МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ

(повне найменування інституту, факультету)

КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи) бакалавра

(ступінь вищої освіти)

на тему **Створення бібліотеки генерації даних для тестування алгоритмів кластеризації**

Виконав: студент(ка) 4 курсу, групи КНТ-819сп

Спеціальності 124 – Системний аналіз

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття рішень в складних системах»

Царенко Є.С. _____

(прізвище та ініціали)

Керівник _____ Рябенко А.Є. _____

(прізвище та ініціали)

Рецензент _____ Пирожок А.В. _____

(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування закладу вищої освіти)

факультет Комп'ютерних наук і технологій

кафедра Системного аналізу та обчислювальної математики

Ступінь вищої освіти бакалавр

Спеціальність 124 – Системний аналіз

Освітня програма

(спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»

ЗАТВЕРДЖУЮ

Завідувач кафедри проф. Г.В.Корніч

_____ «_____» _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Царенка Євгена Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Створення бібліотеки генерації даних для тестування алгоритмів кластеризації

   керівник проєкту (роботи) Рябенко Антон Євгенович, к.ф.-м.н., доц. затверджені наказом закладу вищої освіти від « 11 » травня 2022 року № 121

2. Строк подання студентом проєкту (роботи) « 15 » червня 2022 року

3. Вихідні дані до проєкту (роботи) Створення бібліотеки генерації даних для тестування алгоритмів кластеризації.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) У першому розділі наводиться огляд методів кластеризації та розглядаються їх сильні та слабкі сторони. У другому розділі обґрунтовується вибір для даної роботи мови програмування Python і розглядаються його інструментарій, який можна ефективно застосовувати для кластерного аналізу. У третій роботі описаний метод кластеризації під назвою "метод інкрементальних сфер" , і проводиться його порівняння з іншими відомими алгоритмами. У четвертій частині роботи описаний розроблений програмний продукт у вигляді бібліотеки Python, завдання якого - створювати набори 3-вимірних

даних як спіралеподібних стрічок тексту для тестування роботи алгоритмів кластерного аналізу. У висновку підбиваються підсумки і робляться висновки.

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

_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	Рябенко А.Є., к.ф.-м.н., доц.		
2	Рябенко А.Є., к.ф.-м.н., доц.		
3	Рябенко А.Є., к.ф.-м.н., доц.		
Нормаконтроль	Широкорад Д.В., к.ф.-м.н., ст. викл.		

7. Дата видачі завдання « 08 » жовтня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи	08.10.2021 – 30.10.2021	
2	Опрацювати літературу та існуючі дослідження за темою роботи	02.11.2021 – 20.11.2021	
3	Розробка програмної реалізації для вирішення задачі	20.11.2021 – 19.03.2022	
4	Розрахунки та аналіз даних	19.03.2022 – 30.04.2022	
5	Оформлення пояснювальної записки	03.05.2022 – 24.05.2022	
6	Попередній захист дипломної роботи та отримання рецензій.	25.05.2022 – 07.06.2022	
7	Захист дипломної роботи.	20.06.2022	

Студент _____ Царенко Є.С.

Керівник проєкту (роботи) _____ Рябенко А.Є.

РЕФЕРАТ

Дипломна робота: 43 с., 18 рис., 3 табл., 1 дод., 15 літературних джерел.

Об’єкт дослідження – моделі кластеризації.

Предмет дослідження – метод інкрементальних сфер.

Мета роботи – розробка програмного засобу для створення масивів даних для застосування на них методів кластеризації.

Методи дослідження – моделювання та аналіз.

В роботі розглянуті методи кластеризації та сучасні програмні інструменти для їх реалізації. Визначені проблемні моменти та засоби їх вирішення у запропонованому методі кластеризації (метод інкрементальних сфер). Описаний алгоритм методу та його реалізація мовою програмування Python. Наведені приклади його застосування.

Також розроблений програмний додаток у вигляді бібліотеки Python, який створює набори випадкових 2-вимірних та 3-вимірних даних у вигляді послідовності літер тексту. Ці дані можуть бути використані для тестування на них алгоритмів кластеризації та візуалізації їх результатів.

МАШИННЕ НАВЧАННЯ, АЛГОРИТМИ КЛАСТЕРИЗАЦІЇ, МЕТОД ІНКРЕМЕНТАЛЬНИХ СФЕР. ВІЗУАЛІЗАЦІЯ БАГАТОВИМІРНИХ ДАНИХ, ДАТАСЕТ, МОВА ПРОГРАМУВАННЯ PYTHON, БІБЛІОТЕКИ NUMPY, SCIKIT LEARN, MATPLOTLIB, SEABORN.

ЗМІСТ

ВСТУП

Методи кластеризації застосовують у різних галузях науки для визначення груп подібних за ознаками об’єктів. Для сучасних керівників може бути корисним виявлення проблемних кластерів для зосередження ресурсів підприємства на перспективніших напрямах. Для маркетингових досліджень кластерний аналіз дозволяє виявляти закономірності сезонної активності покупців. У електронній комерції застосовується таргетування аудиторії для підвищення ефективності просування товарів та послуг. З урахуванням наведених міркувань, напрям досліджень цієї роботи може вважатися актуальним.

Об’єктом дослідження є сучасні інформаційні технології, а предметом – спеціалізоване програмне забезпечення для кластерного аналізу та візуалізації багатовимірних даних.

Метою дипломної роботи є створення програмного продукту у вигляді бібліотеки Python, яка створюватиме набори даних складної форми для перевірки роботи різних алгоритмів кластеризації.

У перших двох частинах роботи аналізуються підходи кластерного аналізу та їх реалізація з використанням мови програмування Python.

У третій частині роботи описано метод інкрементальних сфер та показано його переваги порівняно з іншими методами кластерного аналізу.

У четвертій частині представлено опис бібліотеки генерації даних для тестування алгоритмів кластеризації, яка дозволяє створювати тривимірні набори даних у вигляді химерно вигнутих рядків тексту.

У висновку підбиваються підсумки дослідження, формулюються остаточні висновки по темі дослідження.

У додатках наведено програмний код.

1 ТЕОРЕТИЧНІ ЗАСАДИ КЛАСТЕРНОГО АНАЛІЗУ

Кластерний аналіз - багатовимірна статистична процедура, що виконує збір даних, що містять інформацію про вибірку об’єктів, а потім впорядковує об’єкти в порівняно однорідні групи [1-4]. Задачі кластеризації відносять до задач машинного навчання, як правило до задач без вчителя.

Кластеризацію з високою ефективністю застосовують у різноманітних сферах сучасної науки, таких як медицина, соціологія, психологія, економіка, біологія та інші [5].

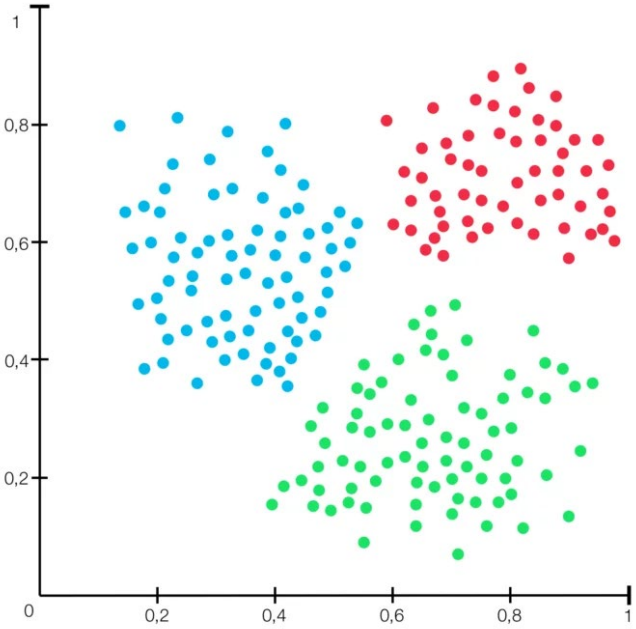

Рисунок 1.1 - Результат кластеризації у 2D-просторі

Під поняттям “кластер” розуміють об’єднання декількох елементів, яке може розглядатися як самостійний об’єкт із певними властивостями. Інше математичне визначення: кластер - це клас споріднених елементів статистичної сукупності. Іначе кажучи, коли ми визначаємо кластери, треба визначити, які

об’єкти можуть бути визнані подібними один до одного з урахуванням їх значень ознак.

На рис. 1.1 наведено приклад виконання кластерного аналізу над множиною точок двовимірного простору. Ті, що належать одному кластеру позначені одним кольором. Тобто, ми можемо бачити три кластери.

Кластерний аналіз виконує такі основні завдання:

- типологія або класифікація;
- концептуальні схеми систем об’єктів;
- перевірка обґрунтованості виділення груп подібності у наявних даних.
- при додаванні нових об'єктів - віднесення їх до вже наявних кластерів
- Застосування кластерного аналізу передбачає такі етапи:
- визначення даних для кластеризації. Можливо із застосуванням їх попередньої обробки;
- визначення простору ознак, тобто змінних які описуватимуть об'єкти;
- вибір міри подоби, за допомогою якої визначатиметься подоба (відмінність) об'єктів вибірки;
- вибір та застосування методу кластеризації, визначення груп об'єктів;
- перевірка достовірності результатів кластерного рішення;
- у разі потреби - передбачення для деяких об'єктів, до яких кластерів їх слід віднести.

