Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДУ РЕКУРЕНТНОЇ ДОСТОВІРНОЇ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ВЕЛИКИХ ДАНИХ З ВИКОРИСТАННЯМ ФУНКЦІЇ НАЛЕЖНОСТІ СПЕЦІАЛЬНОГО ТИПУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-2
Котихін С.Д.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Котихіну Станіславу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та дослідження методу рекурентної достовірної нечіткої кластеризації великих даних з використанням функції належності спеціального типу

затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 4 червня 2022 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, UCI репозиторій дата сет з технічними характеристиками пісень різних жанрів top10s, UCI репозиторій дата сет Iris.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів нечіткої кластеризації та аналіз їх можливостей.
2. Дослідження ефективності застосування функції належності в різних методах нечіткої кластеризації.
3. Дослідження градієнтних алгоритмів оптимізації цільової функції спеціального виду.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми кластеризації даних, постановка задачі, тестові набори даних, лістинг коду програмної реалізації.

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	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-21.04.22	
3	Аналіз літератури з досліджуваної проблеми	22.04.22-25.04.22	
4	Аналіз технічних засобів	26.04.22-30.04.22	
5	Розробка методу	01.05.22-14.05.22	
6	Програмна реалізація	15.05.22-21.05.22	
7	Оформлення пояснювальної записки	21.05.22-31.05.22	
8	Перевірка на плагіат	05.06.22	
9	Рецензування	08.06.22	
10	Підготовка презентації та доповіді	04.06.22-13.06.22	
11	Занесення роботи в електронний архів	04.06.22	
12	Попередній захист кваліфікаційної роботи	13.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____

(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 55 с., 2 табл., 20 рис., 1 дод., 30 джерел.

НЕЧІТКА КЛАСТЕРИЗАЦІЯ, ДОСТОВІРНА НЕЧІТКА КЛАСТЕРИЗАЦІЯ, РІВЕНЬ НАЛЕЖНОСТІ, РІВЕНЬ ДОСТОВІРНОСТІ, МІРА ПОДІБНОСТІ.

Об’єктом роботи є дослідження та розробка методу рекурентної достовірної нечіткої кластеризації.

Метою роботи є розглядання задачі нечіткої кластеризації на основі ймовірнісного, можливісного і достовірного підходів на основі пакетного і online режимів надходження і обробки інформації, введення рекурентної версії достовірного алгоритму та модифікації функції належності.

Запропоновано метод достовірної нечіткої кластеризації для задач, коли дані надходять на обробку або у послідовному онлайн режимі, або формують надвеликі масиви (Big Data). Введені процедури є за суттю градієнтними алгоритмами оптимізації цільової функції спеціального виду, та мають низку переваг перед відомими ймовірнісними та можливісними підходами. Запропоновані процедури є узагальненням відомих методів.

У результаті роботи здійснена програмна реалізація системи, яка визначає до якого кластеру належать дані, які надходять в онлайн та пакетному режимах.

FUZZY CLUSTERING, CREDIBILISTIC FUZZY CLUSTERING, MEMBERSHIP LEVEL, CREDIBILISTIC LEVEL, SIMILARITY MEASURE.

The object of the work is research and development recurrent reliable fuzzy clustering method.

The aim of the research is to develop fuzzy clustering methods based on the probabilistic, possible and reliable approaches based on batch and online modes of information receipt and processing, introduction of a recurrent version of a reliable algorithm and modification of the membership function.

A method of credibilistic fuzzy clustering is proposed for problems when data are fed sequentially, in online mode and forms large arrays (Big Data). The introduced procedures are essentially gradient algorithms for optimizing the objective function of a special type and have a number of advantages over known probabilistic and possible. The proposed procedures are a generalization of the known methods.

As a result of work was implemented software implementation of the system that determine cluster of data row, which is received online and by package.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

FCM – Fuzzy *C*-Means (нечіткі *C*-середні)

GK – Gustafson-Kessel (Густафсон-Кессель)

GG – Gath-Gevva (Гата-Гевва)

PC – Partition Coefficient (коефіцієнт розподілу)

SC – Partition Index (індекс розподілу)

XB – Xie-Beni Index (індекс Ксі-Бені)

ВСТУП

Задача кластеризації (класифікації в режимі самонавчання) багатовимірних даних є важливою частиною інтелектуального аналізу даних (Data Mining), в рамках якої склався ряд напрямків і підходів [1, 2]. Один з таких напрямків утворюють методи нечіткої (фаззі-) кластеризації, в основі яких лежить припущення про те, що класи – кластери, що формуються взаємно перетинаються так, що кожне спостереження – вектор з різними рівнями належності-ймовірності-можливості може належати одночасно до кількох чи всіх класів.

Актуальність роботи полягає у розробці більш ефективнішого та більш якісного методу кластеризації даних, який може значно зекономити час та ресурси. В умовах надвеликої кількості даних, навіть дуже маленький приріст швидкості та скорочення ресурсів може призвести до суттєвої економії грошей на дистанції навіть для компанії з великим обігом.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

1.1 Задача нечіткої кластеризації

На сьогодні актуальною проблемою сучасності є потреба аналізу накопичених даних, які містять корисні знання. З розвитком технології Інтернету речей (Internet of Things), джерелами інформації становляться не тільки люди, а й пристрої (сенсори, смартфони, відеокамери тощо), що підключаються до мережі. Вони формують потоки різнорідних даних, збільшуючи кількість розподілених джерел інформації. Лавиноподібне зростання джерел та обсягів інформації, її різнорідність та розподілений характер зберігання призвели до необхідності перегляду технологій обробки даних.

Одним із важливих завдань аналізу даних є кластеризація [3]. В умовах невизначеності в аналізованому середовищу актуальним є завдання нечіткої кластеризації. Алгоритми нечіткої кластеризації представляють великий інтерес для тих випадків аналізу, коли дані описують об’єкти або процеси умовах невизначеності та реалізують зазвичай ітераційну процедуру наближення до розв’язання задачі.

Кластеризація – це технологія, що дозволяє розподілити вхідні дані на класи – групи однотипних екземплярів вибірки, або кластери – компактні області групування екземплярів вибірки у просторі ознак. Вхідною інформацією для кластеризації є вибірка спостережень:

$$\begin{matrix} x_{11} & x_{12} & \dots & x_{1j} \\ x_{21} & x_{22} & \dots & x_{2j} \\ \dots & \dots & \dots & \dots \\ x_{s1} & x_{s2} & \dots & x_{sj} \end{matrix}, \qquad (1.1)$$

де $s = 1, 2, \dots, S$;

$j = 1, 2, \dots, N$;

S – кількість екземплярів вибірки;

N – кількість ознак, що характеризують екземпляри вибірки.

