, АНАЛІЗ ДАНИХ.

Об’єктом дослідження є методи кластеризації часових рядів.

Метою дослідження є аналіз методів кластеризації часових рядів за значеннями параметрів.

Використано методи аналізу та обробки даних. Проведено дослідження існуючих методівкластеризаціх часових рядів, та аналізу даних.

У результаті роботи здійснена програмна реалізація кластеризації часових рядів на прикладі реальних даних.

TIME SERIES CLUSTERING, *K*-MEANS, DBSCAN, DATA ANALYSIS.

The object of research is the methods of time series clustering.

The aim of the study is to analyze the methods of clustering time series by parameter values. Methods of data analysis and processing are used.

A study of existing method sclustering of time series and data analysis.

As a result, the software implementation of the classification of time series on the example of real data.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Огляд основних методів кластеризації часових рядів 9
1.1 Основні поняття 9
1.2 Affinity Propagation 14
1.3 DBSCAN 16
1.4 *K*-means 17
1.5 Self Organizing Maps 19
1.6 Постановка задачі дослідження 21
2 Огляд методів 23
2.1 *K*-means 23
2.2 DBSCAN 30
3 Реалізація методів 36
3.1 Обґрунтування вибору середовища програмної реалізації 36
3.2 Додаткові бібліотеки 37
3.3 Кластеризація на прикладі бронювання готелів 40
3.4 Кластеризація на прикладі акцій компаній 45
Висновки 51
Перелік джерел посилання 52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DBSCAN – Density-Based Spatial Clustering Of Applications With Noise

AP – Affinity Propagation

SOM – Self Organizing Maps

ВСТУП

Аналіз часових рядів – це популярна галузь науки про дані, яка включає розробку моделей (статистичних або моделей машинного навчання), які можуть найкращим чином описати спостережувані часові ряди та, можливо, пояснити основні причини та закономірності. Для цього можна використовувати декілька методів, таких як виявлення аномалій, створювати авторегресивні моделі, розпізнавати тенденції, вимірювати подібність часових рядів. Що може бути корисним у багатьох різних галузях промисловості. Було проведено детальне дослідження, щоб знайти найкращий спосіб відповісти на ці питання.

Основною метою кластеризації часових рядів є розподіл даних часових рядів на групи на основі подібності чи відстані, щоб часові ряди в одному кластері були подібними. Спочатку здається, що це та сама проблема, що і будь-яка основна кластеризація, але тут треба прийняти конкретні дані та конкретні рішення, перш ніж пристосувати модель.

Актуальність дослідження полягає у тому, що кількість даних, які треба постійно аналізувати постійно зростає. У випадку, якщо є великий масив даних, то найефективніший спосіб зрозуміти, що з ними робити – розсортувати їх у групи для первинного аналізу. Групувати можна за допомогою – сегментації (ви самі задаєте критерії, наприклад, вікові та цінові групи) або кластеризації (математичний алгоритм сам виявляє «сполучний» критерій або ознаку, що поєднує дані). Цінність та основна відмінність кластеризації полягає в тому, що алгоритми виявляють та поєднують параметри зі схожими рисами з первинного масиву даних.

Ось приклади лише кількох практичних завдань, які вимагають кластеризації часових рядів, насправді ж таких завдань набагато більше:

– знаходження фінансових показників зі подібною динамікою;

– об’єднання пацієнтів у однорідні групи за формою електрокардіограми;

– класифікація типів активності людини за показаннями «розумного годинника»;

– виявлення будинків із подібними профілями споживання електроенергії тощо.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ КЛАСТЕРИЗАЦІЇ ЧАСОВИХ РЯДІВ

1.1 Основні поняття

Часовий ряд (або ряд динаміки) – зібраний у різні моменти часу статистичний матеріал про значення будь-яких параметрів (у найпростішому випадку одного) досліджуваного процесу. Рисунок 1.1 демонструє приклад часового ряду. Кожна одиниця статистичного матеріалу називається виміром чи відліком, також допустимо називати його рівнем на зазначений із ним час. У тимчасовому ряді для кожного відліку має бути вказано час виміру або номер виміру по порядку [1]. Тимчасовий ряд істотно відрізняється від простої вибірки даних, оскільки за аналізу враховується взаємозв’язок вимірів з часом, а чи не лише статистичне розмаїтість і статистичні характеристики выбірки.

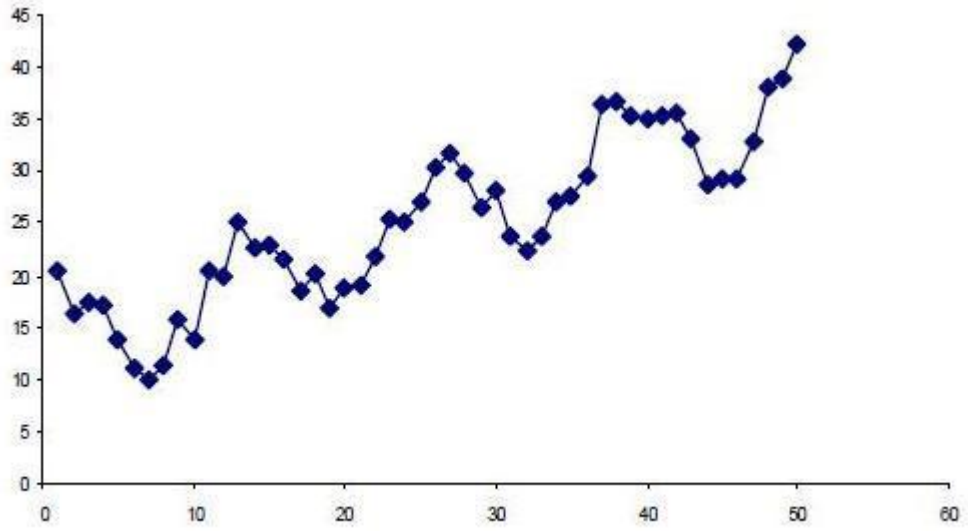

Рисунок 1.1 – Приклад часового ряду

Кластеризація, або кластерний аналіз – це статистична процедура, задача якої полягає в розбитті вибірки об’єктів на підмножини, що не перетинаються і називаються кластерами [2, 3]. Кожен кластер має складається зі схожих об’єктів, а об’єкти різних кластерів мають істотно відрізнятися один від одного. Задача кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без вчителя. Ще її можна описати через задачу класифікації.

Задача кластеризації – це по факту задача класифікації, бо в обох випадках ми ділимо об’єкти на основі їх подібності між собою, але у випадку кластеризації приналежність навчальних об’єктів будь-яким класам не задається. Така задача – загальна, тому для її розв’язання використовуються різні підходи. Алгоритми побудови кластерів можуть дуже відрізнятися у підходах до того, що відносити в один кластер і як їх взагалі ефективніше шукати. Кластери можна утворювати ґрунтуючись на відстані між ними, на щільності ділянок у просторі даних, інтервалах або на конкретних статистичних розподілах [4]. Це все залежать від конкретного набору даних та мети використання результатів. Кластерний аналіз не є автоматизованим, це скоріше ітераційний процес, тому що часто доводиться змінювати метод опрацювання даних та параметри моделі, поки не буде отримано з результат з заданими властивостями. Розв’язок неоднозначний, і на це є кілька причин. По-перше, не існує найкращого критерію якості кластеризації. Відомий цілий ряд досить ефективних критеріїв, а також ряд алгоритмів, які не мають чітко вираженого критерію, але все одно здійснюють досить якісну кластеризацію по побудові. Всі вони можуть давати різні результати. По-друге, число кластерів, як правило, не відомо заздалегідь і встановлюється відповідно до деякого суб’єктивного критерія. По-третє, результат кластеризації істотно залежить від метрики ρ, вибір якої, як правило, також суб’єктивний і визначається спеціалістом.

Задача групування набору об’єктів полягає в тому, що об’єкти в одному кластері більш схожі один на одного, ніж об’єкти в інших кластерах.

Подібність – це буквально кількість, яка собою відображає міцність взаємозв’язку між двома об’єктами. Кластеризація використовується в основному для видобутка даних, а також в інших областях, таких як машинне навчання, розпізнавання образів, аналіз зображень, пошук інформації, біоінформатика, стиснення даних і комп’ютерна графіка. Існує два типа кластеризації: жорстка та м’яка. У жорсткій кластеризації кожен об’єкт даних або повністю належить кластеру чи взагалі не належить. В м’якій кластеризації точка чи об’єкт даних може з певною ймовірністю належати більш ніж одному кластеру [5].

Типи вхідних даних та проблема міри відстані:

– кожен об’єкт описується набором своїх характеристик, які називаються ознаками. Ознаки можуть бути числовими або нечисловими;

– матриця відстаней між об’єктами. Кожен об’єкт описується відстанями до всіх інших об’єктів навчальної вибірки.

Дані, що використовуються в кластерному аналізі, можуть мати інтервальний, порядковий або категоріальний тип. Однак наявність суміші різних типів змінної зробить аналіз більш складним. Це пояснюється тим, що в кластерному аналізі необхідно мати певний спосіб вимірювання відстані між спостереженнями, тобто тип використовуваної міри залежить від типу даних. Для вимірювання відстані для бінарних та категоріальних даних було запропоновано ряд різних варіантів [6]. Наприклад, для інтервальних даних найчастіше використовуваною мірою відстані є евклідова відстань. Кластеризація – це досить суб’єктивна задача, для розв’язання якої може існувати більше одного правильного алгоритму. Кожен алгоритм слідує своєму набору правил для визначення «подібності» між об’єктами даних. Найбільш відповідний алгоритм кластеризації для конкретної проблеми часто потрібно вибирати експериментально, якщо немає математичної причини віддати перевагу одному алгоритму кластеризації над іншим. Алгоритм може добре працювати на певному наборі даних, але не працювати для іншого. Алгоритми кластеризації можна класифікувати на основі

кластерної моделі, яка ґрунтується на тому, як саме вони формують кластери або групи [7]. Наприклад, такі як:

– кластеризація на основі зв’язку: основна ідея кластеризації полягає в тому, що точки даних, які знаходяться ближче в просторі даних, є більш спорідненими (подібними). Кластери формуються шляхом з’єднання точок даних відповідно до їх відстані. На різних відстанях будуть формуватися різні кластери, які можуть бути представлені за допомогою дендрограми (це також називають «ієрархічною кластеризацією»). Ці методи створюють саме ієрархію, а не унікальне розбиття набору даних, з якої користувачі все ще повинні обрати відповідні кластери, тобто рівень, на якому вони хочуть кластерувати. Вони також не дуже стійкі до викидів, які можуть з’являтися у вигляді додаткових кластерів або навіть призводити до злиття інших кластерів;

– кластеризація на основі центроїда: у цьому типі кластери представлені центральним вектором або центроїдом. Цей центроїд не обов’язково має бути в наборі даних. Це ітеративні алгоритми кластеризації, в яких поняття подібності виводиться з того, наскільки близька точка даних до центроїда кластера. *k*-середніх, наприклад, є кластеризацією на основі центроїда.

Також є ієрархічні алгоритми кластеризації Ієрархічні алгоритми кластеризації, або алгоритми таксономії, будують не одне розбиття вибірки на непересічні класи, а систему вкладених розбиттів. Результат таксономії зазвичай представляється у вигляді таксономічного дерева – дендрограми [8]. Класичним прикладом такого дерева є ієрархічна класифікація тварин і рослин. Дендограми дозволяє уявити кластерну структуру у вигляді плаского графіка незалежно від того, яка розмірність початкового простору. Існують і інші способи візуалізації багатовимірних даних, такі як багатовимірне шкалювання або карти Кохонена, але вони привносять в картину штучні спотворення, вплив яких досить важко оцінити. Є два типи методів:

– агломератні методи: нові кластери утворюються шляхом об’єднання дрібніших кластерів, і таким чином дерево створюється від листя до стовбура;

– дивізійні методи: нові кластери створюються шляхом ділення більших кластерів на більш дрібні, і таким чином дерево створюється від стовбура до листя.

Подібність кластерів часто розраховується через «неподібність», наприклад, евклідова відстань між двома кластерами.

Таким чином, чим більше відстань між двома кластерами, тим краще. Ключовою операцією в ієрархічній агломераційній кластеризації є неодноразове об’єднання двох найближчих кластерів у один кластер, але дуже важливо спочатку відповісти на три питання: як ви представити кластер з більш ніж однією точкою, як визначити «близькість» кластерів та коли перестати поєднувати кластери [9]. Злиття кластерів припиняється в залежності від доступної інформації про дані, які є на вході. Якщо групувати футболістів на полі на основі їхніх позицій на полі, яке представлятиме їх координати для розрахунку відстані між гравцями, треба зупинитися на лише двох кластерах, оскільки можуть бути тільки дві команди, які грають у футбольний матч. Алгоритм методу виглядає таким чином:

Крок 1. Обчислити матрицю близькості, що містить відстань між кожною парою шаблонів. Розглядати кожен зразок як окремий кластер.

Крок 2. Знайти найбільш схожу пару кластерів за допомогою матриці близькості. Об’єднати ці два кластера в один більший кластер. Оновити матрицю близькості, щоб відобразити цю операцію злиття.

Крок 3. Якщо всі шаблони знаходяться в одному кластері, зупинитися. В іншому випадку перейти до Кроку 2.

Ієрархічний алгоритм дає дендограму, що представляє собою складене групування шаблонів і рівні схожості, на яких змінюються самі групування. Більшість ієрархічних алгоритмів кластеризації є варіантами однозв’язного і повнозв’язного алгоритму, а також алгоритму мінімальної дисперсії.