Етапи застосування кластерного аналізу:

1. Визначається вибірка об'єктів для кластеризації.
2. Визначається множина змінних для оцінки об'єктів вибірці.
3. За потреби – значень змінних нормалізуються.
4. Обчислюються значення мір відстані між об'єктами.

5. Застосовується метод кластерного аналізу - створюються групи подібних об'єктів, тобто кластери).
6. Подаються результати аналізу.

Класифікація методів кластеризації:

- підхід, що базується на ймовірності (основні методи: k-середніх (k-means), k-медіан (k-medians), дискримінантний аналіз);
- підхід з урахуванням систем штучного інтелекту (до цієї групи можна віднести метод нечіткої кластеризації С-середніх (C-means), генетичний алгоритм);
- ієрархічний підхід (алгоритми: агломеративні (об'єднувальні) та дивізивні (розділяючі), графові алгоритми кластеризації);
- моделі основані на щільності (DBSCAN і в OPTICS);
- нейромережеві моделі (карти, що самоорганізуються (SOM), нейромережі Кохонена)

Формальна постановка задачі кластеризації згідно [6]:

“Розглядається X, що є множиною об’єктів та y - множина міток кластерів (імен або порядкових номерів). Також є заданою функція відстані між об’єктами множини $X\, p(x, x')$. Є кінцева навчальна вибірка об’єктів $X^m = (x_1, x_2, \ldots, x_m) \subset X$. Потрібно розбити вибірку підмножини, що не перетинаються, які називають кластерами. При цьому кожен кластер повинен складатися з об’єктів, близьких за значенням функції відстані, а об’єкти різних кластерів істотно відрізнялися. При цьому кожному об’єкту x_i приписується номер кластера y_i.

Алгоритм кластеризації - це функція, яка будь-якому об’єкту x_i ставить у відповідність номер кластера y_i. Множину Y у деяких випадках відомо заздалегідь, проте найчастіше ставиться завдання визначити оптимальну кількість кластерів, з погляду того чи іншого критерію якості кластеризації.”[6]

Розв’язання задачі кластеризації не є однозначним з урахуванням декількох причин:

- немає однозначно кращого критерію для оцінки якісті результатів кластеризації; проте існує низка критеріїв, які не повністю чітко виражені, проте виконують достатньо розумну кластеризацію «по побудові». Проте, вони можуть давати результати, що відрізняються. Тому, іноді, може бути потрібною думка експерта предметної області для оцінки результатів кластеризації;
- кількість кластерів не завжди відомо заздалегідь, тоді її треба визначати відповідно до певного суб'єктивного критерію;
- від вибору метрики (який також може бути суб'єктивним, та таким що обирається експертом) істотно залежить результат кластеризації.

Розглянемо основні метрики, що використовуються для обчислення міри схожості об'єктів вибірки:

1. Евклідова відстань. Найпоширеніша функція відстані. Є геометричною відстаню у багатовимірному просторі:

$$\rho(x_1, x_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \quad (1.1)$$

2. Відстань міських кварталів (манхеттенська відстань)

$$\rho(x_1, x_2) = \sum_{i=1}^{n}|x_{1i} - x_{2i}| \quad (1.2)$$

3. Відстань Чебишева

$$\rho(x_1, x_2) = max(|x_{1i} - x_{2i}|) \quad (1.3)$$

Дослідник повинен обирати метрику самостійно, оскільки результати кластеризації можуть суттєво відрізнятися за умов використання різних метрик.

Також, треба зазначити, що методи дуже відрізняються один від одного за обчислювальною складністю, що може ускладнювати роботу на великих датасетах.

Обчислювальна складність алгоритмів наведена у таблиці 1.1.

Таблиця 1.1 - Обчислювальна складність деяких алгоритмів кластерного аналізу

Алгоритм	Складність	Пояснення
ієрархічний	$O(n^2)$	n - розмір масиву вхідних даних
k-середніх	O(n k I)	k – кількість кластерів I – число ітерацій
Мінімальне кістякове дерево	$O(n^2 \log n)$	

З урахуванням вищевикладеного, проведення кластерного аналізу потребує обліку безлічі чинників та багато в чому носить експертний характер.

2 ОГЛЯД БІБЛІОТЕК PYTHON ДЛЯ НАУКОВИХ ОБЧИСЛЕНЬ ТА МАШИННОГО НАВЧАННЯ

Мова програмування Python [7] за останні кілька років набула високої популярності. Станом на травень 2022 року гідно з рейтингом TIOBE [8] (tiobe.com) вона займає 1 місце у світі за цитованістю та пропонованим вакансіям, згідно з сайтом github.com [9] - друге місце.

На думку фахівців, це обумовлено двома основними причинами:

- лаконічність синтаксису та ефективність коду
- наявність великої кількості програмних інструментів для різних сфер діяльності.

Зокрема, існують бібліотеки Python для наукових досліджень та аналізу даних, машинного навчання та нейронних мереж, візуалізації даних, лінгвістичного аналізу, веб-розробки, розробки ігор та багато іншого.

Бібліотека Python NumPy [10] - це основний модуль ля наукових обчислень. Це бібліотека Python, яка реалізує багатовимірні масиви, а також функції для ефективних дій з ними, включаючи математичні, логічні, змінення форми, сортування, вибір, лінійну алгебру, статистичні операції, випадкове моделювання та ін.

Оскільки Python відносять до інтерпретованих мов програмування, то швидкість обчислень з використанням базових типів даних є повільнішою у порівнянні с мовами, що компілюються (наприклад C або Java). Тому такі бібліотеки, як NumPy вирішують цю проблему, оскільки їх типи даних та обчислення відбуваються у так званому компільованому шарі з використанням інших мов програмування, таких як C та Фортран. Це дозволяє поєднати швидкодію компіляторів та високорівневість коду Python.

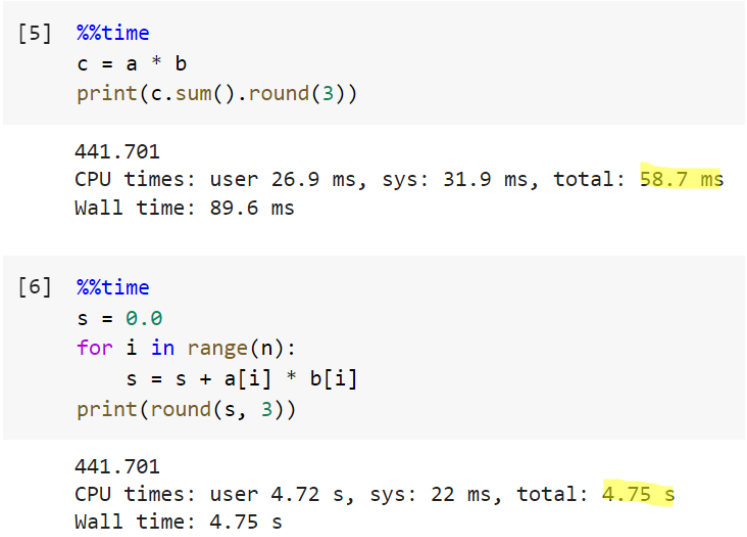

Рисунок 2.1 - Порівняння часу обчислень з використанням різних типів даних

Наприклад, якщо ми порівняємо час, що потрібний на попарне множення 10 мільйонів випадкових значень - у “стандартному” виконанні через циклічний оператор та з використанням спеціалізованої функції NumPy, то ми отримаємо прискорення у 80 раз (рис. 2.1).

Також дії над масивами NumPy реалізовані не тільки ефективно за часом виконання, а також за синтаксисом. Мінімальна роль тут відводиться рутинним діям - циклам та умовним операторам.

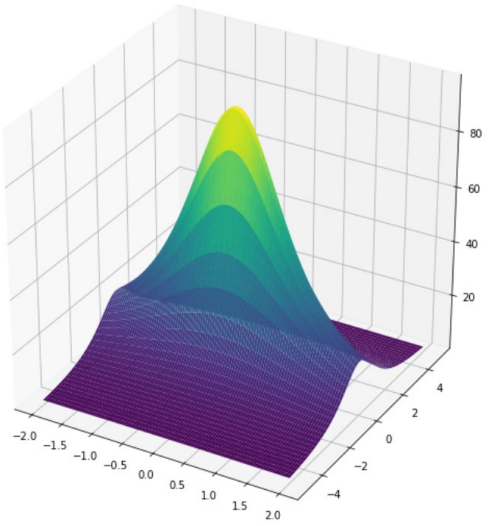

Рисунок 2.2 - Приклад візуалізації 3-вимірних даних за допомогою пакету Matplotlib

Для візуалізації даних у Python існують декілька пакетів, з яких найбільш популярними є Matplotlib та Seaborn [11]. В них реалізована можливість інтерактивної 2D та 3D візуалізації а також анімація. Серед доступних видів графічних функцій є такі, що реалізують: лінійні графіки, стовпчасті діаграми, діаграми розсіювання, контурні графіки, дендрограми та ін. У данії роботі використовуються графічні зображення, побудовані у цих пакетах.