Задача кластеризації полягає в розбитті об’єктів з x на декілька кластерів, у яких об’єкти більш схожі між собою, ніж з об’єктами інших кластерів. У метричному просторі «схожість» звичайно визначають через відстань.

Нечіткі методи кластерного аналізу дозволяють будь-якому екземпляру одночасно належати до всіх визначених кластерів, але з різним ступенем.

Вимога знаходження однозначної кластеризації елементів досліджуваної проблемної області є досить грубою і жорсткою, особливо при вирішенні задач системного аналізу, що слабо структуруються. Методи нечіткої кластеризації послабляють цю вимогу. Послаблення вимоги здійснюється за рахунок введення в розгляд нечітких кластерів і відповідних їм функцій приналежності, що набувають значень з інтервалу [0, 1].

В загальному випадку завданням нечіткої кластеризації є знаходження нечіткого розбиття множини елементів досліджуваної сукупності, які утворюють структуру нечітких кластерів, присутніх у вхідних даних. Це завдання зводиться до знаходження мір приналежності елементів універсуму шуканим нечітким кластерам, які в сукупності і визначають нечітке розбиття вихідної множини елементів (рис. 1.1).

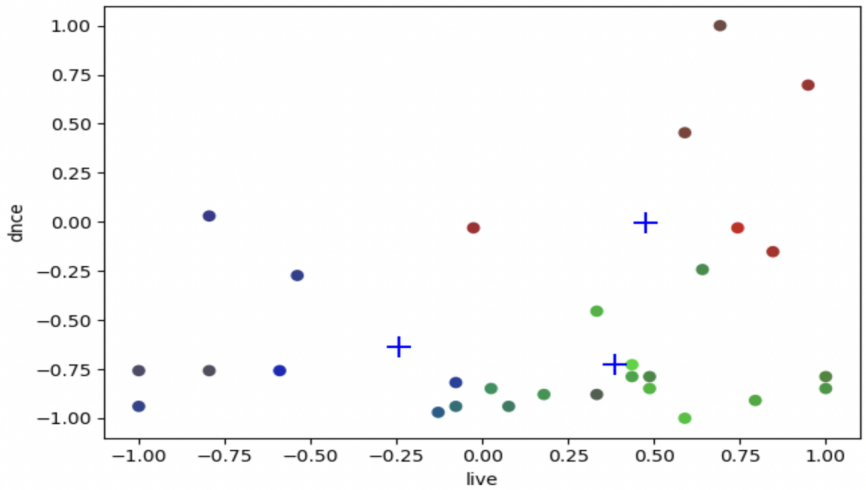

Рисунок 1.1 – Приклад нечіткої кластеризації

1.2 Огляд існуючих методів нечіткої кластеризації

1.2.1 Метод нечітких С-середніх

З розвитком нечіткої теорії, алгоритм *C*-середніх (FCM), який фактично заснований на теорії кластеризації нечітких Руспіні, був запропонований у 1981 році Джеймсом Бездеком [4, 5], як розширення методу кластеризації чітких *C*-середніх. FCM [6] – це неконтрольований алгоритм кластеризації, який застосовується до широкого діапазону проблем, пов'язаних з аналізом ознак, кластеризацією та дизайну класифікатора, наприклад в сільському господарстві, інженерії, астрономії, хімії, геології, аналізу зображень, медичній діагностиці, аналізу форм та розпізнаванні цілей.

Цей алгоритм використовується для аналізу, на основі відстані між різними вхідними даними точки. Кластери формуються відповідно до відстані між точками даних і центрами кластерів та підраховуються для кожного окремого кластеру.

У результаті набір даних групується в n кластерів з кожною одиницею даних, пов'язаний з кожним кластером в певній мірі. Високий ступінь належності до кластера матимуть точки, що лежать недалеку від геометричного центру кластеру, та низький ступінь належності матимуть точки що лежать далеко від центру.

FCM ітеративно вимірює якість розбиття шляхом порівняння відстані від шаблону x, до поточного кандидату кластерного центру w з відстанню від цього шаблону до інших центрів кластерів-кандидатів. Алгоритм фокусується на мінімізації цільової функції. Цільова функція – це оптимізаційна функція, яка обчислює зважену внутрішньогрупову суму квадратів помилки (WGSS) наступним чином:

$$E_\beta(U,W) = \sum_{k=1}^{N}\sum_{q=1}^{m} U_{qk}^{\beta} D_{qk}^{2}, \quad (1.2)$$

де N – кількість вхідних даних;

m – кількість кластерів;

U – матриця належності точок до певних кластерів;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру;

D_{qk} – дистанція між точкою k та центром кластеру;

W – центри кластерів;

β – параметр фаззіфікації, що задає «розмитість» границь кластерів.

Евклідова дистанція може бути порахована за формулою:

$$D_{qk} = \| x_k - w_q \|, \tag{1.3}$$

де x_k– координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру.

Алгоритм має такі обмеження:

$$U_{qk} \in [0, 1], \tag{1.4}$$

де $q = 1, 2, \ldots m$;

$k = 1, 2, \ldots N$.

$$\sum_{k=1}^{N} U_{qk} = 1, \tag{1.5}$$

де $q = 1, 2, \ldots m$.

$$0 < \sum_{q=1}^{m} U_{qk} < m, \tag{1.6}$$

де $k = 1, 2, \ldots N$.

Цільова функція (1.2) описує задачу оптимізації з обмеженнями, яку можна перетворити на задачу без обмежень за допомогою множника Лагранжа:

$$U_{qk}^{\tau} = \frac{1}{\sum_{i=1}^{m}\left(\frac{D_{qk}}{D_{ik}}\right)^{\frac{2}{(\beta-1)}}}, \tag{1.7}$$

де $k = 1, \ldots, N$;

$q = 1, \ldots, m$;

τ – епоха або ітерація.

$$W_{qk}^{\tau} = \frac{\sum_{k=1}^{N}(U_{qk}^{\tau-1})^{\beta}\, x_k}{\sum_{k=1}^{N}(U_{qk}^{\tau-1})^{\beta}}, \tag{1.8}$$

де $k = 1, \ldots, N$;

$q = 1, \ldots, m$;

τ – епоха або ітерація.

FCM починається з набору початкових центрів кластерів, які можуть визначатися випадково, або задаватися заздалегідь. Потім повторюється дві функції оновлення (1.7) та (1.8) до тих пір, поки центри кластерів не будуть стабільними або цільова функція в (1.2) сходитися до локального мінімуму.

Алгоритм дуже простий за своєю реалізацією, що є одним з головних його плюсів, але можливості цього підходу обмежуються ймовірнісними обмеженнями на рівні належності так, що «забруднені» збуреннями і викидами спостереження можуть бути віднесені до різних класів з практично однаковими рівнями належності.