Найбільш популярними з них є однозв'язний та повнозв'язний алгоритми. Вони відрізняються тим, як вони характеризують подібність між парою кластерів. У методі з однозв'язним алгоритмом відстань між двома кластерами – це мінімум відстаней між усіма парами шаблонів, взятих з двох кластерів (один з першого кластера, другий – з другого). У повнозв'язному алгоритмі відстань між двома кластерами – це максимум усіх попарних відстаней між шаблонами в двох кластерах. В обох випадках два кластери об'єднуються для формування більшого кластера на основі мінімальних критеріїв відстані. Повнозв'язний алгоритм створює щільні та компактні кластери, а однозв'язний алгоритм, навпаки, страждає від ланцюгового. Він має тенденцію виробляти кластери, які є занадто громіздкими або подовженими.

1.2 Affinity Propagation

Кластеризацію можна уявити як завдання дискретної максимізації з обмеженнями. Нехай на множині елементів задана функція подібності $s(i, j)$. Наше завдання знайти такий вектор міток $c = \{c_1, c_2 ... c_N\}, c_i \in \{1...N\}$, який максимізує функцію.

$$S(c) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta(c_k), \tag{1.1}$$

де $\delta(c_k)$ – член обмежувач, рівний $-\infty$, якщо існує точка i, яка вибрала точку k своїм лідером $(c_i = k)$, але сама k лідером себе не рахує $(c_k \neq k)$.

Знаходження ідеального c – це NP-складне завдання, відоме як завдання про розміщення об'єктів. Тим не менш, для її вирішення існує кілька наближених алгоритмів. У таких методах s_i, c_i і δ_i представляються вершинами дводольного графа, після чого між ними відбувається обмін

інформації, що дозволяє з імовірнісної точки зору оцінити, яка мітка краще підійде для кожного елемента. Для поширення повідомлень на дводольних графах. Взагалі поширення близькості – це окремий випадок (скоріше, звуження) циклічного поширення переконань (loopy belief propagation – LBP), але замість суми ймовірностей (підтип Sum-Product) у деяких місцях ми беремо лише максимальну (підтип Max-Product) з них. По-перше, LBP-Sum-Product набагато складніше, по-друге, там легше зіткнутися з обчислювальними проблемами, по-третє теоретики стверджують, що для завдання кластеризації це має більший сенс.

Автори AP багато говорять про «повідомлення» від одного елемента графа до іншого [10]. Така аналогія походить з висновку формул через поширення інформації у графі. На мій погляд, вона трохи плутає, адже в реалізаціях алгоритму жодних повідомлень «точка-точка» немає, зате є три матриці S , R та A . Можна запропонувати наступне трактування: При обчисленні $r(i,k)$ йде пересилання повідомлень від точок даних i до потенційних лідерів k – переглядаються матриці S і A вздовж рядків При обчисленні $a(i,k)$ йде пересилання повідомлень від потенційних лідерів k до всіх інших точок i – ми переглядаємо матриці S і R вздовж стовпців Affinity propagation детерміновано. Він має складність $O(N^2T)$, де N – розмір набору даних, а T – кількість ітерацій і займає $O(N^2)$ пам’яті.

Існують модифікації алгоритму для розріджених даних, але все одно AP сильно сумує зі збільшенням розміру датасету. Це досить серйозний недолік. Зате поширення близькості залежить від розмірності елементів даних. Поки що не існує розпаралеленого варіанта AP. Тривіальне ж розпаралелювання у вигляді безлічі запусків алгоритму не підходить через детермінованість. В офіційному FAQ написано, що варіантів з додаванням даних у реальному часі теж немає. Існує прискорений варіант AP, метод ґрунтується на ідеї, що необов’язково обчислювати взагалі всі оновлення матриць доступності та відповідальності, адже всі точки в гущі інших точок

відносяться до далеких від них екземплярів приблизно однаково. Частини $\max(0, x)$ в них виглядають підозріло схожим на нейронно-сіткові ReLu.

Крім того, тепер можна додатавати до формул додаткові члени, відповідальні за необхідну поведінку. Таким чином, можна «підштовхнути» алгоритм у бік кластерів певного розміру або форми. Також можна накласти додаткові обмеження, якщо відомо, що якісь елементи з більшою ймовірністю належать до однієї множини. Affinity propagation, як і багато інших алгоритмів, можна перервати достроково, якщо R і A перестають оновлюватися. Для цього майже у всіх реалізаціях використовується два значення: максимальна кількість ітерацій та період, з яким перевіряється величина оновлень [11]. Affinity propagation схильний до обчислювальних осциляцій у випадках, коли є кілька хороших розбиття на кластери. Щоб уникнути проблем, по-перше, на самому початку до матриці подібності додається трохи шуму (зовсім трохи, щоб не вплинути на детерменованість, у sklearn-імплементації порядку 10^{-16}), а по-друге, при оновленні R і A використовується не просте присвоєння, а присвоєння з екпоненціальним згладжуванням. Друга оптимізація до того ж загалом добре впливає якість результату, але через неї збільшується кількість необхідних ітерацій T. Найкращою практикою є використання параметра згладжування $0,5 \le \gamma \le 1,0$ зі значенням за промовчанням 0,5. Якщо алгоритм не сходиться або частково сходиться, слід збільшити γ до 0,9 або до 0,95 з відповідним збільшенням кількості ітерацій [12].

1.3 DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise, щільнісний алгоритм просторової кластеризації з присутністю шуму), як випливає з назви, оперує щільністю даних. На вхід він потребує вже знайому

матрицю близькості та два параметри – радіус ε-околу та кількість сусідів. Одразу можна поставити питання, як їх обрати, причому тут щільність, і чому саме DBSCAN добре розправляється з гучними даними [13]. Без цього складно визначити межі його застосування. Введемо кілька визначень. Нехай задана деяка симетрична функція відстані $\rho(x, y)$ і константи ε і m. Тоді Назвемо область $E(x)$, для якої $\forall y : \rho(x, y) \le \varepsilon$, ε-околом об'єкта x. Кореневим об'єктом або ядерним об'єктом ступеня m називається об'єкт, ε-окіл якого містить щонайменше m об'єктів: $|E(x)| \ge m$. Об'єкт p безпосередньо щільно досягається з об'єкта q, якщо $p \in E(q)$ і q – кореневий об'єкт. Об'єкт p щільно-досяжний з об'єкта q, якщо $\exists p_1, p_2 ... p_n, p_1 = q, p_n = p$, такі що $\forall i \in 1...n-1 : p_{i+1}$ безпосередньо щільно-досяжний з p_i.

Виберемо якийсь кореневий об'єкт p з датасету, позначимо його і помістимо всіх його безпосередньо щільно-досяжних сусідів у список обходу. Тепер для кожної q зі списку: позначимо цю точку, і якщо вона теж коренева, додамо всіх її сусідів до списку обходу. Тривіально доводиться, що кластери помічених точок, сформовані в ході цього алгоритму максимальні (тобто їх не можна розширити ще однією точкою, щоб задовольнялися умови) і зв'язні в сенсі досяжності. Звідси випливає, що якщо ми обійшли не всі крапки, можна перезапустити обхід із якогось іншого кореневого об'єкта, і новий кластер не поглине попередній [14].

1.4 *K*-means

Алгоритм *K*-means – це ітераційний алгоритм, який намагається розділити набір даних на *K* попередньо визначені окремі підгрупи (кластери), що не перекриваються, де кожна точка даних належить лише одній групі. Він

намагається зробити точки даних всередині кластера якомога схожими, водночас зберігаючи кластери якомога різними (далекими). Він призначає точки даних кластеру таким чином, щоб сума квадратів відстані між точками даних і центроїдом кластера (середнє арифметичне всіх точок даних, які належать до цього кластера) була мінімальною. Чим менше ми маємо варіацій всередині кластерів, тим однорідніші (подібніші) точки даних в одному кластері [15, 16].

Алгоритм *k*-means працює так:

Крок 1. Вкажіть кількість кластерів *K*.

Крок 2. Ініціалізуйте центроїди, спочатку перемішавши набір даних, а потім випадковим чином вибравши *K* точок даних для центроїдів без заміни.

Крок 3. Продовжуйте повторювати, доки не зміниться центроїди. Тобто призначення точок даних кластерам не змінюється.

Крок 4. Обчисліть суму квадратів відстані між точками даних і всіма центроїдами.

Крок 5. Призначення кожної точки даних до найближчому кластеру (центроїду).

Крок 6. Обчисліть центроїди для кластерів, взявши середнє значення всіх точок даних, які належать кожному кластеру.

Підхід *k*-means для вирішення проблеми називається очікуванням-максимізацією. *E*-крок призначає точки даних найближчому кластеру. *M*-крок обчислює центроїд кожного кластера. Нижче наведено розбір того, як ми можемо розв’язати це математично (не соромтеся його пропустити). Оскільки алгоритми кластеризації, включаючи *k*-means, використовують вимірювання на основі відстані для визначення схожості між точками даних, рекомендується стандартизувати дані, щоб вони мали середнє значення нуль і стандартне відхилення одиниці, оскільки майже завжди ознаки в будь-якому наборі даних будуть мати різні одиниці виміру. наприклад, вік проти доходу.

Враховуючи ітераційний характер *k*-means і випадкову ініціалізацію центроїдів на початку алгоритму, різні ініціалізації можуть призвести до різних кластерів, оскільки алгоритм *k*-means може застрягти в локальному оптимумі і не сходитися до глобального оптимуму. Тому рекомендується запустити алгоритм, використовуючи різні ініціалізації центроїдів і вибрати результати запуску, які дали меншу суму квадратів відстані.

1.5 Self Organizing Maps

Самоорганізуючі карти – це неконтрольована модель навчання, яка використовує нейронну мережу для абстрагування та візуалізації багатовимірних даних на двовимірній карті, представленій фіксованою сіткою вузлів. Вперше його представив Теуво Кохонен у 1982 р. [17], як засіб збереження топології у багатовимірному наборі на карта меншого розміру. Карта Кохонена – це набір вузлів, кожен з яких має а вектор ваги моделі $m_i = [\mu_1, ..., \mu_n] \in R_n$, де вектор моделі є такої ж розмірності, як і вхідний простір даних *X*, які мають бути представлені [17].

Для кожної ітерації в процесі навчання алгоритм спочатку знаходить найкращий вузол відповідності до спостереження, який називається BMU (найкраща одиниця відповідності), а потім переміщує БМУ та вузли поблизу БМУ до сьогодення спостереження [18]. Показано високорівневий алгоритм самоорганізаційної карти нижче. Цикл while алгоритму 1 відноситься до критерію зупинки, у даному випадку фіксованого кількість ітерацій *T* через дані. Алгоритм самоорганізуючої карти в деяких термінах штучна нейронна мережа (ANN), без зворотного поширення процедура корекції. Дано набір *k* *n*-вимірних векторів даних $x_i \in R_n$, нехай карта Кохонена – це двовимірна сітка з m нейронів з $m = x * y$, де *x* і *y* визначені вхідні дані. Сітка може бути шістнадцятковою або прямокутної форми. Кожен вузол *M* має ваговий

вектор $m_i \in R_n$, який може бути ініціалізується випадковою $N_n(0, I)$-змінною, де I є ідентичною матрицею. Далі алгоритм вибирає та оцінює спостереження x_k з набору даних. Найкращий відповідний вузол BMU с знаходить за допомогою метрики $d(x, m)$ і вибору BMU, який є найближчим, як зазначено в рівнянні нижче [18]. У цьому випадку ми використовуємо метрика евклідової відстані: $c = \arg\min_i \{\| x - m_i \|\}$.

Потім алгоритм переміщує найефективніший блок с і вузли в ньому околі ближче до спостереження, використовуючи наступне рівняння:

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t)(x(t) - m_i(t)), \qquad (1.2)$$

де $\alpha(t)$ є коефіцієнт швидкості навчання, який спадає з кожною ітерацією;

$h_{ci}(t)$ – це ядро згладжування сусідства, яке також може згасати з часом.

Приклад самоорганізаційної карти можна показати за допомогою тривимірного зображення вектор кольору, візуалізований на двовимірній поверхні карти Кохонена [19].

Кожен тривимірний вектор представляє колір за допомогою системи RGB, вимірюючи кількість червоного, зеленого та синього в кожному кольорі. Кожен вимір може прийняти значення в діапазоні від 0 до 255. Приклад даних містить 100 кольорів, вибраних випадковим чином взявши сто малюнків із тривимірного дискретного однорідного вектора зі значеннями в діапазоні від 0 до 255. Карта Кохонена 8×10 потім навчається на набір даних. Оскільки кожен вектор вузла карти ініціалізується з нормального розподілу із середнім нулем вся карта після ініціалізації буде чорною (оскільки вектор RGB (0, 0, 0) чорний). Під трьома цифрами показано отриману карту після одного, десяти і сто ітерацій. На рисунку 1.2 продемонстровано приклад роботи алгоритму.