Що стосується науки о даних та машинного навчання у Python, то найбільшу популярність у науковців має бібліотека Scikit Learn [12]. В бібліотеці представлений інструментарій для регресії, класифікації даних, зменшення розмірності даних, попередньої обробки даних та кластеризації. Scikit-learn – це пакет з відкритим кодом. Як і значна більшість матеріалів Python, він є безкоштовним для використання. Пакет ліцензовано під ліцензією BSD. Практично усі класи, що представлені в пакеті, мають багато атрибутів для експериментів з даними та вибору оптимальних параметрів моделей. Також, оцінка ефективності роботи моделей може оцінюватися за багатьма показниками (наприклад, на рис. 2.3 наведений фрагмент з офіційної документації пакету, що

показує, які показники можуть бути використані для оцінки якості моделей кластеризації).

Clustering	
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score
'adjusted_rand_score'	metrics.adjusted_rand_score
'completeness_score'	metrics.completeness_score
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score
'homogeneity_score'	metrics.homogeneity_score
'mutual_info_score'	metrics.mutual_info_score
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score
'rand_score'	metrics.rand_score
'v_measure_score'	metrics.v_measure_score

Рисунок 2.3 - Показники оцінки роботи моделей кластеризації Scikit-learn

В пакеті представлені різні методи кластеризації. Вони реалізовані як класи. Наведемо їх перелік: K-Means, Affinity propagation, Mean-shift, Spectral clustering, Ward hierarchical clustering, Agglomerative clustering, DBSCAN, OPTICS, Gaussian mixtures, BIRCH, Bisecting K-Means.

На рис. 2.4 схематично показано особливості застосування різних методів на низці так званих “іграшкових” наборах даних. До речі, їх генерація також реалізована в бібліотеці.

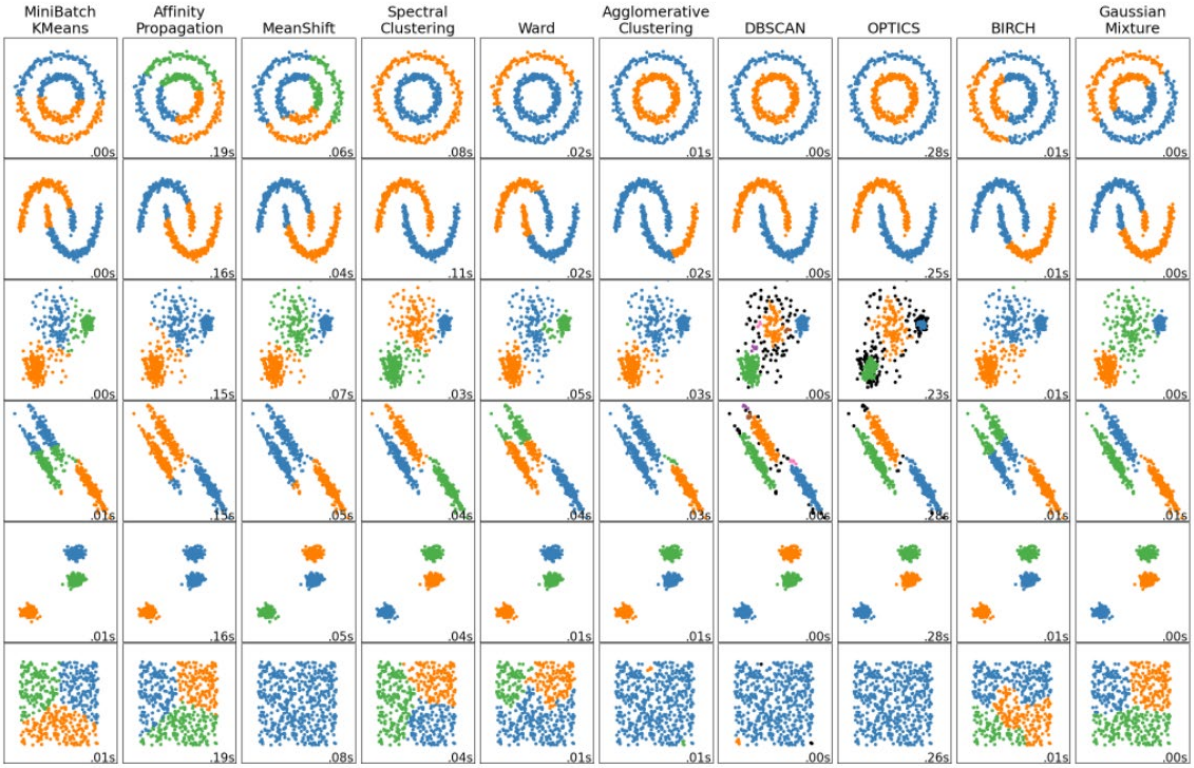

Рисунок 2.4 - Графічне представлення застосування методів кластеризації на “іграшкових” наборах даних

Як можна побачити, наприклад, на наборі даних у вигляді концентричних кіл, або півкіл (два верхні рядки на рис. 2.4), не усі методи адекватно визначають структуру кластерів. Тому, у наступному розділі розглядається реалізація методу інкрементальних сфер [13], який добре працює на таких складних випадках.

Практичне застосування методів на багатовимірних даних полягає в створенні моделі у вигляді об’єкту одного з вказаних класів, налаштування параметрів (наприклад, встановлення кількості кластерів), навчання моделі та оцінка результатів (візуалізація та/або застосування метрик якості).

Якщо X – це багатовимірний масив NumPy Array, то фрагмент кода Python, що виконує кластеризацію за методом k-середніх може виглядати так:

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> y_predicted = kmeans.predict(X)
```

Тобто, спочатку імпортуються певні модулі та класи, потім створюється об’єкт класу KMeans (змінна kmeans) та навчається на датасеті *X* (метод fit). Після завершення навчання за допомогою методу predict ми отримуємо результат у вигляді масиву міток *y_predicted*. Потім, застосовуючи масиви *X* та *y_predicted* можемо побудувати точкову діаграму, на якій різними кольорами або різною формою маркерів позначимо окремі кластери (рис 2.5). На цьому рисунку випадкові дані були сгенеровані за допомогою функції make_blobs з субмодулю sklearn.datasets.

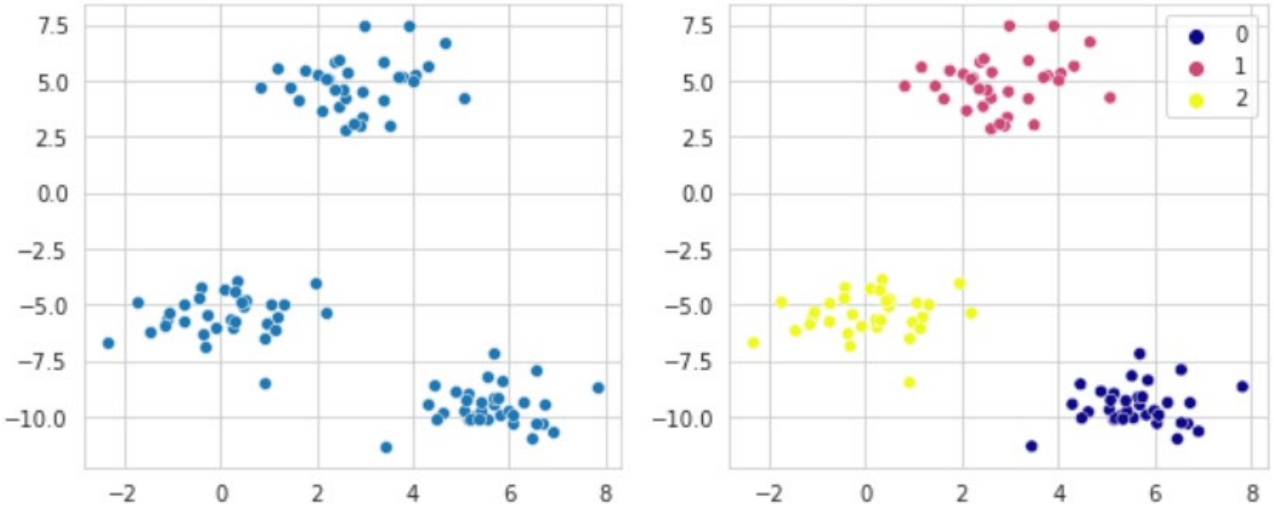

Рисунок 2.5 - Візуалізація даних до та після застосування методу k-середніх (функція make_blobs, кількість точок - 100)

Однак, якщо сформувати набір даних за допомогою функції make_moons у вигляді двох “серпів”, то застосування методу k-середніх не дасть ефективного результату (рис 2.6). Проте, застосування методу інкрементальних сфер приводить до якісного визначення структур даних (рис. 2.7).

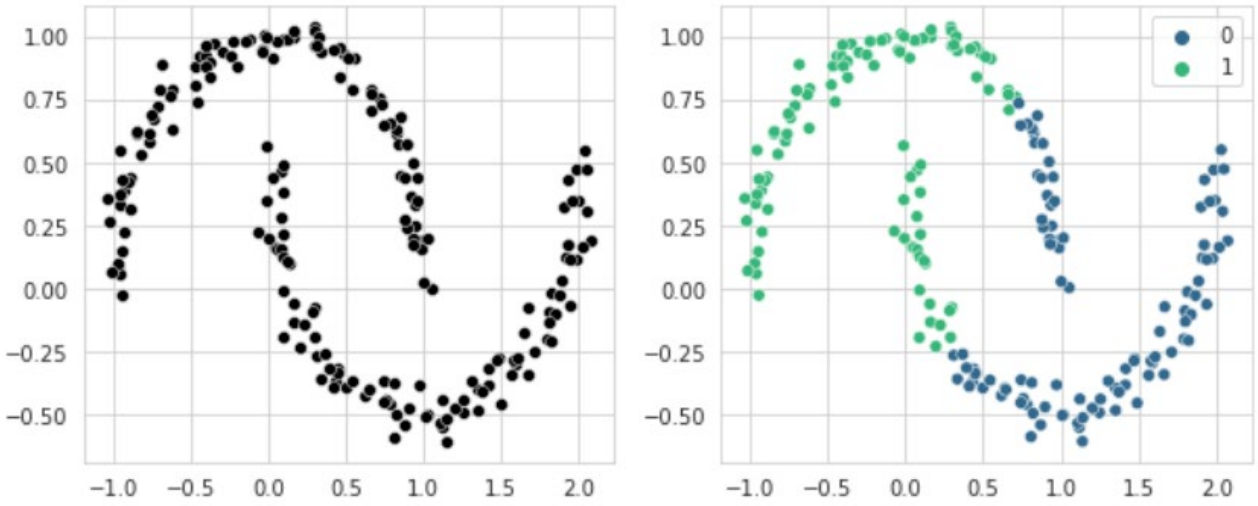

Рисунок 2.6 - Візуалізація даних до та після застосування методу k-середніх (функція make_moons, кількість точок - 200)

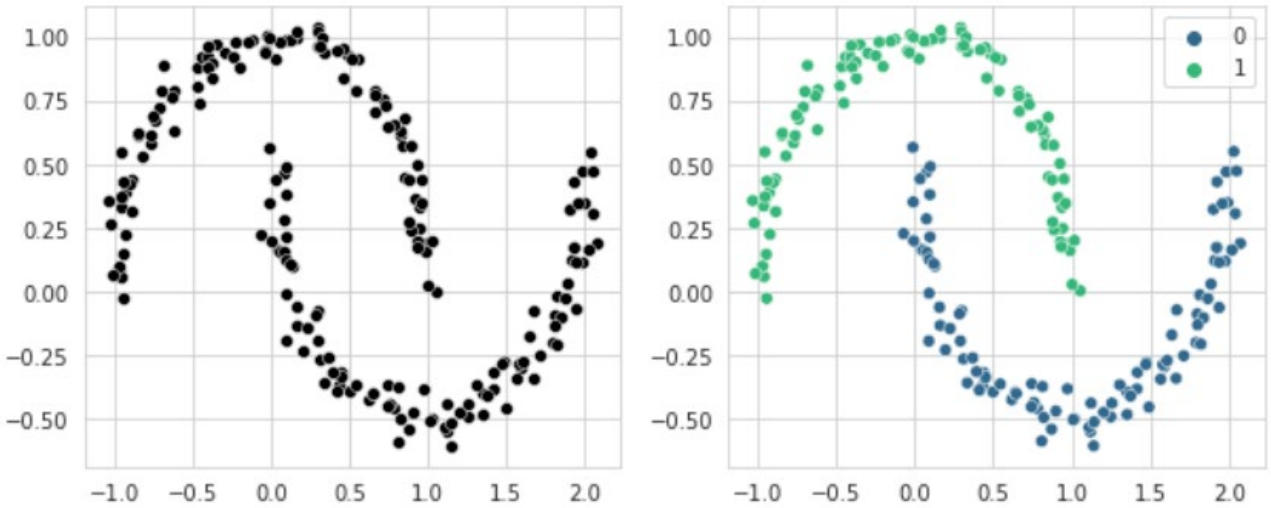

Рисунок 2.7 - Візуалізація даних до та після застосування методу інкрементальних сфер (функція make_moons, кількість точок - 200)

Докладніше про цей метод та про розробку на його основі програмного забезпечення - у наступному розділі.

3 МЕТОД ІНКРЕМЕНТАЛЬНИХ СФЕР ТА ЙОГО ПРОГРАМНА РЕАЛІЗАЦІЯ

Метод кластеризації даних під назвою “метод інкрементальних сфер” був запропонований Пінчуком В.П. у роботі [13].

Слід зазначити, що проблемою багатьох відомих методів кластеризації є відсутність чіткого критерію для визначення кількості кластерів, що є скритними у вихідних даних. Метод інкрементальних сфер пропонує засіб вирішення цієї проблеми.

Сутність метода полягає в наступному. Нехай в нас є набір даних, що представляє характеристики групи об'єктів *H*, що підлягає кластеризації. Характеристики кожного з об’єктів представлені вектором, що визначає точку в *m*-вимірному просторі, в якому визначена метрика ρ. Відстань між елементами з індексами і та j позначимо як $\rho(x_i, x_j)$, де $\rho(x_i, x_j)$- параметричні вектори об’єктів.

Нехай маємо деяке розбиття $P(H) = H_1 \cup H_2 \cup \ldots \cup H_n$ множини вихідних об'єктів *H* на підмножини. Розбиття *P(H)* будемо вважати результатом часткової або повної кластеризації, а підмножини - проміжними або кінцевими кластерами. Визначимо величину відстані між кластерами $\rho(H_i, H_j)$ як відстань між найближчими об'єктами, один із яких належить кластеру H_i , а другий - H_j:

$$\rho(H_i, H_j) = \min_{u \in H_i v \in H_j} (\rho(u, v)) \tag{3.1}$$

Інкрементальною сферою кластеризації називатимемо сферу діаметру *D*, яка асоціюється з кожним об’єктом кластеризації і центр якої знаходиться в точці r_k. Будемо використовувати таке правило кластеризації: якщо інкрементальні сфери об'єктів перетинаються, то вони належать одному

кластеру. Розглянемо залежність кількості кластерів $N_{cl}(D)$, вважаючи що D змінюється від 0 до ρ_{max}.

Відносно залежності $N_{cl}(D)$ можна зазначити:

1. Вона має ступінчастий характер (значення функції є цілим числом).
2. Зі збільшенням D число кластерів зменшується, змінюючись від n до 1.

Типовий вид залежності $N_{cl}(D)$ наведено на рис. 3.2.

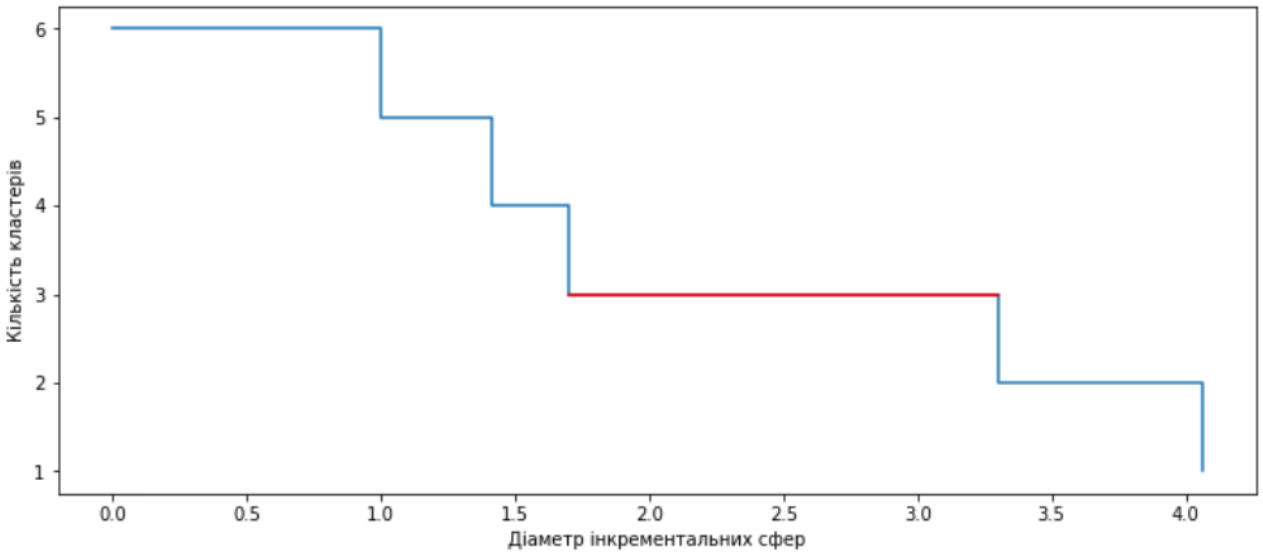

Рисунок 3.1 - Залежність кількості кластерів від діаметра інкрементальних сфер

У запропонованому методі число кластерів визначається характером залежності $N_{cl}(D)$ відповідно до наступного правила: число кластерів відповідає найдовшій сходинці на графіку цієї залежності (виділено червоним кольором).

Для наочності можна уявити об'єкти в 2-мірному просторі як 2-мірних сфер, тобто. кругів радіусу, що збільшується, і вказати кількість кластерів, що відповідають значенням радіусу сфер. (рис. 3.2).

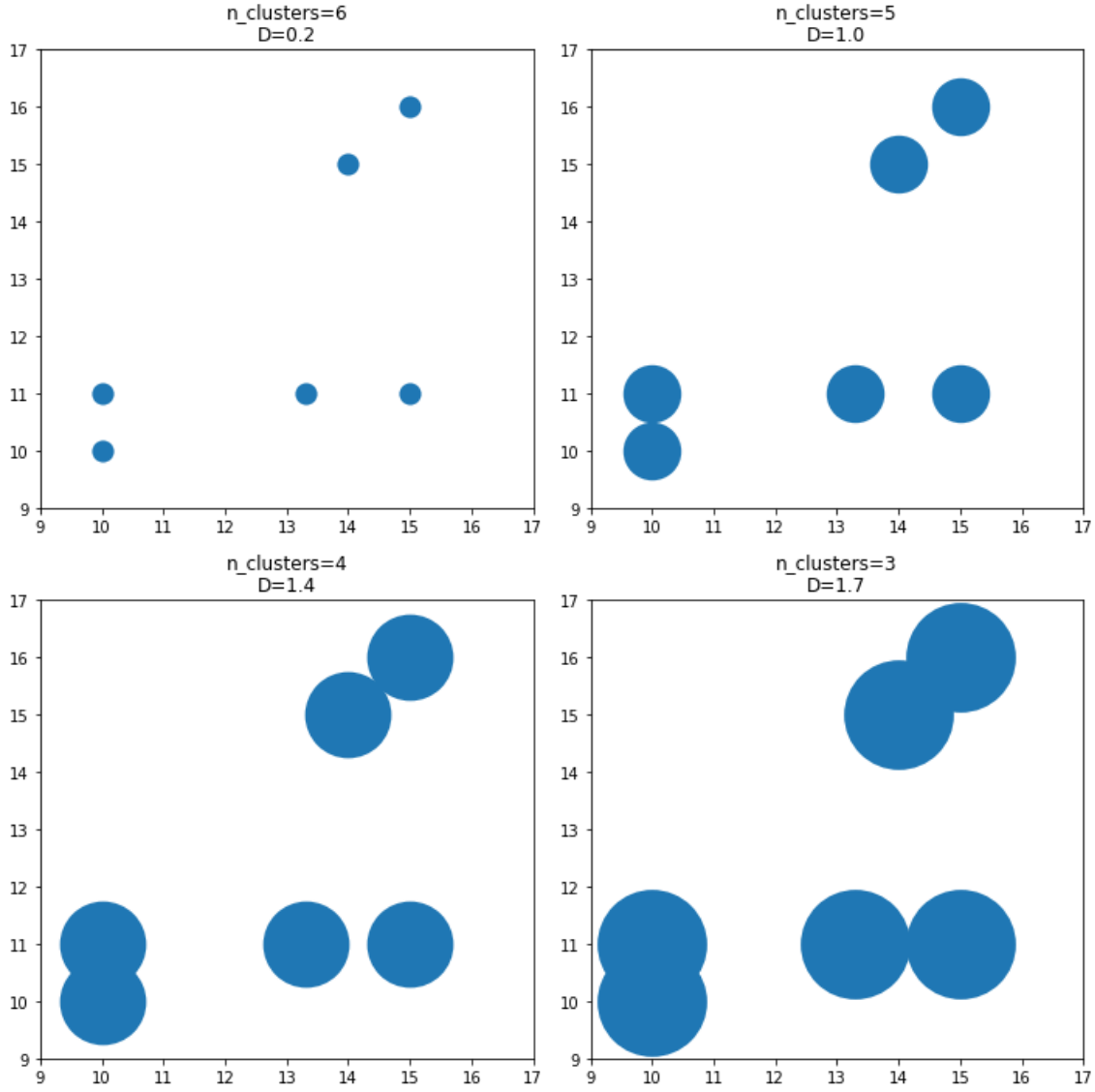

Рисунок 3.2 - Інкрементальні сфери у 2-вимірному просторі

Нехай D_k - найменша величина діаметра інкрементальної сфери, яка призводить до чергового (k-тому за рахунком) зменшення числа кластерів. Позначимо через Δ_k величину k-ої сходинки залежності $N_{cl}(D)$: $\Delta_k = D_{k+1} - D_k$. Тоді кількість кластерів знаходимо за таким правилом:

1) визначаємо величину k, що відповідає максимальному значенню Δ_k;

2) значення функції $N_{cl}(D)$ дає шукане.

Як показник успішності кластеризації заданої множини об'єктів можна використовувати відношення:

$$K = \frac{\Delta_{max}^{(1)}}{\Delta_{max}^{(2)}} \tag{3.2}$$