1.2.2 Метод Густафсона-Кесселя

Запропонований у 1979 році Алгоритм Густафсона-Кесселя [7] є варіацією алгоритма нечітких *C*-середніх. У ньому використовується інша, яка є адаптивною, норма відстані для розпізнавання геометричних форм в даних.

Цей алгоритм допускає кожен кластер як точку, так і матрицю, відповідно, представляючи центр кластера та його коваріацію. У той час як алгоритм нечітких с-середніх висувають неявну гіпотезу, що кластери сферичні, алгоритм Густафсона-Кесселя не підпадає під це обмеження і може ідентифікувати еліпсоїдні кластери.

Алгоритм Густафсона-Кесселя, як і FCM, фокусується на мінімізації цільової функції. У цьому випадку її можна записати як:

$$E_\beta(U, W) = \sum_{k=1}^{N} \sum_{q=1}^{m} U_{qk}^{\beta} D_{qkA_k}^2, \quad (1.9)$$

де N – кількість вхідних даних;

m – кількість кластерів;

U – матриця належності точок до певних кластерів;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру;

D_{qkA_k} – дистанція між точкою k та центром кластеру, яка рахується з коваріаційною матрицею;

W – центри кластерів;

β – параметр фаззіфікації, що задає «розмитість» границь кластерів.

Кожен кластер матиме власну нормо-породжуючу матрицю A_k. Матриця A_k використовується як параметр оптимізації у функціоналі *C*-середніх. Це означає, що для кожного кластера норма відстані може адаптуватися відповідно до локальної топологічної структури даних та відповідає наступному виразу, породженому скалярним добутком:

$$D_{qkA} = (x_k - w_q)^T A_k (x_k - w_q), \quad (1.10)$$

де x_k – координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру;

A_k – коваріаційна матриця.

Підрахування коваріаційної матриці A_k для кожного кластеру S_q за допомогою множників Лагранжа можна виразити наступним чином:

$$A_k = \sqrt{det\,(S_q)}\, S_q, \quad (1.11)$$

де S_q:

$$S_q = \frac{\sum_{k=1}^{N} U_{qk}^{\beta} (x_k - w_q)^T (x_k - w_q)}{\sum_{k=1}^{N} U_{qk}^{\beta}}, \quad (1.12)$$

де x_k – координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру;

β – параметр фаззіфікації, що задає «розмитість» границь кластерів.

Цільова функція залишилась незмінною, порівняно з алгоритмом FCM, тому її перетворення у задачу без обмежень за допомогою множників Лагранжа таке ж саме:

$$U_{qk}^{\tau} = \frac{1}{\sum_{i=1}^{m} \left(\frac{D_{qk}}{D_{ik}}\right)^{\frac{2}{(\beta-1)}}}, \quad (1.13)$$

де $k = 1, \ldots, N$;

$q = 1, \ldots, m$;

τ – епоха або ітерація.

Великою перевагою алгоритму Густафсона-Кесселя є здатність виділяти не тільки сферичні, а й еліпсовидні кластери, що значно підвищує точність кластеризації, але в той же час цей алгоритм ще більше «забруднений» збуреннями і нестійкий до викидів спостережень. До того ж він складніший і потребує більше часу та ресурсів для роботи.

1.2.3 Метод Гата-Геви

Гат та Гева у 1989 [8] припустили, що алгоритм кластеризації нечітких оцінок максимальної правдоподібності (FMLE) можна використовувати для виявлення кластерів різної форми, розміру та щільності. Алгоритм використовує норму відстані, засновану на нечітких оцінках максимальної правдоподібності.

Цей алгоритм є варіацією алгоритму Густафсона-Кесселя, та при підрахуванні дистанції між точками(норми відстані) має експоненціальний член. Це означає, що дана норма відстані буде зменшуватися швидше, ніж норма, що породжується скалярним твором.

Формула норми відстані у алгоритмі Гата-Геви має такий вигляд:

$$D_{qk} = \left(\frac{\sqrt{det\,(S_{\beta k})}}{\alpha_q}\right)^{\left(\frac{1}{2}(x_k - w_q)^T S_{wk}^{-1}(x_k - w_q)\right)}, \qquad (1.14)$$

де $k = 1, \ldots, N$;

$q = 1, \ldots, m$;

N – кількість вхідних даних;

m – кількість кластерів;

x_k – координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру;

$S_{\beta k}$ – нечітка коваріаційна матриця;

α_q – вірогідність вибору кластеру q.

Нечітка коваріаційна матриця обчислюється за формулою, дуже схожою з аналогічною формулою у алгоритмі Густафсона-Кесселя:

$$S_q = \frac{\sum_{k=1}^{N} U_{qk}^{\beta} (x_k - w_q)^T (x_k - w_q)}{\sum_{k=1}^{N} U_{qk}^{\beta}}, \quad (1.15)$$

де x_k – координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру;

β – параметр фаззіфікації, що задає «розмитість» границь кластерів.

Причиною використання змінної β є узагальнення цього виразу. В оригінальному алгоритмі FMLE $\beta = 1$. Приймається, що значення змінної $m = 2$ для компенсації експоненційного члена та отримання більш нечітких кластерів. Через узагальнення утворюються дві виважені підступні матриці.

Змінна α_q у рівнянні (1.14), яка є вірогідність вибору кластеру q, обчислюється за такою формулою:

$$\alpha_q = \frac{1}{N} \sum_{k=1}^{N} U_{qk}, \quad (1.16)$$

де N – кількість вхідних даних;

U_{qk} – значення матриці належності точок до певних кластерів, де j-й елемент належить до i-го кластеру;

$q = 1, \ldots, m$.

Цільова функція та рівняння матриці належності у алгоритмі Гата-Геви залишились незмінними порівняно з двома алгоритмами, розглянутими вище:

$$E_\beta(U,W) = \sum_{k=1}^{N}\sum_{q=1}^{m} U_{qk}^{\beta} D_{qk}^2, \quad (1.17)$$

де N – кількість вхідних даних;

m – кількість кластерів;

U – матриця належності точок до певних кластерів;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру;

D_{qk} – дистанція між точкою k та центром кластеру;

W – центри кластерів;

β – параметр фаззіфікації, що задає «розмитість» границь кластерів.

Та перетворюючи на задачу без обмежень отримуємо:

$$U_{qk}^{\tau} = \frac{1}{\sum_{i=1}^{m}\left(\frac{D_{qk}}{D_{ik}}\right)^{\frac{2}{(\beta-1)}}}, \quad (1.18)$$

де $k = 1, \ldots, N$;

$q = 1, \ldots, m$;

τ – епоха або ітерація.

Алгоритм Гата-Геви буде швидший ніж FCM або GK, особливо коли кластерів небагато, а також точніше ніж FCM і, у деяких випадках, навіть ніж GK. Але головним недоліком цього алгоритму є нестійкість, оскільки експонентна норма відстані може сходитися в локальний оптимум.