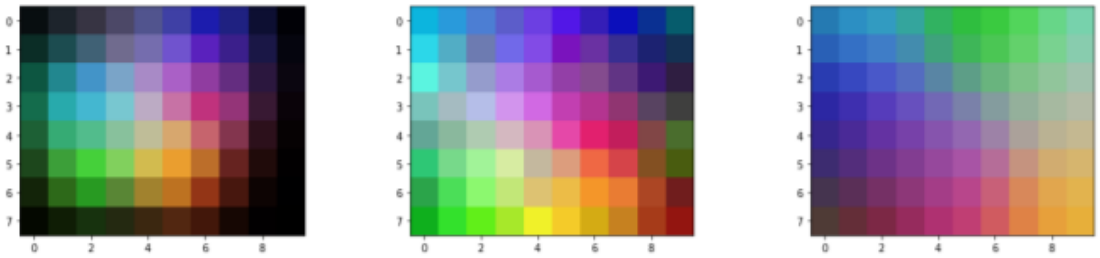

Рисунок 1.2 – Приклад роботи алгоритму SOM

Після однієї ітерації деякі вузли на карті вже переміщено ближче до кольорів у наборі даних. Однак зрозуміло, що впорядкування між вузли не зійшлися, оскільки є дві групи блакитних кластерів. Після 10 ітерацій, чорний тепер став кольорами. Після 100 ітерацій карта показує розумну карту з чіткими угрупованнями синього, зеленого, червоного та жовтого, як можна було б очікувати [20].

1.6 Постановка задачі дослідження

Таким чином, кластеризація є актуальним завданням для обробки і аналізу часових рядів. Тому ставиться завдання аналізу існуючих методів кластеризації часових рядів.

Об’єктом дослідження є методи кластеризації часових рядів.

Метою дослідження є аналіз методів кластеризації часових рядів за значеннями параметрів.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати існуючі методи кластеризації часових рядів, проаналізувати основні проблеми кластеризації;

– детально вивчити методи *K*-means та DBSCAN;

– реалізувати методи кластеризації часових рядів у середовищі Google Colaboratory;

– протестувати реалізовані методи на прикладах реальних часових рядів із вибірок даних про бронювання готелів та про ціну акцій компаній за останній місяць.

2 ОГЛЯД МЕТОДІВ

Аналіз існуючих методів у першому розділі показав, що найкращими методами для реалізації та демонстрації є *K*-means та DBSCAN, тому у другому розділі дані методи були оглянуті більш детально.

2.1 *K*-means

Розглянемо найпоширеніші методи кластеризації часових рядів, які використовують алгоритм *k*-means. Ці методи включають: евклідові *k*-means, *k*-means DBA та *k*-means Soft-DTW. Одним із поширених методів кластеризації часових рядів є підхід *k*-means, де евклідова відстань використовується як міра близькості:

$$d_{E1}(X_i, X_j) = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2}, \tag{2.1}$$

де X_i і X_j – це два часових ряди довжини m.

Суть алгоритму *k*-means полягає в тому, що спочатку вибираються довільні центри. Потім навколо цих центрів групуються решта елементів, які необхідно розділити на класи. На наступному кроці для отриманих кластерів обчислюються нові центри так, щоб квадрат евклідової відстані від елемента кластера до його центроїда був меншим, ніж відстань до центрів решти кластерів. При цьому алгоритм розміщує центри кластерів (центроїди) так, щоб середні значення для списків елементів всередині побудованих кластерів максимально відрізнялися [21]. Таким чином, метод евклідових *k*-середніх розбиває часові ряди довжини вибірки m на k групи (кластери). Це поділ відбувається шляхом мінімізації загального квадрата відхилення точок кластера від центроїдів цих кластерів:

$$\min[\sum_{i=1}^{k} \sum_{x(j) m S_i} \| x^{(j)} - \mu_i \|^2], \quad (2.2)$$

де $x^{(j)} \in R^n$, $\mu_i \in R^n$, μ_i – центроїди кластера S_i.

Використання евклідового методу *k*-means має кілька недоліків: необхідно заздалегідь визначити кількість отриманих кластерів, які не завжди можуть бути відображені. Метод чутливий до вибору початкових центрів кластерів – це призводить до збільшення ймовірності помилки та можливості отримання результатів, що відрізняються один від одного при перезапуску алгоритму. Бувають також випадки, коли об'єкт може належати різним кластери. Незважаючи на недоліки, евклідовий *k*-means є простим алгоритмом, добре підходить для розуміння загальних процесів кластеризації і є хорошою основою для побудови нових розширених алгоритмів на його основі. При кластеризації часових рядів важливо враховувати той факт, що що деякі ряди можуть бути майже однаковими, але в той же час ці ряди можуть бути зміщені в часі (по осі часу). Тому доцільно використовувати аметрику, яка реалізована в алгоритмі динамічного перетворення часової шкали (DTW). Але по-перше, чому загальна евклідова метрика відстані непридатна для часових рядів? Тому, що, він інваріантний до часових зрушень, ігноруючи часовий вимір даних. Якщо два часових ряди сильно корелюють, але один зміщений навіть на один часовий крок, евклідова відстань помилково вимірює їх як більш віддалені один від одного.

Замість цього для порівняння рядів краще використовувати динамічне викривлення часу (DTW). DTW – це методика вимірювання подібності між двома тимчасовими послідовностями, які не вирівнюються точно за часом, швидкістю чи довжиною. DTW обчислюється як квадратний корінь із суми квадратів відстаней між кожним елементом у *X* та його найближчою точкою в *Y*.

Також треба звернути увагу на те, що DTW(*X*, *Y*) ≠ DTW(*Y*, *X*). DTW порівнює кожен елемент серії *X* із кожним елементом серії *Y* (*n*×*m*

порівнянь). Порівняння $d(x_i, y_j)$ — це просто віднімання x_i від y_j [22]. Потім для кожного x_i в X DTW вибирає найближчу точку в Y для обчислення відстані, що й продемонстровано на рисунку 2.1.

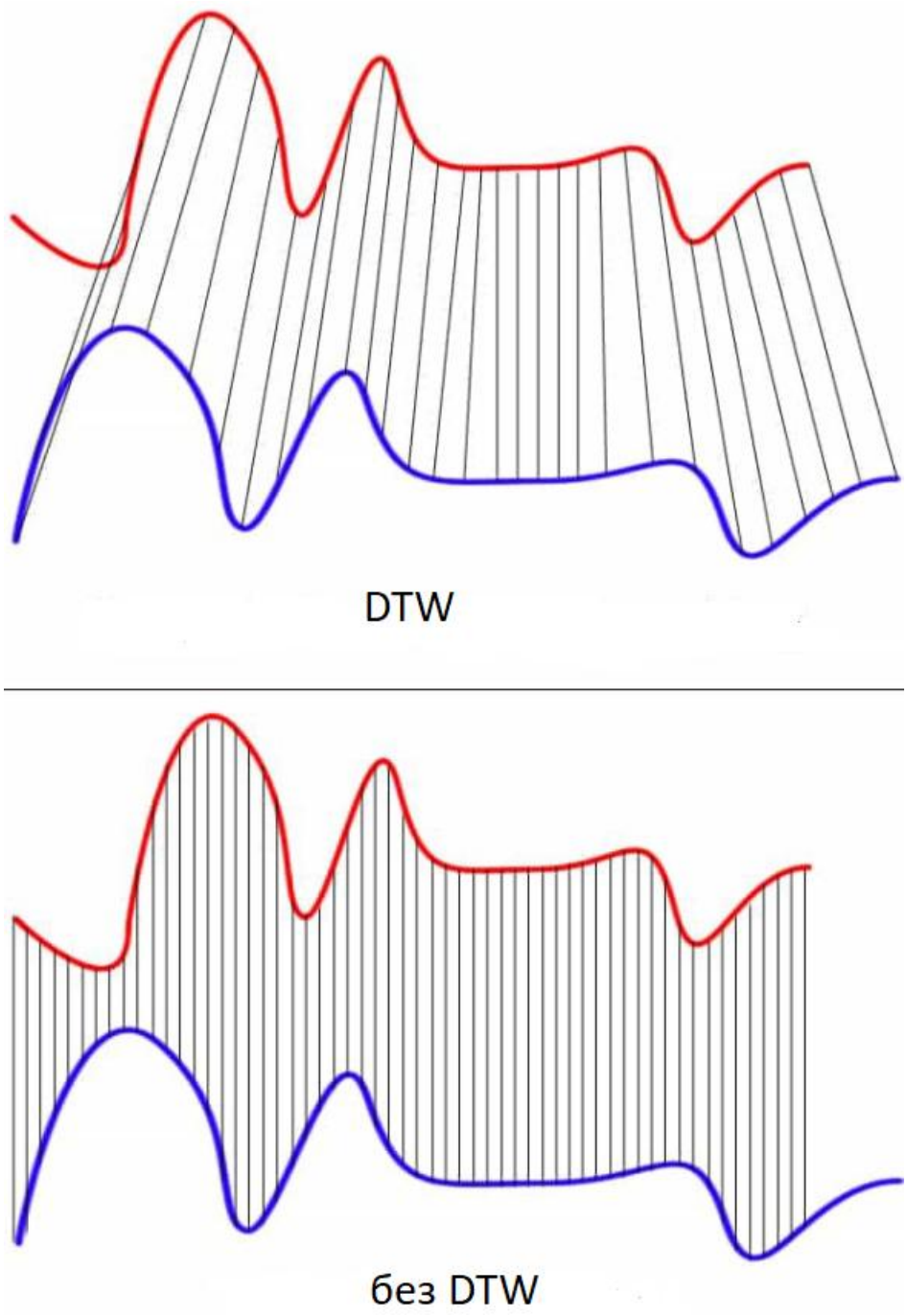

Рисунок 2.1 – Принцип роботи DTW

Це створює деформований «шлях» між X і Y, який вирівнює кожну точку в X до найближчої точки в Y. Шлях є тимчасовим вирівнюванням

часових рядів, що мінімізує евклідову відстань між вирівняними рядами. На рисунку 2.2 є приклад такого деформування часових рядів.

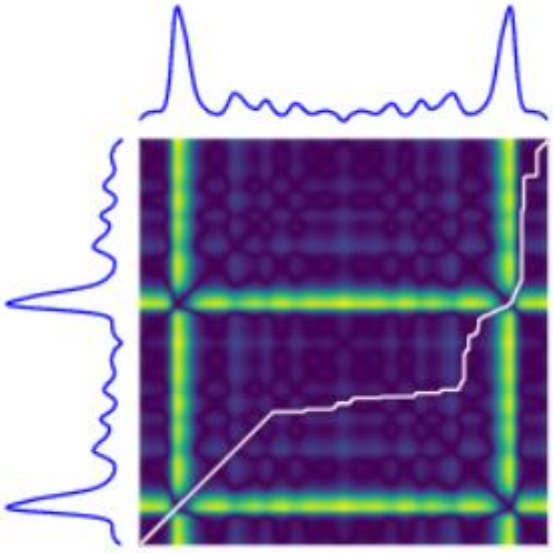

Рисунок 2.2 – Процес деформування ряду

Динамічне деформування часу обчислюється за допомогою динамічного програмування зі складністю O(MN). Алгоритм кластеризації *k*-means можна застосувати до часових рядів з динамічним деформацією часу з наступними модифікаціями.

Динамічна деформація часу (DTW) використовується для збору часових рядів подібних фігур. Центроїди кластерів, або барицентри, обчислюються відносно DTW [23, 24]. Барицентр – це середня послідовність із групи часових рядів у просторі DTW. Алгоритм DTW Barycenter Average (DBA) мінімізує суму квадратів відстані DTW між центром барицентру та серією в кластері. Алгоритм м’якого DTW мінімізує зважену суму відстаней м’якого DTW між барицентром і серією в кластері. Ваги можуть бути налаштовані, але їх сума повинна складати 1.

В результаті центроїди мають середню форму, яка імітує форму членів кластера (рис. 2.3), незалежно від того, де відбуваються тимчасові зрушення між членами кластера.

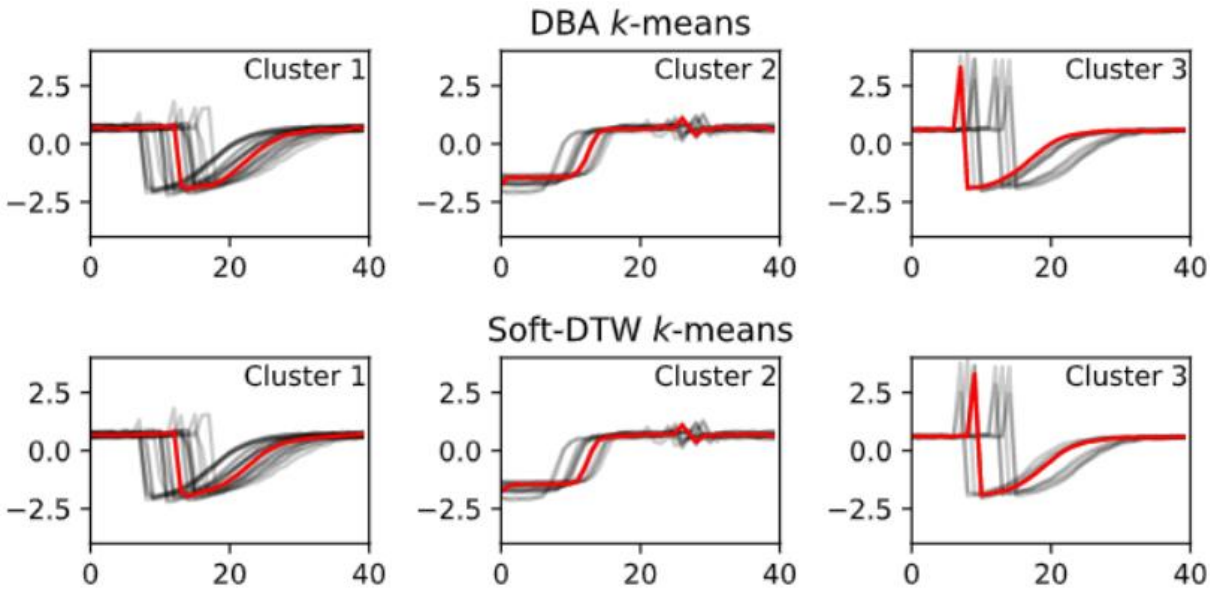

Рисунок 2.3 – Результат роботи алгоритму *k*-means

Також *k*-means потрубеє на вхід параметр який показує кількість кластерів, для знаходження цього параметра можна скористатись методом «ліктя» або силуета.