де $\Delta_{max}^{(1)}$ - найбільше значення Δ_k;

$\Delta_{max}^{(2)}$ - друге за величиною значення Δ_k.

Якщо значення *K* досить велике, кластеризацію можна вважати успішною. Порогове значення величини *K* визначається характером оброблюваних даних (чи є інформація, що міститься у вихідних даних, достатньої для правильної кластеризації).

З метою програмної реалізації методу був розроблений модуль IncSpheres з використанням мови програмування Python, та її бібліотек для обчислення та візуалізаціїї даних NumPy, Matplotlib та Seaborn. Основні результати було викладено в роботі [14].

Програмний компонент модулю створений за методологією об’єктно-орієнтованого програмування, тобто у вигляді класу. Основні методи: fit (обчислює кластеризацию наданих даних у вигляді багатовимірного масиву за методом інкрементальних сфер), predict (повертає масив позначок, що відносять об’єкти до різних кластерів, що потім використовується для графічного представлення даних, див. рис. 1), draw_diagram (будує діаграму залежності кількості кластерів від радіусу сфер).

При автоматичному створенні діаграми кількості кластерів акцент візулізації робиться на представленні оптимальної кількості кластерів (ця “сходинка” акцентується контрастним кольором), що можна побачити на рис. 3.1.

Модуль стандартизовано документований та готується до розміщення у глобальному репозиторії пакетів PyPI, після чого кожний користувач Python буде мати можливість його завантаження та застосування.

Цікавим є також швидкість роботи досліджуваного методу порівняно з іншими методами кластеризації. Було проведено експеримент, у межах якого створювалися набори випадкових даних у кількості від 100 до 10 000 000 об'єктів, до них застосовувалися різні алгоритми кластерного аналізу та засікався час їхньої роботи. Результати представлені у таблиці 3.1.

Ми порівнювали алгоритми k-середніх, DBSCAN (density-based spatial clustering of applications with noise) [15] та метод інкрементальних сфер. Як можна побачити, найбільш швидким є алгоритм k-середніх (це пояснюється його простотою). Метод інкрементальних сфер є повільнішим, але не занадто. Алгоритм DBSCAN вже на вибірці розміром 1 мільйон значно уповільнюється (приблизно 1 година), тому для 10 мільйонів об'єктів для цього метода експеримент не проводився.

Таблиця 3.1 - Час роботи алгоритмів кластерізації

Розмір масиву випадкових даних	Час роботи алгоритмів, мс		
	k-середніх	DBSCAN	Метод інкрементальних сфер
100	67	5	79
1 000	238	13	358
10 000	330	400	587
100 000	1050	14900	2210
1 000 000	3990	2740000	5800
10 000 000	40800	-	72900

4 РОЗРОБКА БІБЛІОТЕКИ ГЕНЕРАЦІЇ ДАНИХ

Оскільки у бібліотеці Scikit-learn є низка методів для створення так званих “іграшкових” датасетів - коли випадкові точки створюють певні структури, на яких наочно можна оцінити, як той чи інший метод кластеризації може передбачити ці структурні групи. Майже усі типи таких датасетів можна побачити на рис. 2.4.

У рамках цієї роботи розроблено програмний продукт подібного призначення. Він може створювати набори 2-мірних та 3-мірних даних, які за структурою будуть схожі на фрагменти тексту. Якщо, наприклад, у 3-мірному просторі, цей "текст" буде закручений у спіраль, то такі дані будуть цікаві для оцінювання результатів роботи методів кластеризації - чи зможуть вони виділити окремі "літери" в кластери.

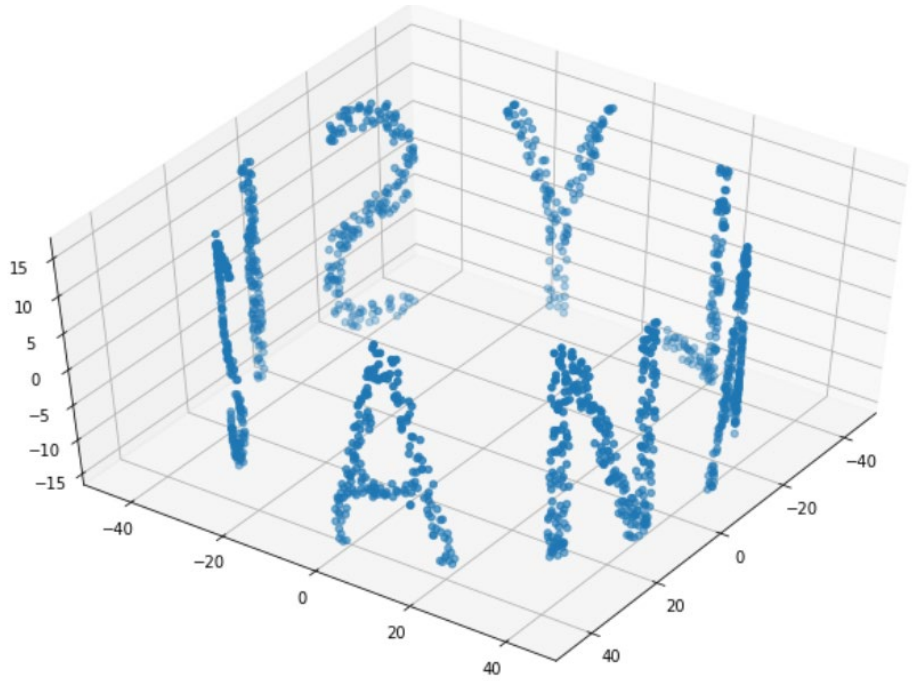

Рисунок 4.1 - Приклад створення 3-вимірного датасету на основі слова “analysis” у вигляді кола

На рис 4.1 наведено приклад, як можна створити 3-вимірний набір даних, що нагадує слово “analisys”, згорнуте у коло.

Можна також "закрутити в спіраль" це слово. Результат можна побачити на рис. 4.2 під двома різними кутами огляду для наочності.

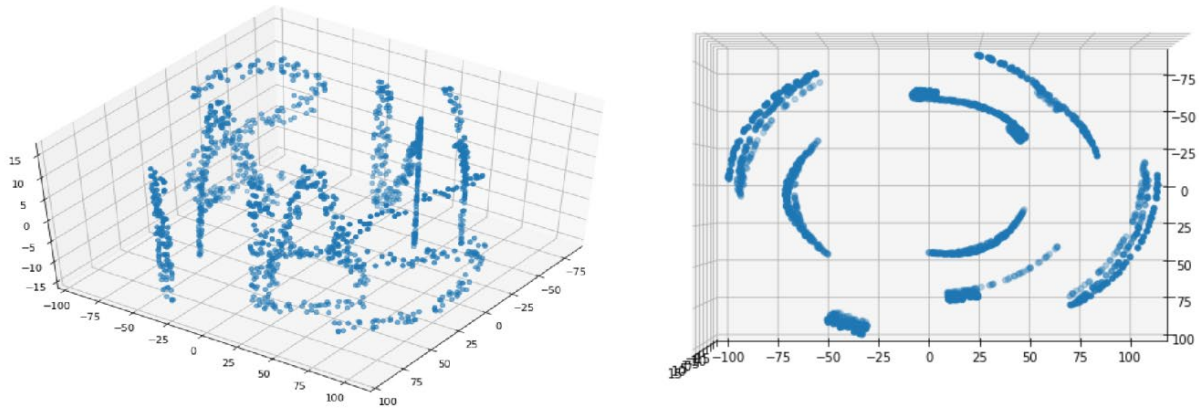

Рисунок 4.2 - Приклад створення 3-вимірного датасету на основі слова “analysis” у вигляді спіралі

Тепер на цих спіралеподібних даних ми можемо порівняти, які результати нам дадуть, наприклад, метод k-середніх і метод інкрементальних сфер. Кількість кластерів - 8 (за кількістю літер у слові). Для наочності, на рис. 4.3 покажемо “вид зверху” на “спіраль”.

Як можна побачити, метод k-середніх не впорався з відокремленням “літер” у кластери (рис. 4.3а). Це пов'язано з тим, що деякі точки з різних “літер” знаходяться ближче одна до одної, ніж точки з однієї “літери”. Проте, метод інкрементальних сфер ефективно визначає “літери” як окремі кластери (рис. 4.3б).

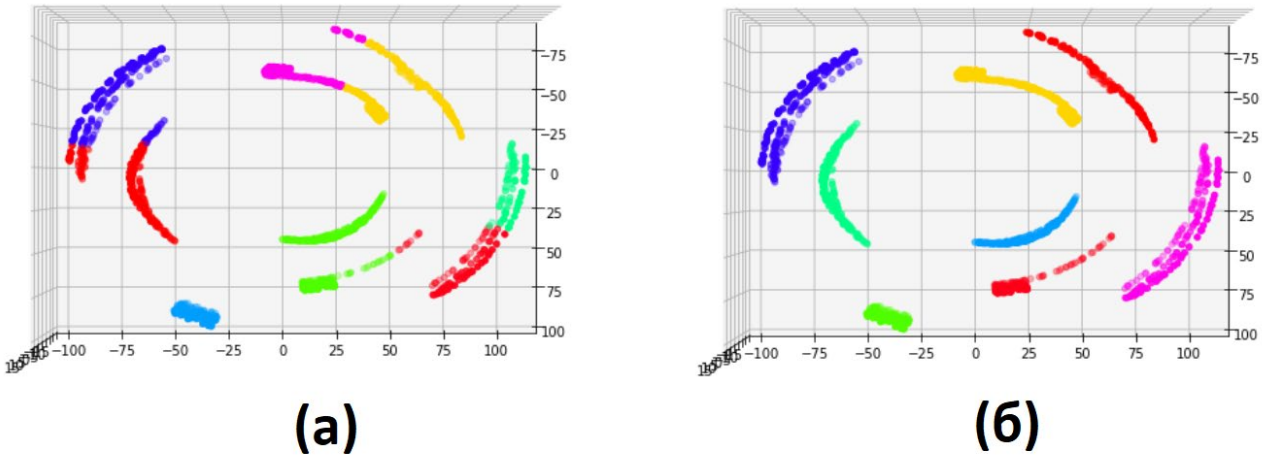

Рисунок 4.3 - Результати кластерного аналізу за методами k-середніх (а) та інкрементальних сфер (б)

Розглянемо основні етапи програми.