1.3 Методи порівняння якості кластеризації

1.3.1 Коефіцієнт розподілу

Бездек запропонував коефіцієнт розподілу (PC), який вимірює величину перекриття між кластерами. Чим ближче до одиниці PC, тим менше розподіл

векторів у наборі даних між різними кластерами. Чим ближче значення PC до m^{-1}, тим нечіткішою є кластеризація. Значення, близьке до m^{-1}, вказує на те, що набір даних не має структури кластеризації, або прийнятий алгоритм кластеризації не зміг її розгадати.

Коефіцієнт розподілу виявляється як:

$$K_p = \frac{\sum_{k=1}^{N}\sum_{q=1}^{m} U_{qk}}{N}, \quad (1.19)$$

де N – кількість вхідних даних;

m – кількість кластерів;

U – матриця належності точок до певних кластерів;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру.

Як видно з формули (1.19), єдиним аргументом даного критерію є матриця належності.

З визначення матриці належності відомо, що:

$$U_{qk} \in [\,0,1\,], \quad (1.20)$$

де $q = 1, 2, \ldots m$;

$k = 1, 2, \ldots N$.

$$\sum_{k=1}^{N} U_{qk} = 1, \quad (1.21)$$

де $q = 1, 2, \ldots m$.

Враховуючи це, нескладно показати, що цей критерій досягає мінімуму $U_{qk}= \text{m}^{-1}$, $q = 1, ..., m$. Це випадок найбільшої невизначеності: всі елементи вхідної множини з рівним ступенем належать кожному з кластерів. Максимум даного критерію досягається на межі області визначення ($U_{ip} = 1$, $U_{iq} = 0$, p,

q=1, …, m, $p \neq q$), причому критерій дорівнює одиниці, що відповідає максимально чіткому розбиттю.

1.3.2 Індекс розподілу

Показник розподілу або індекс розподілу (SC) визначається як відношення між загальною мінливістю всередині кластера та загальною відстанню між кластерами. Відповідно до цього показника, хороша кластеризація даних призводить до низької мінливості всередині кожного кластера та великої відстані між різними кластерами. Для обчислення загальної мінливості всередині кластера мінливість у кожному кластері визначається як середня відстань між кожною парою даних у кластері, а потім усереднюється для всіх кластерів. Відстань між кластерами отримують усередненням усіх попарних відстаней між кластерами. У свою чергу, кожна одинична відстань між кластерами обчислюється шляхом усереднення відстані між кожною парою даних із двох кластерів.

1.3.3 Індекс Ксі-Бені

Ксі і Бені ввели метод індексу Ксі-Бені (XB) [9] у 1991 році. Індекс XB – фокусується на розподільності та компактності. Розподільність – це міра відстані між одним кластером та іншим кластером, а компактність – це міра близькості між точками даних в кластері. Відповідно до цього методу оптимальна кластеризація – це значення з найменшим значенням XB.

Формула індексу Ксі-Бені виглядає так:

$$V_{XB} = \frac{\sum_{k=1}^{N}\sum_{q=1}^{m} U_{qk}^2 \left\|x_k - w_q\right\|^2}{n min_{q,k}\left\|x_k - w_q\right\|^2}, \tag{1.22}$$

де N – кількість вхідних даних;

m – кількість кластерів;

x_k – координати k-ї точки вхідних даних;

w_q – координати центру q-го кластеру;

U_{qk} – значення матриці належності точок до певних кластерів, де k-й елемент належить до q-го кластеру.

1.4 Постановка задачі

У зв'язку з проблемами наведених вище проблем алгоритмів нечіткої кластеризації Рагху Крішнапурамом та Джеймсом Келлером [10] був запропонований можливісний підхід до нечіткої кластеризації (PCM) більш стійкий до шумів і збурень. Разом з тим PCM-алгоритми страждають від, так званої, проблеми співпадіння, коли в процесі обробки інформації деякі кластери починають зливатися один з одним, що в результаті веде до невірної оцінки кількості цих кластерів.

Цих недоліків позбавлені алгоритми достовірної нечіткої кластеризації [11-14], засновані на апараті теорії вірогідності [15]. В рамках цього підходу в процесі розрахунків оцінюються як рівні нечіткої належності, так і рівні довіри, засновані на мірі належності спеціального вигляду [16]. Результати експериментів показали [17-19], що достовірний підхід забезпечує більш високу якість кластеризації в порівнянні з ймовірнісними і можливісними методами.

Таким чином, задача кластеризації вирішується в пакетному режимі, коли весь масив даних обробляється багаторазово на основі почергового оцінювання.

Якщо ж дані надходять на обробку у вигляді потоку або утворюють надвеликі масиви, що досить часто зустрічаються у реальних ситуаціях, пакетний режим не дозволяє ефективно вирішити розглянуту задачу.

У цій ситуації найбільш ефективними є рекурентні процедури нечіткої кластеризації, що дозволяють вирішувати завдання в online режимі і уточнювати отримані рішення по мірі надходження кожного нового спостереження. Так, у [15, 16] були запропоновані рекурентні варіанти FCM, які є за суттю градієнтними процедурами оптимізації прийнятої цільової функції, а в [20, 21] були введені рекурентні модифікації PCM, що призначені для послідовної обробки даних.

У зв’язку з цим є доцільною розробка рекурентної модифікації методу достовірної нечіткої кластеризації, що дозволяє уточнювати шукані характеристики кластерів по мірі надходження кожного нового спостереження.

Об’єктом роботи є дослідження та розробка методу рекурентної достовірної нечіткої кластеризації.

Метою роботи є розглядання задачі нечіткої кластеризації на основі ймовірнісного, можливісного і достовірного підходів на основі пакетного і online режимів надходження і обробки інформації, введення рекурентної версії достовірного алгоритму та модифікації функції належності.

Для досягнення мети необхідно вирішити такі завдання:

- провести аналіз існуючих методів нечіткої кластеризації;
- розробити метод рекурентної модифікації на основі достовірної нечіткої кластеризації;
- реалізувати алгоритм рекурентної достовірної нечіткої кластеризації;
- реалізувати комп’ютерну модель для кластеризації даних за допомогою методу рекурентної достовірної нечіткої кластеризації.

2 МАТЕМАТИЧНА МОДЕЛЬ РЕКУРЕНТНОГО АЛГОРИТМУ ДОСТОВІРНОЇ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

2.1 Вихідна інформація для рішення

Вихідною інформацією для рішення задачі нечіткої кластеризації є масив n-вимірних векторів спостережень:

$$X = \{x_1, x_2, \ldots, x_N\} \subset R^n, \tag{2.1}$$

де $x(k) \in X$;

$k = 1, \ldots, N$;

Масив повинен бути розбитий на заздалегідь заданих m класів-кластерів з деяким рівнем належності-можливості-достовірності $U_q(k)$ – k-го вектора x до q-го кластеру.