2.1.1 Метод «ліктя»

При кластеризації методом *k*-середніх кількість кластерів найчастіше оцінюють за допомогою «методу ліктя». Він передбачає багаторазове циклічне виконання алгоритму зі збільшенням кількості кластерів, що вибираються, а також подальшим відкладанням на графіку бала кластеризації, обчисленого як функція від кількості кластерів.

Основним показником методу ліктя є SSE (сума квадратів помилок).

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} | p - m_i |^2 , \qquad (2.3)$$

де C_i – i-й кластер;

p – точка вибірки у C_i;

m_i – центр тяжкості C_i (середнє значення для всіх вибірок у C_i);

SSE – помилка кластеризації всіх вибірок.

Саме SSE відображає якість кластеризації. Основна ідея методу ліктя полягає в тому, що в міру збільшення числа кластерів k розподіл вибірок буде більш точним, а ступінь агрегації кожного кластера поступово збільшуватиметься, тому квадрат помилок і SSE, природно, поступово стануть меншими. Більше того, коли k менше числа істинних кластерів, збільшення k значно збільшить ступінь агрегації кожного кластера, тому зниження SSE буде більшим, і коли k досягне числа істинних кластерів, значення, отримане шляхом збільшення k Ступінь агрегації, повертається швидко стає менше, тому зниження SSE буде різко зменшуватися, а потім поступово згладжуватиметься, оскільки значення k продовжує збільшуватися, що означає, що відносини між SSE і k мають форму коліна, і це коліно Відповідне значення k є кількістю справжніх кластерів у даних. Звичайно, саме тому метод називається методом ліктя [25]. Далі зображено приклад результату роботи метода, на якому видно, що в даному випадку оптимальною кількістю кластерів буде 4, адже це і є той самий згин, який продемонстровано на рисунку 2.4.

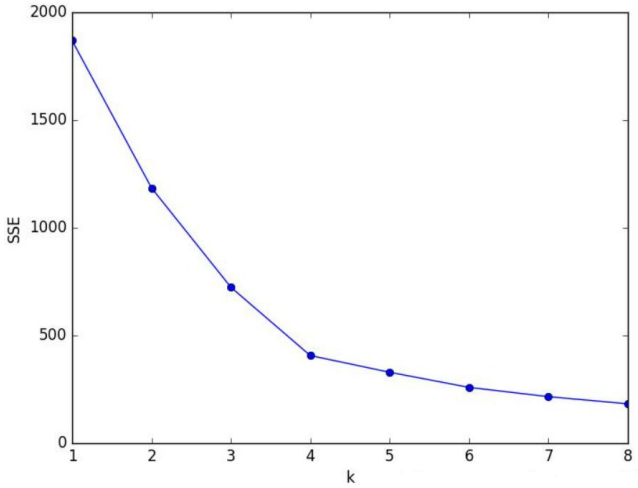

Рисунок 2.4 – Приклад роботи методу «ліктя»

2.1.2 Коефіцієнт «силует»

Метод силуету – це також метод пошуку оптимальної кількості кластерів, інтерпретації та перевірки узгодженості усередині кластерів даних. Метод силуету обчислює коефіцієнти силуету кожної точки, які вимірюють, наскільки точка схожа свій власний кластер проти іншими кластерами. шляхом надання короткого графічного уявлення у тому, наскільки добре кожен об’єкт було класифіковано [26].

Обчисліть коефіцієнти силуету кожної точки і усередніть їх всім зразків, щоб отримати оцінку силуету.

Значення силуету є мірою того, наскільки об’єкт нагадує його власний кластер (зчеплення) проти іншими кластерами (поділ). Значення силуету знаходиться в діапазоні [1, -1], де високе значення вказує, що об’єкт добре відповідає своєму кластеру і погано відповідає сусіднім кластерам. Якщо більшість об’єктів мають високе значення, конфігурація кластеризації підходить. Якщо багато точок мають низьке чи негативне значення, то конфігурації кластеризації може бути занадто багато чи занадто мало кластерів.

Кроки, щоб знайти коефіцієнт силуету i-ї точки:

Крок 1. Обчислити $a(i)$: середня відстань від цієї точки до всіх інших точок у тих самих кластерах.

Крок 2. Обчислити $b(i)$: середня відстань від цієї точки до всіх точок у найближчому кластері до цього кластера.

Крок 3. Обчислити $s(i)$ – коефіцієнт силуету або i-ю точку, використовуючи формулу нижче:

$$s(i) = \frac{b(i) - a(i)}{\max(b(i), a(i))}. \tag{2.4}$$

На рисунку 2.5 зображено приклад роботи метода силуету та в даному випадку оптимальною кількістю кластерів є 4.

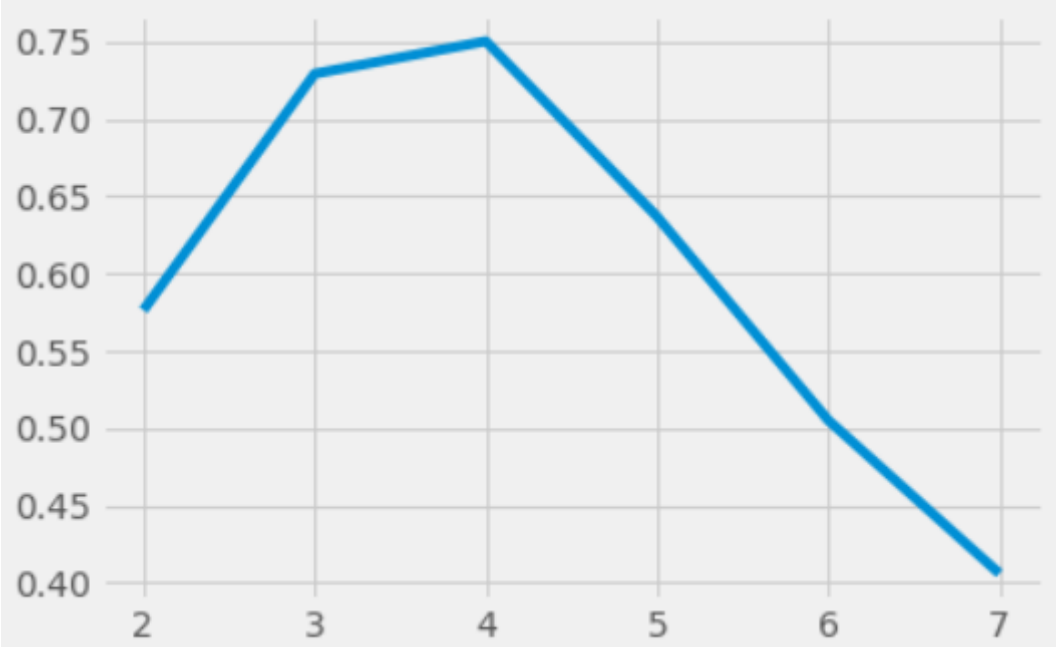

Рисунок 2.5 – Приклад роботи метода «силуета»

2.2 DBSCAN

Щільність – це лише кількість речовини, присутня в одиниці об’єму. Можна легко поширити цю ідею об’єму на вищі або навіть на нижчі виміри. Наприклад, є довільний регіон, та деякі точки даних у цьому регіоні. Також є інший регіон з тією ж областю, і велика кількість даних. Таким чином, з ідеї щільності та із (рис. 2.6) випливає, що щільність першої області більша, ніж другої області. Тому, що в перший регіон входить більше точок даних ніж в другий. DBSCAN використовує цю концепцію щільності для кластеризації набору даних. Тепер, щоб чітко зрозуміти алгоритм DBSCAN, потрібно знати деякі важливі параметри.

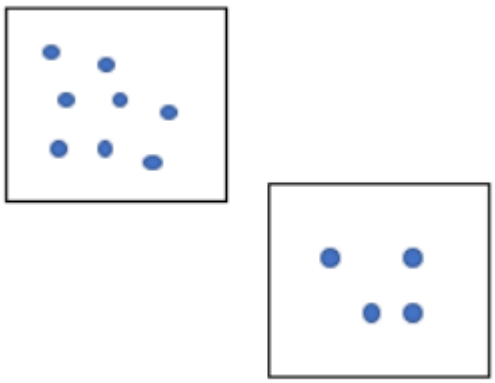

Рисунок 2.6 – Приклад знаходження щільності

Першим важливим параметром алгоритму DBSCAN є ε – це міра сусідства, тобто, припустимо, це довільна точка, навколо якої провели коло, судячи з цього ε – це відстань від центра області до її края.

Таким чином, ε – це лише число, яке представляє радіус кола навколо певної точки, яку алгоритм будемо розглядати навколо центральної точки. Наступний параметр – min_sample. Це поріг найменшої кількості точок, які ми хочемо бачити в околі точки. Припустимо, ми є приймаючи z = 3. Якщо в околі 4 точки, це також задовольнит поріг z = 3. Оскільки цей поріг представляє мінімальну кількість вибірок у околі (рис. 2.7).

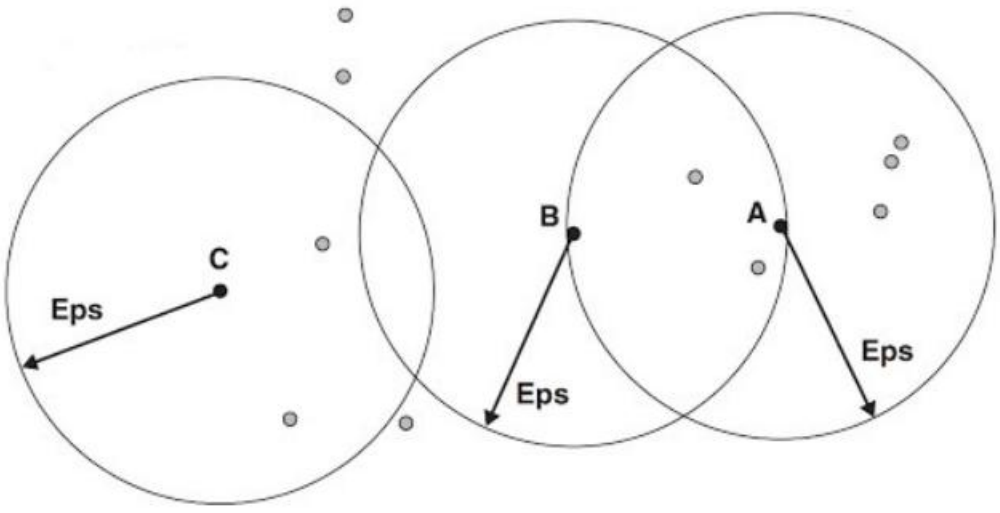

Рисунок 2.7 – Кластеризація точок в методі DBSCAN

Тепер, виходячи з цих двох параметрів, тобто ε та min_samples, треба спочатку класифікувати кожну точку набору даних на три категорії. Вони є основними моментами в розумінні алгоритму, граничні точки або граничні точки точки шуму [27]. Спочатку треба розглянути ці основні моменти. Якщо вказати точку *A* як основною точкою, то вона повинна задовольняти одній умові. Умова полягає в тому, що кількість сусідів має бути більшою або рівною нашому пороговому значенню min_samples або *z*. Якщо встановити $z = 3$, то ця точка задовольняє цій умові. Далі треба розглянути другий тип точок – межі. Якщо визначити довільну точку як граничну, то вона повинна задовольняти деяким наступним умовам. Кількість сусідів має бути меншою від *z*. Точка повинна бути в околі основної точки. Розглянемо таку саму цифру, зазначену вище та визначимо прикордонну точку *B*. Точка має менше, ніж кількість сусідів у її околу, і вона знаходиться в околі іншої базової точки. Таким чином, можна зробити висновок, що, ця точка B є граничною точкою, так як задовольняє всім необхідним умовам. Далі розглянемо точки шуму. Якщо точка не є ні центральною, ні граничною точкою, то її називають точкою шуму. На вищезгаданому рисунку 2.7 точка *C* не є ні центральною, ні граничною точкою. Таким чином, можна зробити висновок, що точка C є точкою шуму, так як не належить до жодного іншого класу точок. Тепер всі точки класифіковані на три кластери, перший крок клгоритму DBSCAN виконаний. Далі треба зрозуміти інше поняття. Краї щільності та точки сполучення щільності.

Припустимо, що є два довільних випадки. Якщо дві точки є сусідами, вони з’єднуються ребром, таке ребро називається ребром густини (рис. 2.8).

Якщо припустити, що кожна точка є основною, та всі вони пов’язані через краї щільності (рис. 2.9).

Рисунок 2.8 – Ребро густини

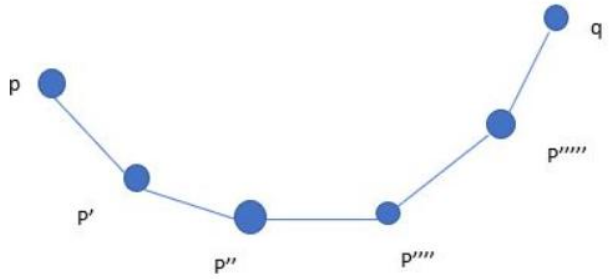

Рисунок 2.9 – Ланцюг точок

В результаті виходить, що *p'* знаходиться в околі *p''*, подібно до інших точок. Але *p* і *q* не є сусідами. Якщо виникає така ситуація, коли дві основні точки з'єднані ребрами щільності, тоді виходить, що *p* і *q* – точки, пов'язані одне з одним щільністю. Тепер розглянемо кроки цього алгоритму:

Крок 1. Класифікація точок.