Перший етап - створення та обробка зображення. На цьому етапі за допомогою пакету Matplotlib візуалізується текст у вигляді чорно-білого зображення , яке потім трансформується у двовимірний масив чисельних значень пікселів (використовується бібліотека обробки зображень PIL). Чисельні значення залежать від яскравості пікселів зображення. Наведемо приклад зображення 8 на 8 пікселів та масив даних, що йому відповідає (рис. 4.4).

У цьому випадку, що більш темним є піксель, то більшим є числове значення, яке йому відповідає. І навпаки – значення близькі до нуля відповідають світлим областям зображення.

Після цього проводимо перетворення масиву даних: тим значення, які більші за середній по всьому масиву, привласнюємо 1, решті - 0. Таким чином позбавляємося півтонів у зображенні (рис. 4.5).

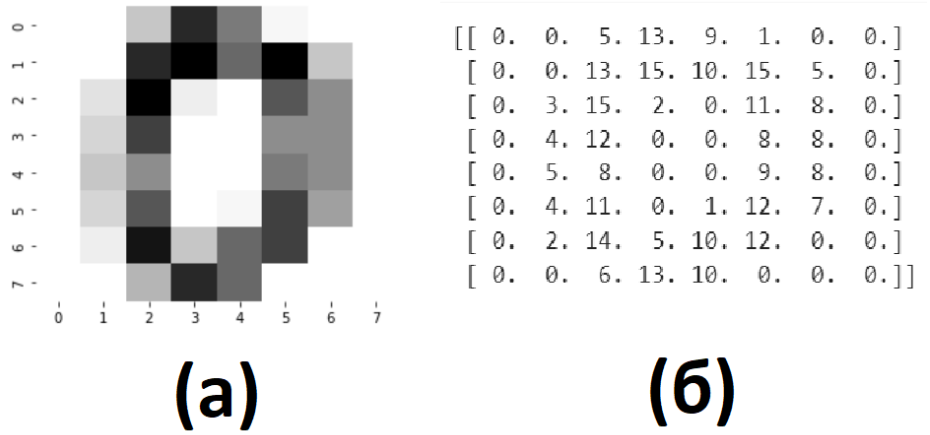

Рисунок 4.4 - Цифрове зображення літери “О” (а) та масив, що йому відповідає (б)

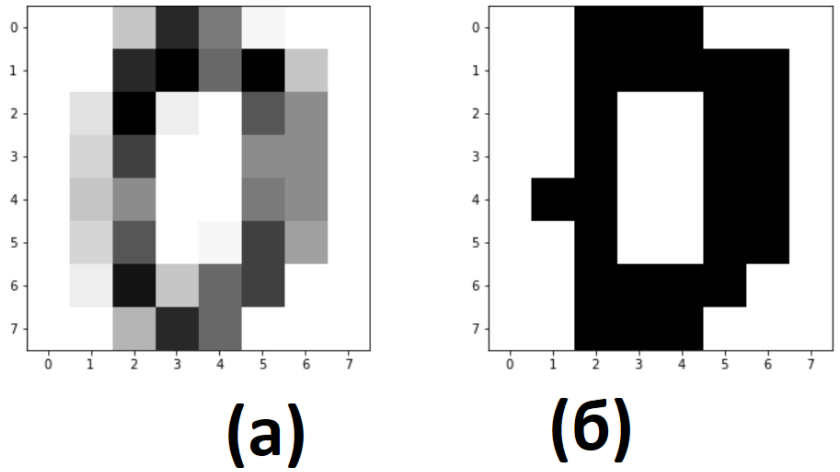

Рисунок 4.5 - Цифрове зображення літери “О” до (а) та після (б) перетворення

Другий етап полягає в "засіюванні" двовимірної області випадковими точками по трафарету редукованого зображення. Якщо взяти за основу структуру зображення з рис. 4.5 та заповнити її 500 випадковими точками, то вийде набір випадкових даних, представлений на рис. 4.6. Позначимо його через *X*.

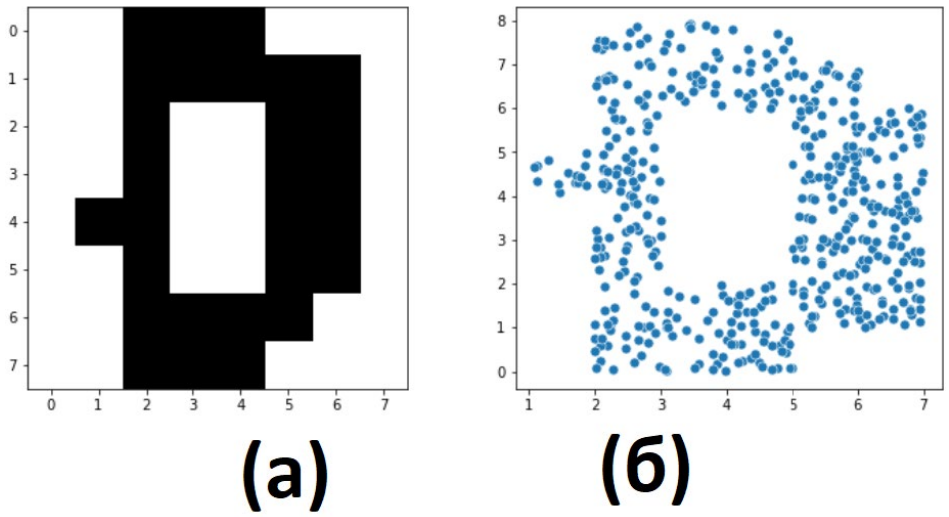

Рисунок 4.6 - “Трафарет” зображення (а) та масив з 500 випадкових точок за його структурою (б)

Третій етап полягає у перетворенні двовимірних даних на тривимірні. Якщо потребується форма у вигляді кола, то задля цього виконуються наступні дії. Нехай x_2 – множина значень 1-ї координати об'єктів X, y_2 – множина значень 2-ї координати об'єктів X. Тоді можна розрахувати радіус кола (4.1), значення фази для кожної точки (4.2) . Використовуючи ці значення можна розрахувати 3-вимірні координати x_3, y_3, z_3 для кожної точки (4.3).

$$R = \frac{maxx_2 - minx_2}{2\pi} \tag{4.1}$$

$$\phi_i = 2\pi \frac{x_2(i) - minx_2}{maxx_2 - minx_2} \tag{4.2}$$

$$\begin{aligned} x_3(i) &= R cos\big(\phi(i)\big) \\ y_3(i) &= R sin\big(\phi(i)\big) \\ z_3(i) &= y_2(i) \end{aligned} \tag{4.3}$$

В результаті цих дій отримаємо набір даних, подібний тому, що зображений на рис. 4.1. Тобто, перша координата 2D набору даних перетворена на першу та другу координати 3D набору відповідно, а друга координата 2D – на третю координату 3D. Схематично це представлено на рис. 4.7.

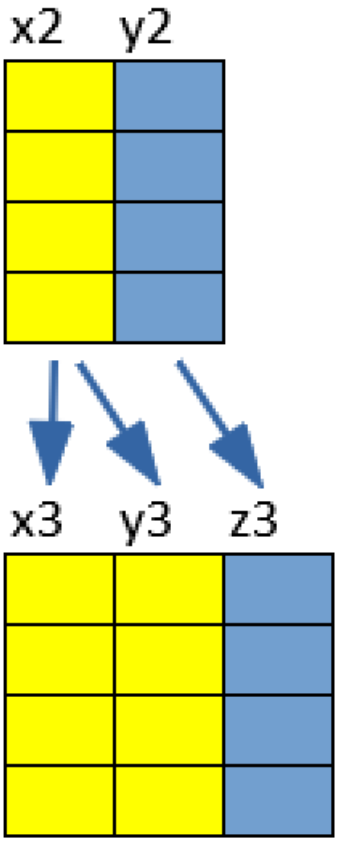

Рисунок 4.7 - Схема перетворення значень 2D координат в 3D координати

Якщо потребується 3D набір у вигляді спіралі (рис. 4.2), то формулу 4.3 треба модифікувати з урахуванням нового параметру k – кількості обертів спіралі:

$$\begin{aligned} x_3(i) &= R\cos\left(k\phi(i)\right)\left(1+\frac{k\phi(i)}{2\pi}\right) \\ y_3(i) &= R\sin\left(k\phi(i)\right)\left(1+\frac{k\phi(i)}{2\pi}\right) \\ z_3(i) &= y_2(i) \end{aligned} \tag{4.4}$$

Отримані дані можна експортувати у бінарний файл з розширенням “.npy” або у текстовий файл для збереження даних або для обміну даними з іншими додатками.

Створено засіб інтерактивної візуалізації отриманих 3-вимірних даних. За допомогою модулю Python ipywidgets, який дозволяє змінювати параметри

діаграм, такі як кути огляду, у спеціальних віджетах, можемо побачити дані з різних сторін (рис. 4.8).

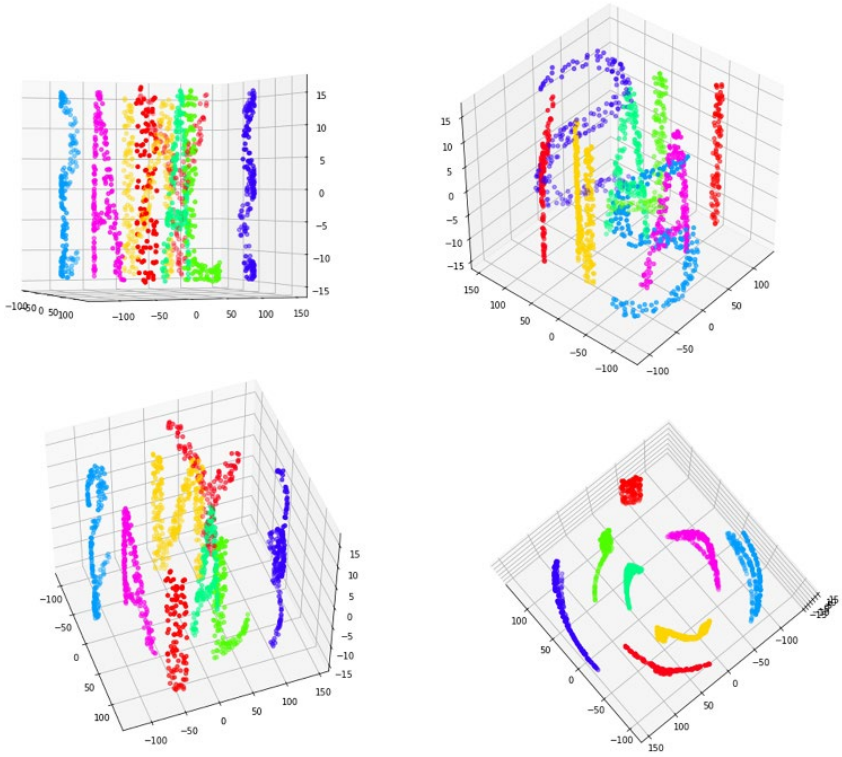

Рисунок 4.8 - Візуалізація 3-вимірних даних під різними кутами огляду

При розробці програмного додатку використовувався об'єктно-орієнтований підхід. Назви класів та методів створювались відповідно до стандартів бібліотеки Scikit-Learn. Програмний код мовою Python наведений у додатку А.

ВИСНОВКИ

В ході виконання роботі були проаналізовані методи кластерного аналізу та розглянуті можливі ускладнення при кластеризації певних структур даних.

Для виконання поставлених цілей роботи було обрано мову програмування Python, цей вибір був обґрунтований наявністю високорівневих інструментів для обчислень та візуалізації даних. Розглянуті представлені в бібліотеці Scikit-learn алгоритми кластеризації даних.

Описані переваги кластеризаційного “методу інкрементальних сфер” та його програмна реалізація. Наведені приклади успішного застосовування методу у порівнянні з методом k-середніх.

В основній частині роботи представлені результати розробки програмного додатку для створення багатовимірних даних у вигляді спіралеподібних стрічок тексту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мандель, И. Д. Кластерный анализ [Текст] / И. Д. Мандель – М. : Финансы и статистика, 1988. – 176 с.

2. Жамбю, М. Иерархический кластер-анализ и соответствия [Текст] / М. Жамбю. – М. : Финансы и статистика, 1988. – 345 с.