Тобто:

$$(1 < m < N, 1 \leq q \leq m). \tag{2.2}$$

Необхідно також відзначити, що вихідні дані попередньо повинні бути предоброблені (нормовані) так, щоб вони відповідали :

$$-1 \leq x_{ki} \leq 1 \, (1 \leq i \leq n), \tag{2.3}$$

де x_{ki} – i-та компонента k-го вектора x.

2.2 Метод ймовірнісної нечіткої кластеризації

Найбільш популярний метод ймовірнісної нечіткої кластеризації пов'язаний з мінімізацією цільової функції [6]:

$$E(U_{qk}, w_q) = \sum_{k=1}^{N} \sum_{q=1}^{m} U_{qk}^{\beta} D^2(x_{k,} w_q). \quad (2.4)$$

За обмежень:

$$\sum_{k=1}^{N} U_{qk} = 1, \quad (2.5)$$

$$0 < \sum_{q=1}^{m} U_{qk} < m, \quad (2.6)$$

де N – кількість вхідних даних;

m – кількість кластерів;

U_{qk} – рівень нечіткої належноті x_k спостереження до q-го кластера, тобто елемент матриці належності U;

$\beta > 1$ – параметр фаззіфікації, що задає «розмитість» границь кластерів;

$Cl_q(1 \le q \le m)$, w_q – прототип-центроїд q-го кластера;

$D(x_k, w_q)$ – відстань між x_k та w_q у прийнятій метриці. У нашому випадку це просто скалярна відстань;

Вирішуючи задачу нелінійного програмування за допомогою методу невизначених множників Лагранжа, приходимо до відомого результату:

$$U_{qk}^{(\tau+1)} = \left(D^2(x_k, w_q^{\tau})\right)^{\frac{1}{(1-\beta)}} \times \left(\sum_{i=1}^{m} (D^2(x_k, w_i^{\tau}))^{\frac{1}{(1-\beta)}}\right)^{-1}, \quad (2.7)$$

де $\tau = 1, 2, \ldots$ – індекс епохи обробки інформації в режимі почергового оцінювання.

$$w_{qk}^{\tau+1} = \frac{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta} x_k}{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta}}. \quad (2.8)$$

При цьому процес обчислень триває до виконання умови зупинки:

$$w_q^{(\tau+1)} - w_q^{(\tau)} \leq \varepsilon \; \forall \; 1 \leq q \leq m, \quad (2.9)$$

де ε – наперед заданий поріг точності обчислень.

У випадку $\beta = 2$ та евклідової метрики $D^2(x_k, w_q) = \left\| x_k - w_q \right\|_2^2$ приходимо до популярного алгоритму нечітких *C*-середніх (FCM) [13] вигляду:

$$\mathrm{U}_{\mathrm{qk}}^{(\tau+1)} = \left(\left\| \mathrm{x_k} - \mathrm{w_q^{\tau}} \right\|_2^2 \right)^{-2} \times \left(\sum_{\mathrm{i}=1}^{\mathrm{m}} (\| \mathrm{x_k} - \mathrm{w_i^{\tau}} \|_2^2)^{-2} \right)^{-1}, \quad (2.10)$$

$$w_{qk}^{\tau+1} = \frac{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta} x_k}{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta}}. \quad (2.11)$$

Якщо дані надходять на обробку послідовно в онлайн режимі, задача нелінійного програмування може бути вирішена за допомогою алгоритму Ерроу-Гурвіца-Удзави, що є за суттю градієнтною процедурою пошуку сідлової точки функції Лагранжа на основі критерію (2.4) з обмеженнями на суму належностей.

При цьому співвідношення (2.7), (2.8) можуть бути переписані у формі:

$$\begin{cases} U_{q(k+1)} = \left(D^2\left(x_{k+1}, w_q^k\right) \right)^{\frac{1}{(1-\beta)}} \times \\ \times \left(\sum_{i=1}^{m} \left(D^2\left(x_{k+1}, w_i^k\right) \right)^{\frac{1}{(1-\beta)}} \right)^{-1}, \\ w_{q(k+1)} = w(k) + \eta(k+1) U_{q(k+1)}^{\beta} \left(x_{k+1} - w_{qk} \right), \end{cases} \quad (2.12)$$

де $\eta(k)$ – параметр кроку навчання.

А співвідношення (2.10), (2.11) можуть бути переписані так:

$$\begin{cases} U_{q(k+1)} = \left\|x_k - w_q^k\right\|^{-2} \times \\ \times \left(\sum_{i=1}^{m} \left\|x_k - w_i^k\right\|^{-2}\right)^{-1}, \\ w_{q(k+1)} = w(k) + \eta(k+1) U_{q(k+1)}^2 \left(x_{k+1} - w_{qk}\right). \end{cases} \quad (2.13)$$

Це ж і є узагальненням рекурентних процедур Парка-Деггера [16] і Чанга-Лі [22].

2.3 Можливісний алгоритм нечіткої кластеризації

Можливісні алгоритми нечіткої кластеризації також засновані на мінімізації цільової функції, але такого виду [15]:

$$E(U_{qk}, w_q, \mu_q) = \sum_{k=1}^{N} \sum_{q=1}^{m} U_{qk}^{\beta} D^2(x_k, w_q) + \\ + \sum_{q=1}^{m} \mu_q \sum_{k=1}^{N} (1 - U_{qk})^{\beta}, \quad (2.14)$$

де $\mu_q \leq 0$ – визначає відстань, на якій рівень належності приймає значення 0.5, тобто $U(k) = 0$, якщо $D(x, w) = \mu$.

Мінімізація критерію (2.14) дозволяє отримати аналітичний розв'язок у вигляді:

$$U_{qk}^{\tau} = \left(1 + \left(\frac{D^2(x_k, w_q^{\tau})}{\mu_q^{\tau}}\right)^{\frac{1}{\beta - 1}}\right)^{-1}, \quad (2.15)$$

$$w_q^{\tau+1} = \frac{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta} x_k}{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta}}, \quad (2.16)$$

$$\mu_q^{\tau} = \sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta} D^2(x_k, w_q^{\tau}) \sum_{k=1}^{N} (U_{qk}^{\tau+1})^{\beta -1}. \quad (2.17)$$

Яке у квадратичному випадку набуває вигляду:

$$U_{qk}^{\tau+1} = (1 + \frac{\|x_k - w_q^{\tau}\|^2}{\mu_q^{\tau}})^{-1}, \quad (2.18)$$

$$w_q^{\tau+1} = \frac{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^2 x_k}{\sum_{k=1}^{N} (U_{qk}^{\tau+1})^2}, \quad (2.19)$$

$$\mu_q^{\tau+1} = \sum_{k=1}^{N} (U_{qk}^{\tau+1})^2 \|x_k - w_q^{\tau+1}\|^2 \sum_{k=1}^{N} (U_{qk}^{\tau+1})^2 {}^{-1}. \quad (2.20)$$