Крок 2. Знешумлення даних.

Крок 3. Призначення кластера до базової точки.

Крок 4. Розфарбувати всі точки, пов'язані щільністю з основною точкою.

Крок 5. Розфарбувати граничні точки відповідно до найближчої основної точки.

Перший крок вже описано вище. Друге – це просто усунення точок шуму. Третій крок, наприклад, основній точці призначити кластер червоного

кольору. На четвертому кроці треба пофарбувати всі точки, пов’язані щільністю з обраною основною точкою, яка була розфарбована на третьому кроці в червоний колір. Проте, не можна зафарбовувати граничні точки до закінчення кластеризації. Треба повторити третій і четвертий кроки для кожної незафарбованої основної точки і після цього розфарбувати границі. Так, як алгоритм DBSCAN дуже чутливий до своїх початкових параметрів, треба ретельно визначати визначити ε та z [28].

Таким чином, якщо навіть трохи змінити значення ε і z, то алгоритм може дати кардинально різні результати. Це є одним із мінусів цього алгоритму. Існує емпіричне правило, якщо є велика кількість прикладів, ви можна обрати z у порядку розмірності вихідного набору даних. Наприклад, якщо, мова іде про 10 розмірностей, то краще вибрати значення z, близьке до 10, наприклад 12 або 15. Припустимо, ви вибрали $z = 5$. Алгоритм знайде відстань 5-го сусіда від кожної точки даних. Таким чином, в результаті, буде масив відстаней, а i-й запис у цьому масиві відображатиме відстань 5-го сусіда i-ої точки даних. І тоді треба сортувати цей масив відстаней і будувати його таким чином – на осі у буде лише відстань, а на осі x-індекс (i). В ідеалі треба отримати такий графік (рис. 2.10).

Як тільки сортування виконане, модна зробити висновок, що, оскільки індекс буде збільшуватись, відстань до 5-ї точки даних від цієї точки також збільшиться. Також є висова ймовірність знайти цю закономірність методом ліктя. Можна продемонструвати це горизонтальною лінією, і горизонтальна лінія в певному місці виріже вісь у, далі значення яке було отримане на перетині моджна використовувати як ε. Але в реальному світі на реальних даних такий гладкий вигин форми ліктя отримати не завжди вдасться. Але це хитрість, яку можна застосувати в деяких випадках.

DBSCAN дуже ефективний в усуненні шуму [29 ,30]. У прикладі, який розглядається протягом другого розділу, точки класифікувались за трьома категоріями, і була категорія точок шуму.

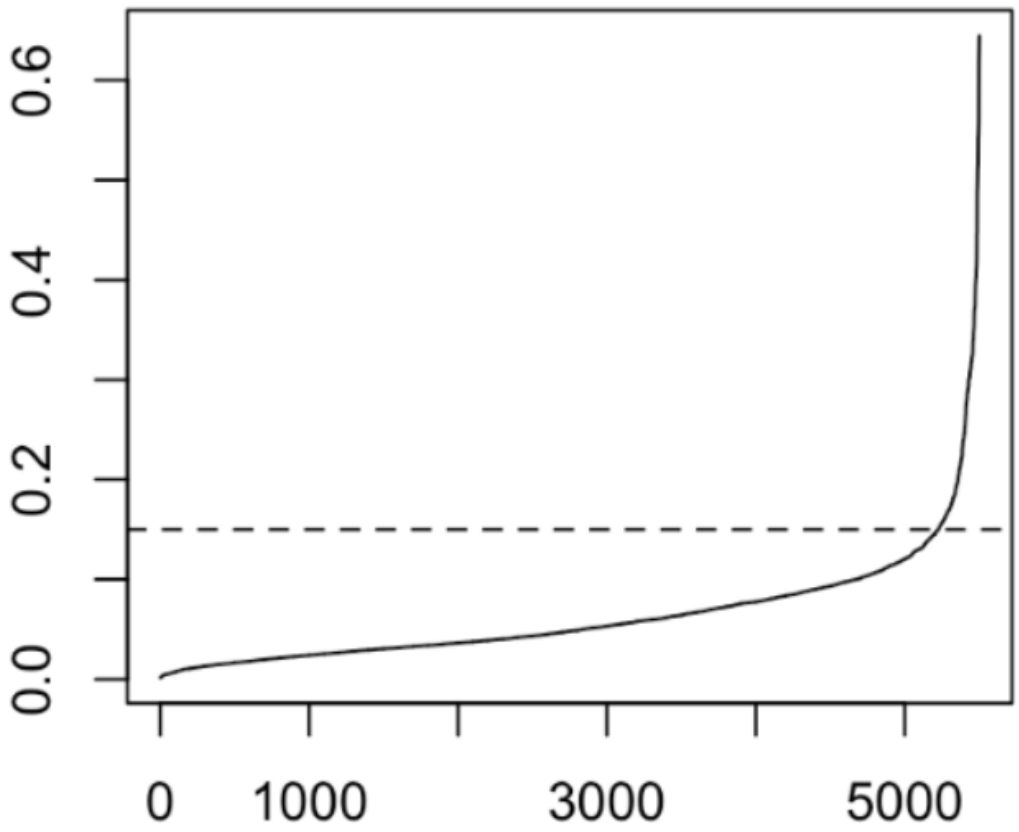

Рисунок 2.10 – Сортування точок DBSCAN

Таким чином, цей алгоритм можна дуже успішно застосувати до навантажених наборів даних. І останній момент – DBSCAN не може дуже добре обробляти дані вищих розмірів. Це помилка багатьох алгоритмів кластеризації. Зі збільшенням розмірності доводиться шукати більший обсяг, щоб знайти таку ж кількість сусідів. Таким чином, подібність між точками зменшується. Це призведе до помилок кластеризації [31].

Тепер можна перейти до розділу з реалізацією і демонстрацією алгоритмів.

3 РЕАЛІЗАЦІЯ МЕТОДІВ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи були розроблені алгоритми для демонстрації кластеризації часових рядів. Для реалізації було обране середовище Google Colaboratory. Google Colaboratory використовує одну з найзручніших мов для програмування та аналізу даних – Python [32].

Google Colaboratory надає доступ до швидких GPU та TPU, тобто всі обчислення виконуються на сервері. Google Colaboratory – це хмарний сервіс, на основі Jupyter Notebook, який, в свою чергу, є зручним середовищем для проведення різних досліджень та аналізу працює, з мовою програмування Python. Python – об’єктно-орієнтована мова загального призначення, який розроблений з метою підвищення продуктивності програміста. Із плюсів можна виділити:

– низький поріг входження. Синтаксис Python більш зрозумілий для новачка;

– логічний, лаконічний і зрозумілий. У порівнянні з багатьма іншими мовами Python має легкочитаємий синтаксис. Кроссплатформовий: підходить для різних платформ: і Linux, і Windows;

– широке застосування. Використовується для розробки веб-додатків, ігор, зручний для автоматизації, математичних обчислень, машинного навчання, в області інтернету речей. Існує реалізація під назвою Micro Python, оптимізована для запуску на мікроконтролерах (можна писати інструкції, логіку взаємодії пристроїв, організовувати зв’язок, реалізовувати розумний будинок);

– у світі Python багато якісних бібліотек, які й будуть використовуватись для реалізації алгоритмів, так що не потрібно винаходити велосипед, якщо треба терміново вирішити якесь завдання;

– python відрізняється суворою вимогою до написання коду (вимагає відступи), що є перевагою, за моїми спостереженнями. Спочатку мова сприяє писати код організовано і красиво.

Звичайно, у сторони дві медалі, і якщо говорити про мінуси, то Python - це мова з динамічною типізацією. З одного боку код простіше і швидше писати, але продуктивність поступається таким компільовані мов, як C ++ і Golang [33]. Але для більшості завдань: для веб-розробки, для скриптів, прототипирования, машинного навчання та роботи з великими даними, – один з кращих мов. Хоча, якщо реалізувати алгоритми суперпіксельної сегментації зображень наприклад на C++, то показники швидкодії будуть збільшуватись в рази.

Середа розробки Google Colaboratory ще більше посилює плюси дослідження алгоритмів на мові програмування Python [34].

3.2 Додаткові бібліотеки

Для роботи з масивами даних була використана бібліотека NumPy. NumPy – це фундаментальний пакет для наукових обчислень на Python. Це бібліотека Python, яка надає багатовимірний об’єкт масиву, різні похідні об’єкти (такі як замасковані масиви та матриці), а також набір підпрограм для швидких операцій з масивами, включаючи математичні, логічні, маніпуляції з формою, сортування, вибір, введення/виводу, дискретні перетворення Фур’є, базова лінійна алгебра, основні статистичні операції, випадкове моделювання та багато іншого [35, 36]. В основі пакету NumPy лежить об’єкт ndarray. Це інкапсулює *n*-вимірні масиви однорідних типів даних, при цьому багато операцій виконуються в скомпільованому коді для підвищення продуктивності.

Існує кілька важливих відмінностей між масивами NumPy і стандартними послідовностями Python: Масиви NumPy мають фіксований

розмір під час створення, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру ndarray створить новий масив і видалить оригінал. Усі елементи в масиві NumPy повинні мати один тип даних і, таким чином, мати однаковий розмір пам’яті. Виняток: можна мати масиви об’єктів (Python, включаючи NumPy), що дозволяє використовувати масиви елементів різного розміру.

Масиви NumPy полегшують розширені математичні та інші типи операцій над великою кількістю даних [37]. Як правило, такі операції виконуються ефективніше і з меншою кількістю коду, ніж це можливо з використанням вбудованих послідовностей Python. Все більша кількість науково-математичних пакетів на основі Python використовує масиви NumPy; хоча вони зазвичай підтримують введення послідовності Python, вони перетворюють такі вхідні дані в масиви NumPy перед обробкою, і вони часто виводять масиви NumPy.

Іншими словами, для ефективного використання значної частини (можливо, навіть більшості) сучасного науково-математичного програмного забезпечення на основі Python, недостатньо просто знати, як використовувати вбудовані в Python типи послідовностей – потрібно також знати, як використовувати масиви NumPy [38].

Для зручного завантаження даних та отримання ознак була використана бібліотека Pandas – програмна бібліотека Python для обробки та аналізу даних. Робота Pandas з даними будується поверх бібліотеки NumPy, що є інструментом нижчого рівня. Надає спеціальні структури даних та операції для маніпулювання числовими таблицями та тимчасовими рядами. Назва бібліотеки походить від економетричного терміна «панельні дані», який використовується для опису багатовимірних структурованих наборів інформації.

Основними можливостями бібліотеки є об’єкт DataFrame для маніпулювання індексованими масивами двовимірних даних, інструменти для обміну даними між структурами пам’яті та файлами різних форматів,

вбудовані засоби суміщення даних та способи обробки відсутньої інформації, переформатування наборів даних, зокрема створення зведених таблиць, зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних, вставка та видалення стовпців даних, можливості угруповання дозволяють виконувати триетапні операції на кшталт «поділ, зміна, об'єднання», злиття та об'єднання наборів даних, ієрархічне індексування дозволяє працювати з даними високої розмірності у структурах меншої розмірності, робота з тимчасовими рядами: формування тимчасових періодів та зміна інтервалів, тощо.

Бібліотека оптимізована для високої продуктивності, найважливіші частини коду написані на Cython та C [39].

Для отримання деяких функція для самого процесу кластеризації була використана бібліотека Scikit-learn. Scikit-learn – одна з найбільш широко використовуваних бібліотек Python для Data Science та Machine Learning. Вона дозволяє виконувати безліч операцій та надає безліч алгоритмів. Scikit-learn також пропонує чудову документацію про свої класи, методи та функції, а також опис використовуваних алгоритмів. Scikit-Learn підтримує: попередню обробку даних; зменшення розмірності; вибір моделі; регресії; класифікації; кластерний аналіз [40].

Вона також надає кілька наборів даних, які можна використовувати для тестування моделей.

Scikit-learn не реалізує все, що пов'язане із машинним навчанням. Наприклад, вона не має комплексної підтримки для: нейронних мереж; самоорганізуються карт (мереж Кохонена); навчання асоціативних правил; навчання з підкріпленням (reinforcement learning).

Scikit-learn заснований на NumPy та SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати Scikit-learn.

Scikit-learn – це пакет з відкритим вихідним кодом. Як і більшість матеріалів з екосистеми Python, вона безкоштовна навіть для комерційного використання.

3.3 Кластеризація на прикладі бронювання готелів

Для демонстрації алгоритмів було використано набір даних з бронювання готелів [41], в якому є такі поля, як: daily_booking – кількість бронювань за день, days_in_waiting_list – кількість днів у листі очікування, total_of_special_request – кількість спеціальних запитів, adr – статистичний показник середньої вартості проданого номера, booking_changes – кількість змін здійснених вже після бронювання, previous_cancellations – попередні відміни бронювання, stays_in_weekend_nights – кількість постояльців на вихідних, stays_in_week_nights – кількість постояльців в будні дні. Метою кластеризації є знаходження кореляції між існуючими даними для того, щоб отримані дані можна було використати у майбутньому для підвищення кількості бронювань номерів.

Також для кластеризації були отримані деякі параметри такі, як середнє значення, мінімальне, максимальне і т.д.