3. Odell, P. L. Cluster Analysis [Текст] / P. L. Odell, B. S. Duran. – Berlin, Heidelberg: Springer-Verlag, 1974. – VI. – 140 p.

4. Гитис, Л. Х. Статистическая классификация и кластерный анализ [Текст] / Л. Х. Гитис. – М. : Издательство Московского государственного горного университета, 2003. – 157 с.

5. Everitt, B. S. Cluster Analysis [Текст] / B. S. Everitt, S. Landau, M. Leese, D. Stahl. – 5th Edition. – Chichester, West Sussex: U.K: Wiley, 2011. – 352 p.

6. Ким, Дж.-О. Факторный, дискриминантный и кластерный анализ [Текст] / Дж.-О. Ким, Ч. У. Мьюллер, У. Р. Клекка, И. С. Енюков. – М.: Финансы и статистика, 1989. – 215 с.

7. Python.org – Python is a programming language that lets you work quickly and integrate systems more effectively [Електронний ресурс]. – Режим доступа: https://python.org/ (дата звернення 20.05.2022) – Назва з екрана.

8. TIOBE Index [Електронний ресурс]. – Режим доступа: https://www.tiobe.com/tiobe-index/ (дата звернення 20.05.2022) – Назва з екрана.

9. Github. Learn Git and GitHub without any code! [Електронний ресурс] – Режим доступа: https://github.com/ (дата звернення 20.05.2022) – Назва з екрана.

10. NymPy - The fundamental package for scientific computing with Python. [Електронний ресурс]. – Режим доступа: https://numpy.org/ (дата звернення 20.05.2022) – Назва з екрана.

11. Matplotlib: Visualization with Python [Електронний ресурс]. – Режим доступа: https://matplotlib.org/ (дата звернення 20.05.2022) – Назва з екрана.

12. Scikit-learn. Machine Learning in Python. [Електронний ресурс]. – Режим доступа: https://scikit-learn.org/stable/ (дата звернення 20.05.2022) – Назва з екрана.

13. Пинчук, В.П. Кластеризация данных методом инкрементальных сфер [Текст] / В.П. Пинчук, А.Е. Рябенко // Системний аналіз. Інформатика. Управління (САІУ-2013) : матеріали IV Міжнародної науково-практичної конференції (м. Запоріжжя, 13–16 березня 2013 року) / Міністерство освіти і науки, молоді та спорту України, Академія наук вищої школи України, Запорізька обласна державна адміністрація, Класичний приватний університет. – Запоріжжя : КПУ, 2013. – С. 206-207.

14. Терещенко, Е.В. Створення програмного модуля для кластеризації даних методом інкрементальних сфер [Електронний ресурс] / Е.В. Терещенко, Д.В. Широкорад, А.Є. Рябенко, Є.С. Царенко // Тиждень науки-2022 : тези доповідей науково-практичної конференції, Запоріжжя, 18–22 квітня 2022 р. / НУ «Запорізька політехніка». – Запоріжжя, 2022. – С.886-888.

15. Ester, M. A density-based algorithm for discovering clusters in large spatial databases with noise [Текст] / M. Ester, H.-P. Kriegel, J. Sander, X. Xu // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). – AAAI Press, 1996. – c. 226–231.

ДОДАТОК А

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn
import seaborn as sns

np.set_printoptions(suppress=True)

from copy import deepcopy

    class IncSpheres:
        def __init__(self):
            pass

        def fit(self, X):
            self.X = X
            self.calc_distances()
            self.calc_clusters()
            self.calc_best_params()

        def calc_distances(self):
            jj, ii = np.meshgrid(
    np.arange(self.X.shape[0]),
    np.arange(self.X.shape[0]))
            ii = ii.reshape(-1, 1)
            jj = jj.reshape(-1, 1)

            @np.vectorize
```

```
            def get_dist(i, j):
                x1 = self.X[i, :]
                x2 = self.X[j, :]
                return np.power(np.sum(
                          (x1 - x2) ** 2),
                          0.5)

            dist = get_dist(ii, jj)
            self.D = np.concatenate(
                [ii, jj, dist],
                axis=1
            )
            mask = self.D[:, 2] > 0
            self.D = self.D[mask]
            self.D = self.D[self.D[:, 2].argsort()]
            self.D = self.D[::2, :]

        def calc_clusters(self):
            self.clusters = [[[i]
    for i in range(self.X.shape[0])]]
            self.R = [0.0]
            idx = 0
            # print(self.D)
            while len(self.clusters[-1]) > 1:
                i, j, r = self.D[idx]
                i, j = int(i), int(j)
                # print(i, j, r)
                current_situation = deepcopy(
                    self.clusters[-1])
```

```
                # print(current_situation)
                in_same_cluster = False
                for cluster in current_situation:
                    if i in cluster:
                        idx_cluster_i =
    current_situation.index(cluster)
                    if j in cluster:
                        idx_cluster_j =
    current_situation.index(cluster)
                    if i in cluster and j in cluster:
                        in_same_cluster = True
                # print(in_same_cluster, idx_cluster_i,
idx_cluster_j)
                if not in_same_cluster:
                    self.R.append(r)
                    current_situation[idx_cluster_i].ext
end(current_situation.pop(idx_cluster_j))
                    self.clusters.append(current_situati
on)
                idx = idx + 1

        def calc_best_params(self):
            dR = np.array(self.R[1:]) - np.array(
    self.R[:-1])
            am = np.argmax(dR)
            # print(am)
            self.best_n_clusters = self.X.shape[0] - am

        def predict(self, n_clusters='best'):
```

```
            if n_clusters == 'best':
                n_clusters = self.best_n_clusters

            situation = [s for s in self.clusters if len
(s) == n_clusters][0]
            # print(situation)
            labels = np.zeros(shape=(self.X.shape[0],),
dtype=int)
            for i, c in enumerate(situation):
                for el in c:
                    labels[el] = i
            # print(labels)
            return labels

        def draw_diagram(self):
            x_plot, y_plot = [], []
            dR = np.array(self.R[1:]) - np.array(self.R[
                :-1])
            am = np.argmax(dR)
            for counter, r in enumerate(self.R):
                if 0 < counter:
                    x_plot.append(r)
                    y_plot.append(
                self.X.shape[0] - counter + 1)
                x_plot.append(r)
                y_plot.append(
                self.X.shape[0] - counter)
            plt.figure(figsize=(12, 5))
            plt.plot(x_plot, y_plot)
```

```
                plt.plot([self.R[am], self.R[am + 1]],
                [self.best_n_clusters,
                self.best_n_clusters], color='red')
                plt.xlabel('Radius of Incremental Spheres')
                plt.ylabel('Number of Clusters')
                plt.show();

import PIL

text = 'analysis'
text = text.upper()
text = ' '.join(list(text))
FILENAME = 'image.png'

plt.text(x=0, y=0, s=text, size=40)
plt.axis('off');

plt.savefig(FILENAME)
image = PIL.Image.open(FILENAME).convert('L')
image_array = np.array(image)

image_array = np.where(image_array < 100, 1, 0)

xx, yy = np.meshgrid(np.arange(image_array.shape[1]),
np.arange(image_array.shape[0]))
X = np.concatenate([xx.reshape(-1, 1), yy.reshape(-1,
1)], axis=1)

y = image_array.flatten()
```

```
X.shape, y.shape

%%time

from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n_neighbors=5)
neigh.fit(X, y);

X_random = np.random.rand(100000, 2) *
image_array.shape[::-1]
y_pred = neigh.predict(X_random)

plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_random[:, 0], y=X_random[:, 1],
hue=y_pred);

X_extracted = X_random[y_pred > 0, :]
X_extracted *= [1, -1]
X_extracted -= X_extracted.mean(axis=0)

plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_extracted[:, 0], y=X_extracted[:, 1])
plt.axis('equal');

%%time

from sklearn.cluster import KMeans
```

```
kmeans =
KMeans(n_clusters=len(text.split())).fit(X_extracted)
labels = kmeans.predict(X_extracted)

plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_extracted[:, 0], y=X_extracted[:, 1],
hue=labels, palette='plasma')
plt.axis('equal');

x = X_extracted[:, 0]

radius = (x.max() - x.min()) / (2 * np.pi)
phi = (x - x.min()) / (x.max() - x.min()) * (2 * np.pi) /
1.1

x3d = np.cos(phi) * radius
y3d = np.sin(phi) * radius
z3d = X_extracted[:, 1]


plt.figure(figsize=(12, 8))
plt.hsv()
ax = plt.axes(projection='3d')
ax.view_init(50, 35)
ax.scatter3D(x3d, y3d, z3d, s=20);

x = X_extracted[:, 0]
```

```
radius = (x.max() - x.min()) / (2 * np.pi)
phi = (x - x.min()) / (x.max() - x.min()) * (2 * np.pi) /
1.1

k = 2.5
x3d = np.cos(phi * k) * radius * (1 + phi / (2 * np.pi) *
(k))
y3d = np.sin(phi * k) * radius * (1 + phi / (2 * np.pi) *
(k))
z3d = X_extracted[:, 1]


plt.figure(figsize=(12, 8))
plt.hsv()
ax = plt.axes(projection='3d')
ax.view_init(50, 35)
ax.scatter3D(x3d, y3d, z3d, s=20, c=labels);

plt.figure(figsize=(12, 8))
plt.hsv()
ax = plt.axes(projection='3d')
ax.view_init(90, 0)
ax.scatter3D(x3d, y3d, z3d, s=20, c=labels);

Xkm = np.concatenate([x3d.reshape(-1, 1), y3d.reshape(-1,
1), z3d.reshape(-1, 1)], axis=1)
Xkm.shape

kmeans1 = KMeans(n_clusters=len(text.split()))
```

```
kmeans1.fit(Xkm)

labels1 = kmeans1.predict(Xkm)

plt.figure(figsize=(12, 8))
plt.hsv()
ax = plt.axes(projection='3d')
ax.view_init(90, 0)
ax.scatter3D(x3d, y3d, z3d, s=20, c=labels1);

with open('python.npy', 'wb') as f:
        np.save(f, labels)
        np.save(f, x3d)
        np.save(f, y3d)
        np.save(f, z3d)
```