Онлайн версії формул (2.15)-(2.20) при цьому мають вигляд:

$$\begin{cases} U_{q(k+1)} = (1 + (\frac{D^2(x_k, w_q^{\tau})}{\mu_q^{\tau}})^{\frac{1}{\beta-1}} \\ w_{q(k+1)} = w(k) + \eta(k+1) U_{q(k+1)}^{\beta} (x_{k+1} - w_{qk}), \\ \mu_{q(k+1)} = \sum_{p=1}^{k+1} U_{qp}^{\beta} D^2(x_p, w_{q(k+1)}) \sum_{p=1}^{k+1} (U_{qp}^{\beta})^{-1}, \end{cases} \quad (2.21)$$

та у випадку $\beta = 2$ евклідової метрики:

$$\begin{cases} U_{q(k+1)} = (1 + \frac{\|x_k - w_q^{\tau}\|^2}{\mu_q^{\tau}})^{-1} \\ w_{q(k+1)} = w(k) + \eta(k+1) U_{q(k+1)}^{2} (x_{k+1} - w_{qk}), \\ \mu_{q(k+1)} = \sum_{p=1}^{k+1} U_{qp}^{2} \|x_k - w_{q(k+1)}\|^2 \sum_{p=1}^{k+1} (U_{qp}^{2})^{-1}. \end{cases} \quad (2.22)$$

2.4 Метод достовірної нечіткої кластеризації

Достовірна нечітка кластеризація пов’язана з мінімізацією такої цільової функції:

$$E(U_{qk}, w_q) = \sum_{k=1}^{N} \sum_{q=1}^{m} Cr_{qk}^{\beta} D^2(x_{k,} w_q), \quad (2.23)$$

за обмежень для будь-яких q та k, для яких $Cr_{qk} \geq 0{,}5$:

$$\begin{aligned} 0 \leq Cr_{qk} \leq \varepsilon \; \forall \; q, k, \\ supCr_{qk} \geq 0{,}5 \; \forall \; k, \\ Cr_{qk} + supCr_{qk} = 1, \end{aligned} \quad (2.24)$$

де Cr_{qk} – рівень нечіткої достовірності того, що спостереження x_k належить кластеру q.

При цьому рівень достовірності розраховується на основі функції належності [23]:

$$U_k = \gamma\,(D(x, w)), \quad (2.25)$$

де $\gamma_q(.)$ – монотонно зменшується на інтервалі [0,∞];

$\gamma_q(0) = 1$;

$\gamma_q(\infty) \rightarrow 0$.

Нескладно помітити, що функція (2.25) є за суттю мірою подібності, заснованій на відстані [24].

В якості такої функції у [25] було запропоновано використовувати вираз:

$$U_k = (1 + D(x, w)). \quad (2.26)$$

Цей вираз описує за суттю звичайну дзвонувату функцію належності, яка використовується в системах нечіткого висновування.

Опираючись на всі вище перераховані формули та фактори, можна зауважити, що формулу розрахунку матриці належності (2.4) можна переписати у вигляді:

$$U_{qk} = \left(D^2(x_k, w_{qk})\right)^{\frac{1}{(1-\beta)}} \times \left(\sum_{i=1}^{m}(D^2(x_k, w_{ik}))^{\frac{1}{(1-\beta)}}\right)^{-1} =$$

$$= (D^2(x_k, w_{kk}))^{\frac{1}{(1-\beta)}} \left(D^2(x_k, w_{qk})\right)^{\frac{1}{(1-\beta)}} + \left(\sum_{i=1}^{m}(D^2(x_k, w_{ik}))^{\frac{1}{(1-\beta)}}\right)^{-1} =$$

$$= (1 + \left(D^2(x_k, w_{qk})\right)^{\frac{1}{(1-\beta)}} \sum_{i=1}^{m}(D^2(x_k, w_{ik}))^{\frac{1}{(1-\beta)}})^{-1}. \quad (2.27)$$

Для евклідової метрики $\boldsymbol{\beta} = 2$ ця формула (2.27) приймає вигляд функції щільності розподілу Коші з параметром ширини σ_q^2 [26]:

$$\mathrm{U_{qk}} = (1 + \frac{\left\|\mathrm{x_k} - \mathrm{w_{qk}}\right\|^2}{\sigma_q^2})^{-1}, \quad (2.28)$$

$$\sigma_q^2 = (\sum_{i=1}^{m}\|x_k - w_{ik}\|^{-2})^{-1}. \quad (2.29)$$

Тож функція належності (2.26) є окремим випадком формули (2.28) при $\sigma_q^2 = 1$.

Остаточно пакетний алгоритм достовірної нечіткої кластеризації може бути записаний у вигляді [26, 27]:

$$U_{qk}^{\tau+1} = (1 + D^2\left(x_k, w_q^{\tau}\right))^{-1}, \quad (2.30)$$

$$U *_{qk}^{\tau+1} = U_{qk}^{\tau+1}(supU_{ik}^{\tau+1})^{-1}, \quad (2.31)$$

$$Cr_{qk}^{(\tau+1)} = \frac{1}{2}(U *_{qk}^{\tau+1} + 1 - supU_{ik}^{*}), \quad (2.32)$$

$$w_q^{\tau+1} = \frac{\sum_{k=1}^{N} (Cr_{qk}^{(\tau+1)})^{\beta} x_k}{\sum_{k=1}^{N} (Cr_{qk}^{(\tau+1)})^{\beta}}. \quad (2.33)$$

І тепер, на підставі вище згаданих формул (2.23) - (2.24), (2.28) - (2.33) можна ввести у розгляд онлайн рекурентну версію алгоритму достовірної нечіткої кластеризації [28, 29] у вигляді:

$$\begin{cases} \sigma_{q(k+1)}^2 = (\sum_{i=1}^{m} \|x_k - w_{ik}\|^{-2})^{-1}, \\ U_{q(k+1)} = (1 + \frac{\|x_k - w_{qk}\|^2}{\sigma_{q(k+1)}^2})^{-1}, \\ U_{k+1}^* = U_{q(k+1)} (sup U_{i(k+1)})^{-1}, \\ Cr_{q(k+1)} = \frac{1}{2}(U_{q(k+1)}^* + 1 - sup U_{i(k+1)}^*), \\ w_{q(k+1)} = w(k) + \eta(k+1) Cr_{q(k+1)}^{\beta} (x_{k+1} - w_{qk}). \end{cases} \quad (2.34)$$

Як можна побачити, рекурентний алгоритм достовірної нечіткої кластеризації не сильно складніший стандартних рекуретних алгоритмів FCM та PCM, але при цьому зберігає усі переваги достовірного підходу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РЕКУРЕНТНОГО АЛГОРИТМУ ДОСТОВІРНОЇ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

3.1 Обґрунтування вибору середовища програмної реалізації

Як було зазначено в попередніх розділах, метою роботи є створення методу рекурентної достовірної нечіткої кластеризації. Для того щоб перевірити його роботу на практиці, був розроблений десктопний застосунок.