Першим алгоритмом був класичний *k*-means, який в свою чергу потребує в якості вхідного параметра кількість кластерів. В якості вхідних параметрів даний алгоритм потребує n_clusters – кількість кінцевих кластерів, max_iter – максимальна кількість ітерацій алгоритму, metric – метрика, яка буде використовуватись при обчисленні евклідова, DTW або soft-DTW.

Для того щоб знайти кількість кластерів можна скористатись методом «ліктя» або силуету.

Для вибору оптимальної кількості кластерів було обрано метод силуету, який показав найвищий пік на 3 кластерах (рис. 3.1).

В результаті виконання кластеризації методом *k*-середніх були отримані такі кластери (рис. 3.2-3.4).

Далі була виконана кластеризація методом DBSCAN, який потребує вхідний параметр ε, в якості максимальної відстані між двома кластерами. В данному випадку було обрано 0,65. min_samples – мінімальна кількість точок

(або загальна вага) в околі для точки, яка вважається основною. Сюди входить і сама точка. Metric – параметр який є метрикою в процесі обчислення кластерів. Та після виконання кластеризації були отримані такі результати (рис. 3.5-3.7).

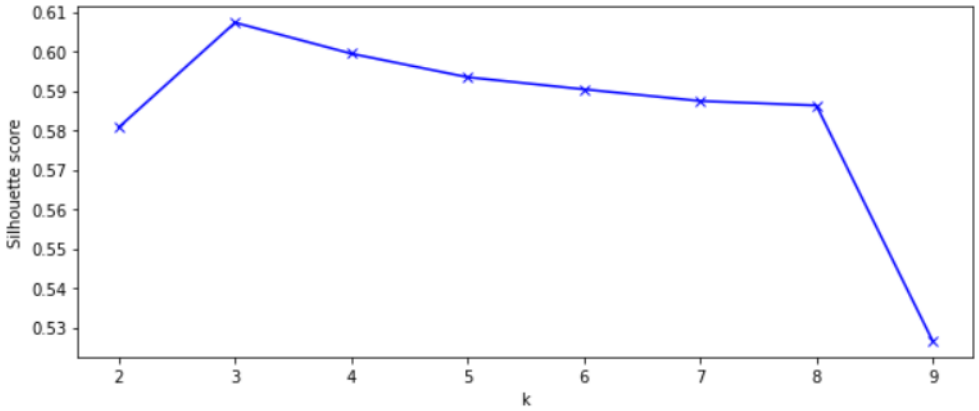

Рисунок 3.1 – Коефіцієнт ситуета

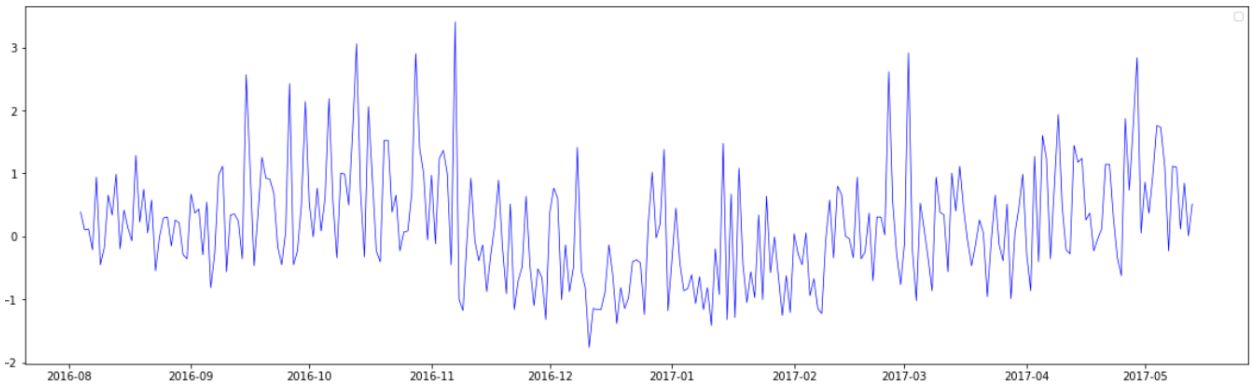

Рисунок 3.2 – Часовий ряд daily_booking

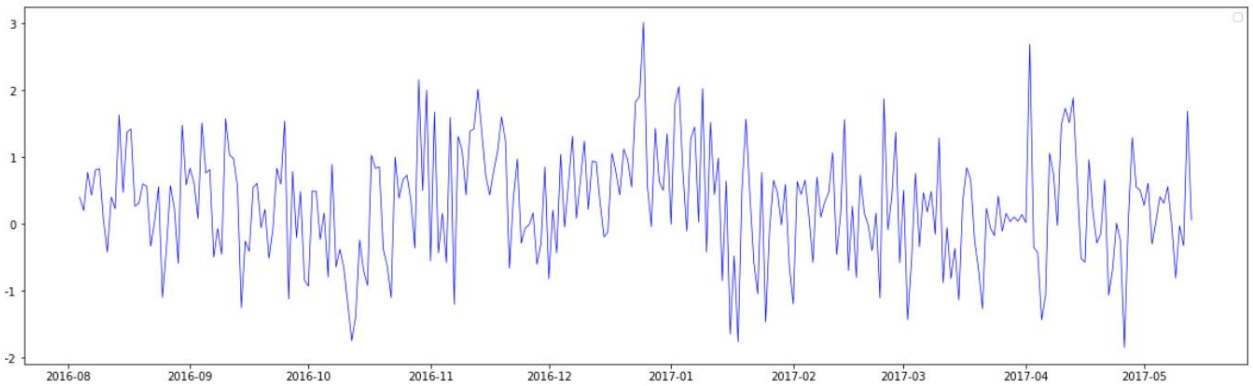

Рисунок 3.3 – Часовий ряд adr

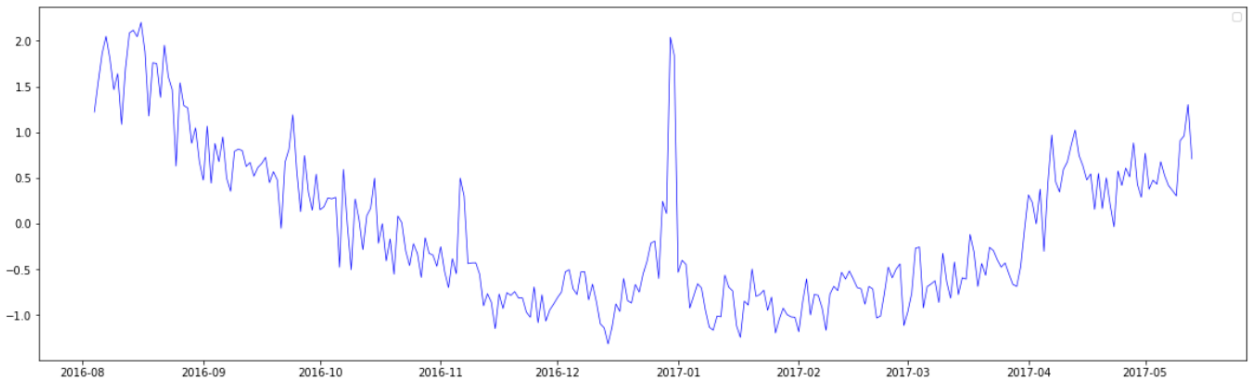

Рисунок 3.4 – Часовий ряд previous_cancellations

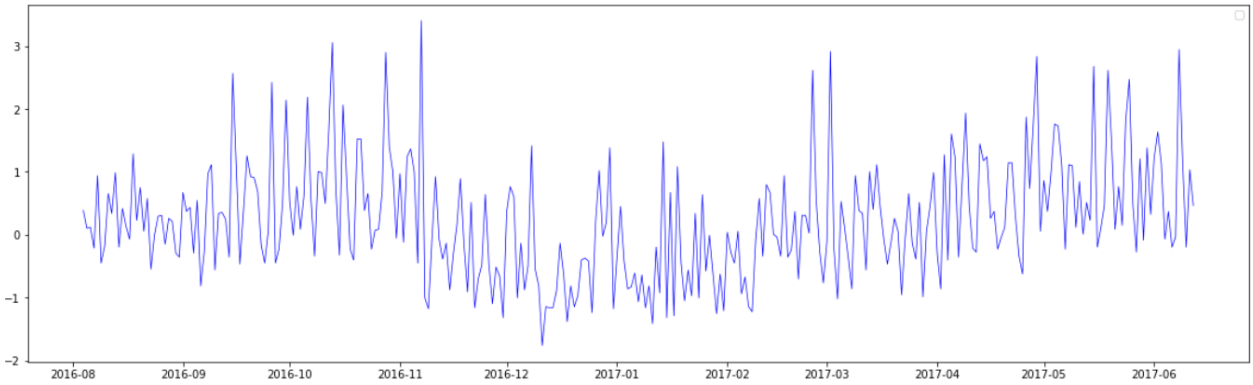

Рисунок 3.5 – Часовий ряд daily_booking

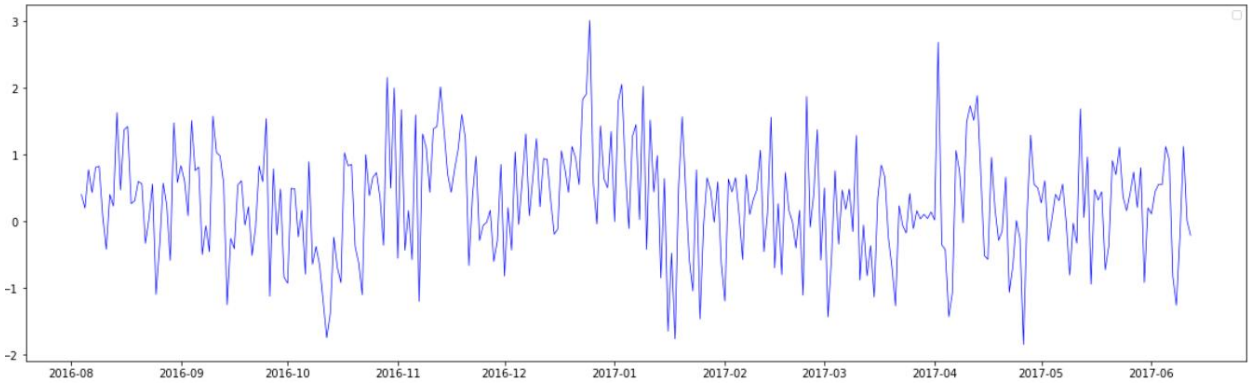

Рисунок 3.6 – Часовий ряд adr

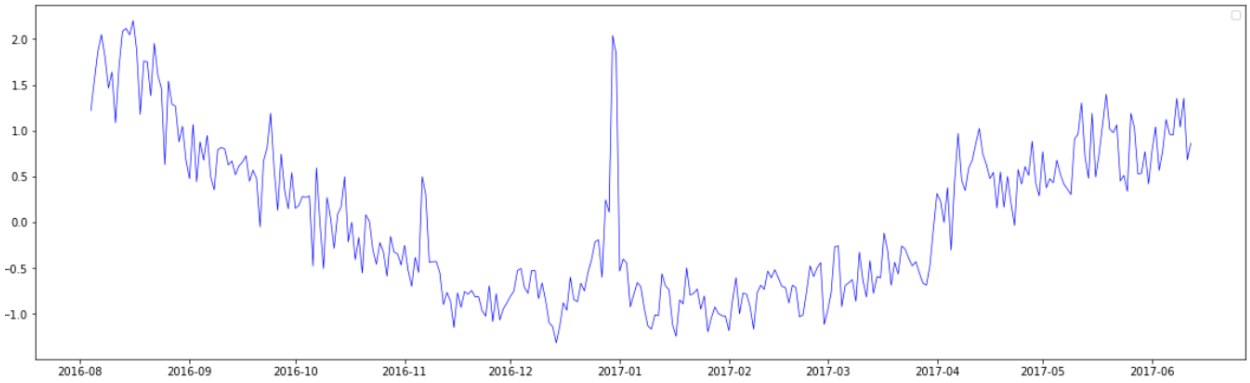

Рисунок 3.7 – Часовий ряд previous_cancellations

Обидва алгоритми кластеризували дані схожим чином, проте для кожного можна виділити такі особливості, наприклад, DBSCAN краще використовувати тоді, коли використовується дуже велика вибірка даних. Якщо під рукою оптимізована та розпаралена реалізація.

Наперед відома функція близькості, симетрична, бажано, не дуже складна. KD-Tree оптимізація часто працює тільки з евклідовою відстанню.

Також якщо очікується набір даних нестандартної форми: вкладені та аномальні кластери або кластери малої розмірності.

Щільність меж між згустками менше щільності найменш щільного кластера. Краще якщо кластери зовсім відокремлені один від одного.

Складність елементів датасета значення не має. Однак їх має бути достатньо, щоб не виникало сильних розривів у густині. Кількість елементів у кластері може змінюватись як завгодно. Кількість викидів значення не має (в розумних межах), якщо вони розсіяні за великим обсягом. Кількість кластерів значення немає.

Метод *k*-means в свою чергу має деякі обмеження, наприклад, такі як: необхідно заздалегідь знати кількість кластерів.

Алгоритм дуже чутливий до вибору початкових центрів кластерів. Класичний варіант має на увазі випадковий вибір кластерів, що дуже часто було джерелом похибки. Як варіант рішення необхідно проводити дослідження об’єкта для більш точного визначення центрів початкових

кластерів. У моєму випадку на початковому етапі пропонується приймати як центи найвіддаленіші точки кластерів, не справляється із завданням, коли об’єкт належить до різних кластерів однаковою мірою чи належить жодному.

Також була побудована та продемонстрована на рисунку 3.8 матриця кореляції, яка показала деяку залежність параметрів датасета.

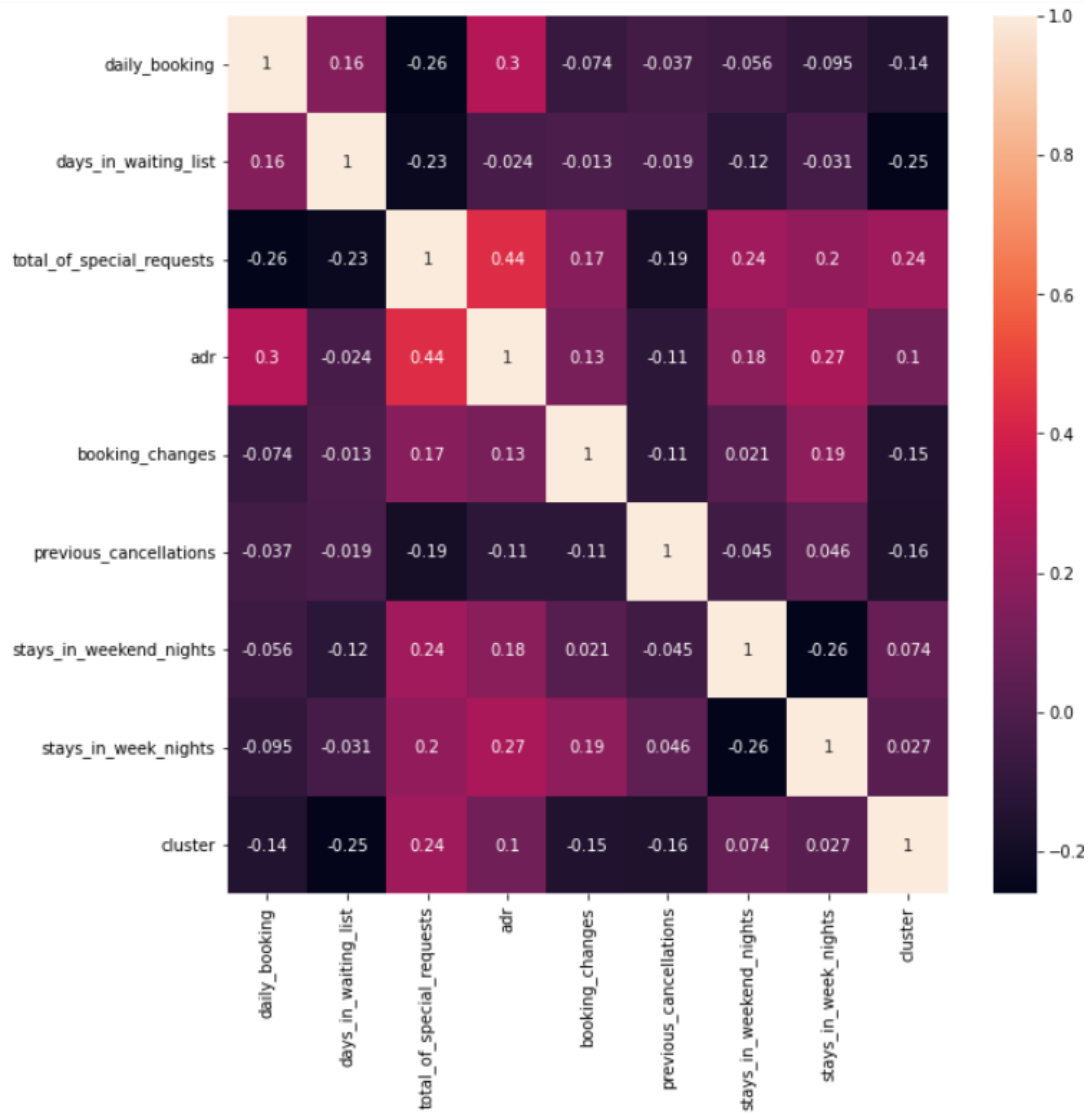

Рисунок 3.8 – Мартиця кореляції

Із матриці кореляції можна зробити висновок, що неможливо виділити параметри з сильною кореляцією, так, як максимальним коефіцієнтом кореляції є 0,44, тому було прийнято рішення розглянути параметри з

найбільшою кореляцією.Таким чином, в питанні бронювання готельних номерів найбільшу залежність має adr (статистичний показник середньої вартості проданого номера на день) до total_of_special_requests (відображає кількість спеціальних запитів клієнта таких, кондиціонер у номері, внутрішня паркова і т.д), та adr до daily_booking (кількість днів на яку клієнт бронює номер). Аналіз даних показав, що ціна напряму залежить від того, наскільки часто номер буде забронюваним, так само як і кількість спеціальних запросів клієнта, тобто можна зробити висновок, що для підвищення кількості бронювання номерів треба робити меншу ціну за номер, та за цю ціну давати найбільшу кількість зручностей для клієнта.

3.4 Кластеризація на прикладі акцій компаній

Другим прикладом була реалізована кластеризація акцій компаній з різних галузей за останній місяць. Тобто метою кластеризації є знаходження з усієї вибірки даних різних компаній тих, які за останній місяць зросли, та тих, які за останній місяць впали. Така інформація може знадобитись в майбутньому для трейдерів, та людей, котрі займаються аналізом акційних ринків різних коспаній.

Для цього самі дані були завантажені з сервісу yahoo finance за допомогою відкритого API [42]. Сам процес кластеризації був виконаний за допомогою алгоритму *k*-means. Коефіцієнт силуету показав (рис. 3.9) що для цього набору даних краще 2 або 3 кластери, було обрано 2, так як це будуть ті кластери з акціями які падали або росли.

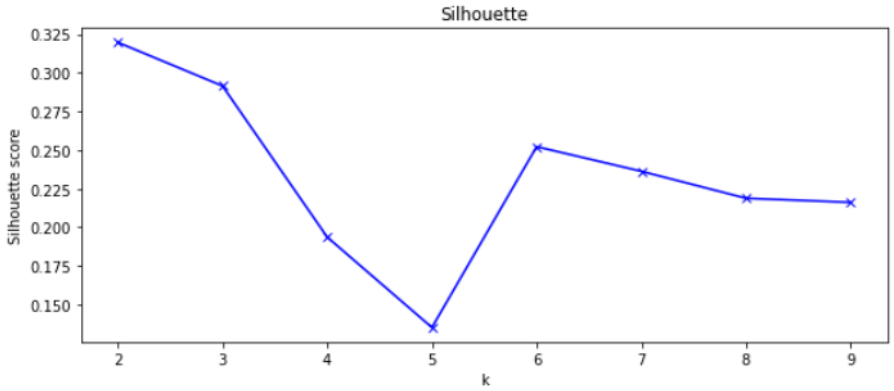

Рисунок 3.9 – Коефіцієнт «силует»

Після кластеризації та побудування цетроїдів було отримано такий часовий ряд з якого можна зробити висновок, що акції тих компаній, що належать до 0 кластера за останній місяць зменшилися, тих хто з 1 кластера за останній місяць зросли (рис. 3.10).

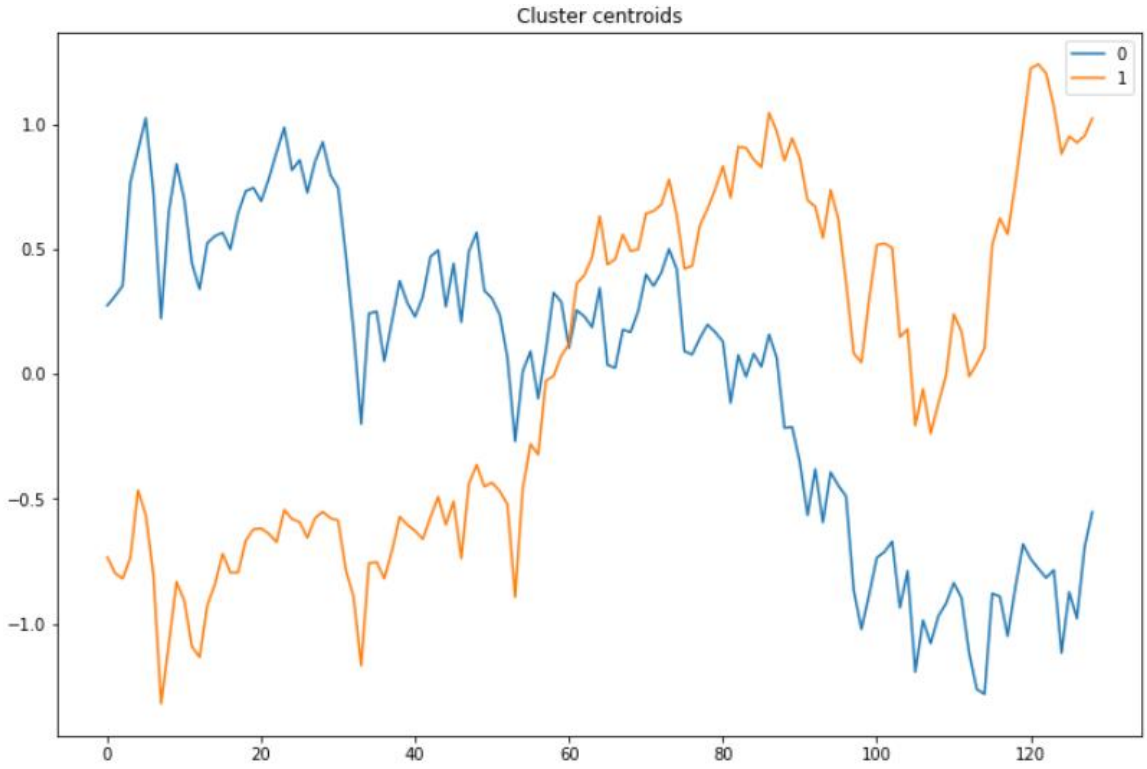

Рисунок 3.10 – Отримані кластери K-means

Далі продемонстровані галузі до яких належать спостереження з кожного кластера (рис. 3.11) з яких можна зробити висновок, що за останній

місяць найбільше за все зменшились акції тих компаній, які відповідають таким галузям як комунікації торгівля та здоров'я. В той же час зросли акції тих компаній, які відповідають галузям індустріальним, технологічним, фінансовим та знову ж таки галузям, які займаються переважно здоров'ям із цього можна зробити припущення що галузь здоров'я так неоднозначно поводиться останнім часом через епідемію та вакцини для цього можна проаналізувати, які саме компанії до якого кластера потрапили.

cluster	sector	
0	Communication Services	3
	Consumer Cyclical	3
	Healthcare	3
	Consumer Defensive	2
	Financial Services	2
	Technology	2
	Energy	1
	Industrials	1
	Real Estate	1
1	Industrials	7
	Technology	6
	Financial Services	4
	Healthcare	4
	Utilities	3
	Communication Services	2
	Consumer Cyclical	2
	Consumer Defensive	2
	Real Estate	2

Рисунок 3.11 – Галузі компаній

Далі наведені приклади, компаній які потрапили до 1 кластера (рис. 3.12). У всіх простежується схожа тенденція до зменшення ціни за

акцію за останній місяць окрім Cerner Corporation, акції якої почали зростати в останні дні місяця, але в цілому, тенденція схожа.

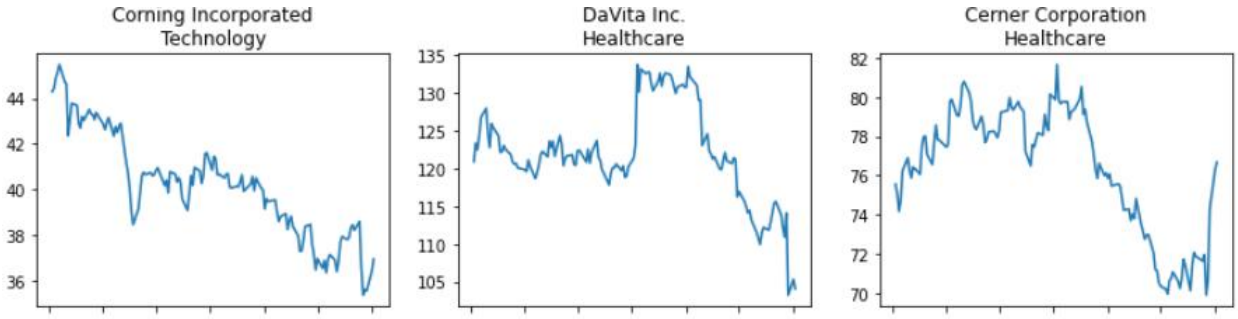

Рисунок 3.12 – Акції компаній 1 кластера

Далі наведені приклади компаній, які потрапили до 2 кластера (рис. 3.13). В даному випадку також у всіх простежується однакова тенденція до зросту ціни за акцію.