Застосунок повинен приймати дані, причому як у офлайн (великим масивом даних), так і в онлайн режимі – по одному запису для подальшої кластеризації. Також програма повинна рисувати графіки наприкінці роботи, які будуть візуалізувати результат роботи кластеризації.

Виходячи з поставленого завдання була обрана мова програмування Python 3.9 з використанням бібліотек pandas, sklearn, matplotlib. Visual Studio Code було обрано як середовище під час виконання.

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Динамічна типізація та особливості синтаксису мови роблять її код дуже лаконічним, порівнюючи з конкурентами, що значно прискорює швидкість розробки.

Також програми на мові Python можна запускати на будь-якої операційній системі без змін у програмному коді, що є великою перевагою для десктопних застосунків.

Ще однією величезною перевагою мови Python є велика кількість інструментів для роботи з великими даними. Ці інструменти багатофункціональні та спрощують і отримання, і обробку, і їх візуалізацію. Як наслідок – Python це найпопулярніша мова для Machine Learning та Data Science інженерів.

Треба зазначити, що одним з найголовніших недоліків цієї мови є швидкість виконання коду. Це обумовлено декількома факторами, серед них –

динамічна типізація, не моментальна (JIT) компіляція та інше. Але в нашому випадку швидкість виконання коду не важливий фактор, а ось швидкість написання коду та компактність коду – це гарна перевага.

Pandas – програмна бібліотека мовою Python для обробки та аналізу даних. Надає спеціальні структури даних та операції для маніпулювання числовими таблицями та тимчасовими рядами.

Sklearn – це безкоштовна програмна бібліотека машинного навчання, яка надає функціональність для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації.

Matplotlib – бібліотека мовою програмування Python для візуалізації даних двовимірною графікою.

Visual Studio Code – це популярний редактор вихідного коду. Особливість цього редактору в тому, що він дуже швидкий та не заточений на якусь конкретну мову програмування, а може налаштовуватися під будь-яку за допомогою розширень. Працює на будь-якій операційній системі.

Таким чином, для створення десктопного застосунку для класифікації методом рекурентної достовірної нечіткої кластеризації були вибрані всі вищезазначені інструменти.

3.2 Етапи розроблення застосунку для класифікації методом рекурентної достовірної нечіткої кластеризації

Першим етапом у розробці застосунку буде робота з вихідними даними. Треба мати на увазі, що даних може бути дуже багато, тому треба зробити введення та обробку найефективнішим способом. До того ж, застосунок повинен підтримувати як онлайн, так і пакетний варіант методу рекурентної достовірної нечіткої кластеризації. Це означає, що і введення даних має відбуватися двома способами.

На основі вищезгаданих умов була застосована Python-бібліотека Pandas. Самі дані надходять у застосунок у форматі csv.

Формат csv – це такий формат, коли дані між собою розділяються комами чи точками з комами. Перший рядок – це зазвичай заголовки колонок. В нашому випадку, в першому рядку через крапку з комою повинні бути назви ознак, за якими буде проводитися порівняння та класифікація. Кожен наступний рядок – одиниця даних, яку треба кластеризувати, із своїми значеннями ознак. Приклад, як може виглядати вхідні дані у форматі csv є на рисунку 3.1.

```
1   title;artist;top genre;bpm;nrgy;dnce;dB;live;dur;acous;spch
2   Hey, Soul Sister;Train;neo mellow;97;89;67;-4;8;217;19;4
3   Love The Way You Lie;Eminem;detroit hip hop;87;93;75;-5;52;263;24;23
4   TiK ToK;Kesha;dance pop;120;84;76;-3;29;200;10;14
5   Bad Romance;Lady Gaga;dance pop;119;92;70;-4;8;295;0;4
6   Just the Way You Are;Bruno Mars;pop;109;84;64;-5;9;221;2;4
7   Baby;Justin Bieber;canadian pop;65;86;73;-5;11;214;4;14
8   Dynamite;Taio Cruz;dance pop;120;78;75;-4;4;203;0;9
9   Secrets;OneRepublic;dance pop;148;76;52;-6;12;225;7;4
```

Рисунок 3.1 – Вхідні дані у форматі csv

В обох варіантах – і в пакетному і в онлайн режимі дані повинні надходити саме в такому варіанті. Лістинг коду для завантаження навчальної вибірки top10s приведено на рисунку 3.2.

```
df = pd.read_csv('top10s.csv', sep=';')
```

Рисунок 3.2 – Лістинг коду для завантаження навчальної вибірки

Після завантаження даних, вони потребують обробки. По-перше, треба позбавитися від нечислових значень в ознаках, тому що за умовами метод обробляє тільки числові значення.

По-друге, всі числові значення треба привести в єдину шкалу, для того щоб їх можна було порівнювати. Це називається нормалізація. Як було

зазначено у попередньому розділі, усі значення будуть нормалізуватися у проміжок від -1 до 1. Сама нормалізація була виконана за допомогою Python-бібліотеки Sklearn.

Функція обробки даних приведено на рисунку 3.3.

```
def normalize_input_data(data: DataFrame):
    new_data = copy(data)
    # clear any string features, remain only numeric
    for col_name in new_data.columns:
        if any(type(value) is str for value in data[col_name]):
            new_data.drop(col_name, axis = 1, inplace = True)
    # delete id if exist
    new_data.drop('Id', axis=1, inplace=True, errors='ignore')
    # make normalization in range -1 to 1
    for feature_name in new_data.columns:
        column_feature = new_data[[feature_name]]
        scaler = MinMaxScaler((-1, 1)).fit_transform(column_feature)
        new_data.drop(column_feature, axis = 1, inplace = True)
        new_data[feature_name] = scaler.flatten()
    return new_data
```

Рисунок 3.3 – Функція обробки даних

Наступний етап – це побудова матриці належності та визначення центрів кластерів. Початкові центри кластерів обираються випадково, а потім уточнюються по ходу роботи алгоритму. Лістинг коду для цього етапу приведений на рисунках 3.4 – 3.7.

```
def random_cluster_centers(self):
    centers = []
    for i in range(self.n_clusters):
        coordinates = []
        for c in range(self.n_features):
            point = random.uniform(-1, 1)
            coordinates.append(point)
        centers.append(coordinates)
    self.cluster_centers_ = np.array(centers)
    return centers
```