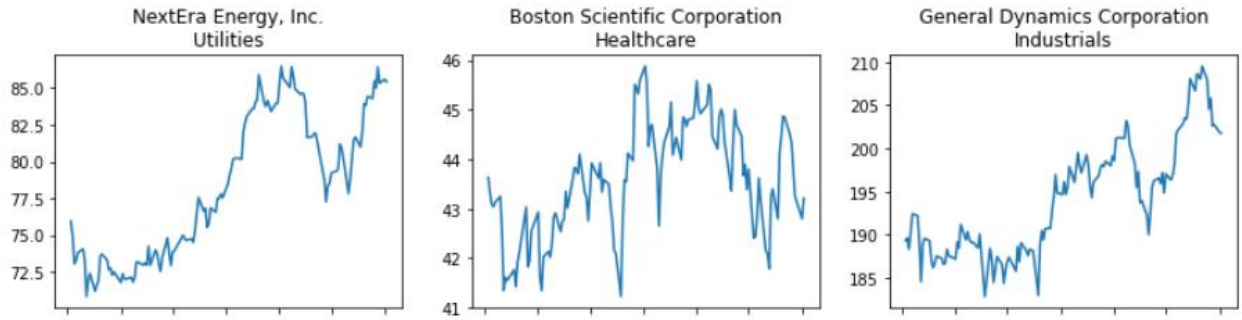

Рисунок 3.13 – Акції компаній 2 кластера

DBSCAN, в свою чергу, показав схожі результати при $\varepsilon = 0{,}8$. Параметр ε було обрано таким чином, щоб утворилось 2 кластери для зручності порівняння з першим алгоритмом. В результаті були отримані 2 кластери, які так само, як і в першому випадку є відображенням тих акцій, які за останній місяць зростали або падали в ціні (рис. 3.14). Аналіз галузей демонструє те, що до другого кластера (до зростаючих акцій) належать переважно акції компаній із технологічної галузі фінансової та медицинської. Рисунок 3.15 демонструє розбиття кластерів по галузям. Далі (рис. 3.16) продемонстровані

представники першого кластера, а (рис. 3.17) продемонстровані представники другого кластера із який акції компанії Moderna, Inc. в останній момент місяця різко втратили ціну, проте все одно потрапили до цього кластера, через те, що в цілому простежувалась тенденція до росту.

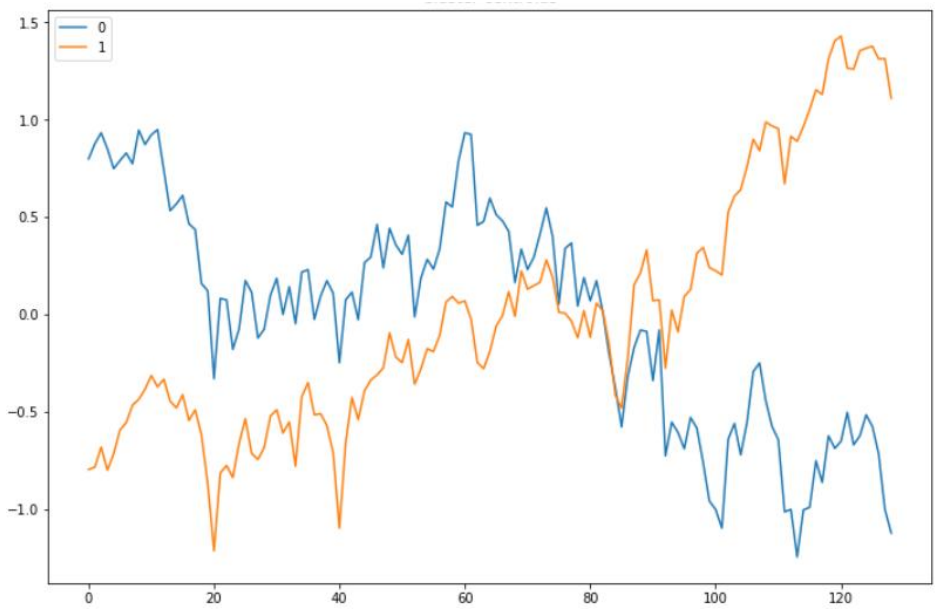

Рисунок 3.14 – Отримані кластери DBSCAN

cluster	sector	
0	Consumer Defensive	3
	Healthcare	3
	Consumer Cyclical	2
	Utilities	2
	Financial Services	1
	Industrials	1
	Technology	1
1	Technology	9
	Financial Services	7
	Healthcare	5
	Industrials	5
	Consumer Cyclical	4
	Energy	3
	Real Estate	2
	Basic Materials	1
	Consumer Defensive	1

Рисунок 3.15 – Галузі компаній

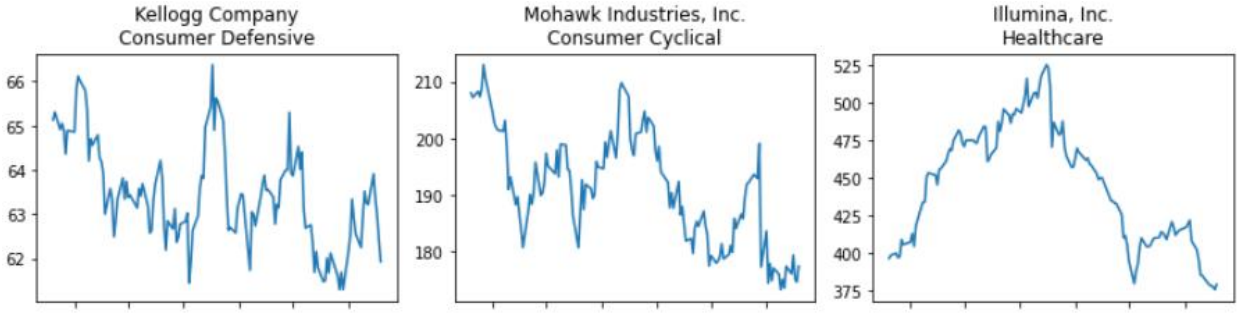

Рисунок 3.16 – Галузі компаній

Рисунок 3.17 – Галузі компаній

Таким чином, обидва алгоритми, в цілому, показали схожі результати кластеризації.

Тобто отримані після кластеризації результати можна використовувати для подальшого аналізу, або для вирішення задачі передбачення, також можна в цілому проаналізувати ринок.

ВИСНОВКИ

У рамках кваліфікаційної роботи було проведено огляд відомих методів кластеризації часових рядів. Аналіз розглянутих методів дозволяє більш точно підібрати необхідний метод і тим самим значно покращити ефективність кластеризації, що, у свою чергу, значно підвищує ефективність подальшої роботи з отриманими даними. Були розглянуті різні підходи до кластеризації часових рядів.

Для експериментальних досліджень було обрано методи DBSCAN та Kmeans. Експериментальні дослідження було проведено завдяки середовищу Google Colaboratory, всі алгоритми були реалізовані мовою програмування Python.

Під час проведення експериментальних досліджень виявилося, що жоден з алгоритмів не володіє явниими перевагами по всім параметрам перед іншими алгоритмами. Кожен із них має свої сильні та слабкі сторони. В ході дослідження була проведена кластеризація таких часових рядів, як дані бронювань готелів, та ціни на акції найпопулярніших компаній за останній місяць.

Результати роботи було апробовано на конференції IT&I 2021 [43].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering–a decade review. *Information Systems*, *53*, 16-38.
2. Kalpakis, K., Gada, D., & Puttagunta, V. (2001, November). Distance measures for effective clustering of ARIMA time-series. In *Proceedings 2001 IEEE international conference on data mining* (pp. 273-280). IEEE.
3. Peker, K. A. (2005, April). Subsequence time series (sts) clustering techniques for meaningful pattern discovery. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2005.* (pp. 360-365). IEEE.
4. Junkui, L., Yuanzhen, W., & Xinping, L. (2006, December). LB HUST: A symmetrical boundary distance for clustering time series. In *9th International Conference on Information Technology (ICIT'06)* (pp. 203-208). IEEE.
5. Serra, J., & Arcos, J. L. (2012, September). A competitive measure to assess the similarity between two time series. In *International Conference on Case-Based Reasoning* (pp. 414-427). Springer, Berlin, Heidelberg.
6. Marteau, P. F. (2008). Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, *31*(2), 306-318.
7. Batista, G. E., Wang, X., & Keogh, E. J. (2011, April). A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM international conference on data mining* (pp. 699-710). Society for Industrial and Applied Mathematics.
8. Vinh, V. T., & Anh, D. T. (2015, October). Compression rate distance measure for time series. In *2015 IEEE international conference on data science and advanced analytics (DSAA)* (pp. 1-10). IEEE.
9. Sitaram, D., Dalwani, A., Narang, A., Das, M., & Auradkar, P. (2015, May). A measure of similarity of time series containing missing data using the

mahalanobis distance. In *2015 second international conference on advances in computing and communication engineering* (pp. 622-627). IEEE.

10. Paparrizos, J., & Gravano, L. (2015, May). k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 1855-1870).

11. Ferreira, L. N., & Zhao, L. (2016). Time series clustering via community detection in networks. *Information Sciences*, *326*, 227-242.

12. Zhu, Z., Peng, Q., & Guan, X. (2016, December). A time series clustering method based on hypergraph partitioning. In *2016 International Conference on Progress in Informatics and Computing (PIC)* (pp. 27-31). IEEE.

13. Jicheng, S., & Weike, L. (2016, December). Clustering algorithm for time series based on peak interval. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 410-414). IEEE.

14. Rani, S. (2016, March). Modified hierarchical clustering algorithm for time series data. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 4036-4040). IEEE.

15. Rodrigues, P. P., Gama, J., & Pedroso, J. (2008). Hierarchical clustering of time-series data streams. *IEEE transactions on knowledge and data engineering*, *20*(5), 615-627.

16. You, S. Y., Wang, Y. D., Luo, L. K., & Peng, H. (2016, August). Finding the clusters with potential value in financial time series based on agglomerative hierarchical clustering. In *2016 11th International Conference on Computer Science & Education (ICCSE)* (pp. 77-81). IEEE.

17. Cinar, G. T., & Principe, J. C. (2014, May). Clustering of time series using a hierarchical linear dynamical system. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6741-6745). IEEE.

18. Kirichenko, L. O., Kobitskaya, Y. A., & Habacheva, A. Y. (2014). Comparative analysis of the complexity of chaotic and stochastic time series.

19. Daradkeh, Y. I., Kirichenko, L., & Radivilova, T. (2018). Development of QoS methods in the information networks with fractal traffic. *International Journal of Electronics and Telecommunications*, *64*.

20. Pandas. URL: https://ru.wikipedia.org/wiki/Pandas (дата звернення: 01.11.2021).

21. Vitalii, B., Kirichenko, L., & Radivilova, T. (2018, May). Classification of multifractal time series by decision tree methods. In *14th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. CEUR Workshop Proceedings* (Vol. 2105, pp. 457-460).

22. Kirichenko, L., Radivilova, T., & Zinkevich, I. (2017, September). Comparative analysis of conversion series forecasting in e-commerce tasks. In *Conference on Computer Science and Information Technologies* (pp. 230-242). Springer, Cham.

23. Kirichenko, L., Radivilova, T., & Bulakh, V. (2018, October). Classification of fractal time series using recurrence plots. In *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)* (pp. 719-724). IEEE.

24. Ivanisenko, I., Kirichenko, L., & Radivilova, T. (2016, August). Investigation of multifractal properties of additive data stream. In *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)* (pp. 305-308). IEEE.

25. Kirichenko, L., Radivilova, T., & Deineko, Z. (2011). Comparative analysis for estimating of the Hurst exponent for stationary and nonstationary time series. *Information Technologies & Knowledge*, *5*(1), 371-388.

26. Kirichenko, L. O., & Habacheva, A. Y. (2014). Comparative analysis of the complexity of chaotic and stochastic time series. *Радіоелектроніка, інформатика, управління*, (2 (31)).

27. Stavrinides, S. G., Papathanasiou, K., & Anagnostopoulos, A. N. (2015). Using modern RF tools to detect chaotic behaviour of electronic circuits and systems. *International Journal of Electronics*, *102*(2), 233-247.

28. Dabi-Prashad, O., & Kirichenko, L. (2009). Investigation of Time Series of Original Values of Currency Rates Measured on Small Time Frames on FOREX Using Methods of Chaos Theory. *Радиоэлектроника и информатика*, (4).

29. Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering–a decade review. *Information Systems*, *53*, 16-38.

30. Haykin, S. (1994). Neural networks: a comprehensive foundation. Prentice Hall PTR.

31. Abonyi, J., & Feil, B. (2007). Cluster analysis for data mining and system identification. Springer Science & Business Media.

32. Borgelt, C. (2006). Prototype-based classification and clustering.

33. Gustafson, D. E., & Kessel, W. C. (1979, January). Fuzzy clustering with a fuzzy covariance matrix. In 1978 IEEE conference on decision and control including the 17th symposium on adaptive processes (pp. 761-766). IEEE.

34. Miyamoto, S., Ichihashi, H., Honda, K., & Ichihashi, H. (2008). Algorithms for fuzzy clustering (pp. 1394-1399). Heidelberg: Springer.

35. Poljak, B. T., & Tsypkin, J. Z. (1980). Robust identification. Automatica, 16(1), 53-63.

36. Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., & Stahel, W. A. (2011). Robust statistics: the approach based on influence functions (Vol. 196). John Wiley & Sons.

37. Ljung, L. (2002). System Identification: Theory for the User Pers. Peking: Tsinghua University Press and Prentice.

38. Li, S. Z. (2009). Markov random field modeling in image analysis. Springer Science & Business Media.

39. Aggarwal, C. C., & Reddy, C. K. (2014). Data clustering. Algorithms and Application, Boca Raton: CRC Press.

40. Cruz, L. P., Vieira, S. M., & Vinga, S. (2015, September). Fuzzy clustering for incomplete short time series data. In Portuguese Conference on Artificial Intelligence (pp. 353-359). Springer, Cham.

41. Hotel Booking Dataset. URL: https://github.com/lauracarpaciu/Hotel-booking-demand/blob/master/hotel_bookings.csv (дата звернення: 05.11.2021).

42. Yahoo Finance API. URL: https://www.yahoofinanceapi.com/ (дата звернення: 10.11.2021).

43. Kirichenko, L., Pichugina, O., Zinchenko, H. (2021, December) Clustering time series of complex dynamics by features. In Kyiv Conference on Information Technology and Implementation.