Рисунок 3.4 – Функція початкового вибору центрів кластерів

```
def update_membership(self, X):
    """
    update the membership matrix
    :param X: data points
    :return: nothing

    For performance, the distance can be computed once, before the loop instead of computing it every time
    """
    for i in range(self.n_points):
        for c in range(len(self.cluster_centers_)):
            self.u[i][c] = self.compute_membership_single(X, i, c)
```

Рисунок 3.5 – Функція підрахування матриці залежності

```
def compute_membership_single(self, X, datapoint_idx, cluster_idx):
    """
    :param datapoint_idx:
    :param cluster_idx:
    :return: return computer membership for the given ids
    """
    clean_X = X
    d1 = self.distance_squared(clean_X[datapoint_idx], self.cluster_centers_[cluster_idx])
    sum1 = 0.0
    for c in self.cluster_centers_:  # this is to compute the sigma
        d2 = self.distance_squared(c, clean_X[datapoint_idx])
        if d2 == 0.0:
            d2 = SMALL_VALUE
        sum1 += (d1/d2) ** (1.0/(self.m-1))
        if np.any(np.isnan(sum1)):
            self.logger.debug("nan is found in compute_membership_single")
            self.logger.debug("d1: %s" % str(d1))
            self.logger.debug("sum1: %s" % str(sum1))
            self.logger.debug("d2: %s" % str(d2))
            self.logger.debug("c: %s" % str(c))
            self.logger.debug("X[%d] %s" % (datapoint_idx, str(clean_X[datapoint_idx])))
            self.logger.debug("centers: %s" % str(self.cluster_centers_))
            raise Exception("nan is found in computer_memberhip_single method in the inner for")
    if sum1 == 0:  # because otherwise it will return inf
        return 1.0 - SMALL_VALUE
    if np.any(np.isnan(sum1 ** -1)):
        self.logger.debug("nan is found in compute_membership_single")
        self.logger.debug("d1: %s" % str(d1))
        self.logger.debug("sum1: %s" % str(sum1))
        self.logger.debug("X[%d] %s" % (datapoint_idx, str(clean_X[datapoint_idx])))
        self.logger.debug("centers: %s" % str(self.cluster_centers_))
        raise Exception("nan is found in computer_memberhip_single method")
    return sum1 ** -1
```

Рисунок 3.6 – Функція підрахування 1 елемента матриці залежності

```
def compute_cluster_centers(self, X):
    """
    :param X:
    :return:

    vi = (sum of membership for cluster i ^ m  * x ) / sum of membership for cluster i ^ m  : for each cluster i

    """
    centers = []
    for c in range(self.n_clusters):
        sum1_vec = np.zeros(self.n_features)
        sum2_vec = 0.0
        for i in range(self.n_points):
            interm1 = (self.u[i][c] ** self.m)
            interm2 = interm1 * X[i]
            sum1_vec += interm2
            sum2_vec += interm1
            if np.any(np.isnan(sum1_vec)):
                self.logger.debug("compute_cluster_centers> interm1 %s" % str(interm1))
                self.logger.debug("compute_cluster_centers> interm2 %s" % str(interm2))
                self.logger.debug("compute_cluster_centers> X[%d] %s" % (i, str(X[i])))
                self.logger.debug("compute_cluster_centers> loop sum1_vec %s" % str(sum1_vec))
                self.logger.debug("compute_cluster_centers> loop sum2_vec %s" % str(sum2_vec))
                self.logger.debug("X: [%d] %s" % (i-1, X[i-1]))
                self.logger.debug("X: [%d] %s" % (i+1, X[i+1]))
                self.logger.debug("X: ")
                self.logger.debug(X)
                raise Exception("There is a nan in compute_cluster_centers method if")
        if sum2_vec == 0:
            sum2_vec = 0.000001
        centers.append(sum1_vec/sum2_vec)
    self.cluster_centers_ = np.array(centers)
    return centers
```

Рисунок 3.7 – Функція підрахування центру кластеру

У процесі ітеративного перерахування центрів кластерів та матриці належності застосунок виводить проміжні значення цих елементів у консоль. Так було зроблено для того, щоб можна було наглядно слідкувати як «навчається» алгоритм. Також наприкінці навчання, виводиться кількість ітерацій, яка була витрачена на досягнення точного результату. Приклад виводу проміжного значення та кількість затрачених ітерацій наведені на рисунку 3.8.

```
updated membership is:
[[0.4240182  0.4142797  0.16170209]
 [0.35739864 0.36161319 0.28098816]
 [0.43518606 0.44413313 0.12068081]
 ...
 [0.40443599 0.41012815 0.18543587]
 [0.38476303 0.38238268 0.23285429]
 [0.41258941 0.40369949 0.1837111 ]]
updated cluster centers are:
[[ 0.17226439  0.51314825  0.35812975  0.88563039 -0.51815355 -0.38141819
  -0.81864087 -0.67032328]
 [ 0.17383069  0.51256546  0.35640119  0.88551856 -0.5087749  -0.38038767
  -0.81731986 -0.66755713]
 [ 0.12258767  0.14278882  0.2463457   0.83804575 -0.54947637 -0.29899695
  -0.23713688 -0.67501712]]
took 93 iterations
```

Рисунок 3.8 – Приклад виводу проміжного значення та кількість затрачених ітерацій

Наступний етап – перевірка достовірності і перерахунок матриці належності. Може так вийти, що ця перевірка не вплине на остаточний результат. Але все ж таки частіше після неї результат змінюється в кращу сторону. Це буде доказано в одному з наступних розділів з порівняльними тестами. Функція перевірки достовірності приведений на рисунку 3.9.

```
def credibilistic_recalculation(self):
    max_credibiliscic = self.u.max()
    for i in range(len(self.u)):
        for j in range(len(self.u[i])):
            self.u[i][j] = (self.u[i][j] + 1 - max_credibiliscic) / 2
```

Рисунок 3.9 – Функція перевірки достовірності

Та останній етап застосунку – виведення результатів роботи. Головною задачею було вивести велику кількість даних та не заплутати користувача. Тому було вирішено виводити результати одразу у двух варіантах – у вигляді таблиць, та у вигляді графіків залежності між різними ознаками. Для того щоб користувач міг вибрати більш зручний для себе варіант. Графіки були нарисовані за допомогою Python бібліотеки matplotlib.

Для того, щоб графіки могли бути відкриті на кожному комп’ютері без зайвих програм, вони виводяться у html файл. Лістинг коду для виведення результату у вигляді таблиць приведений на рисунку 3.10, а для малювання графіків на рисунку 3.11.

```
print("\n\Original data")
print(df)
print("\n\nNormalized data")
print(normal_data)
print("\nCluster centers after FCM")
print(fcm.cluster_centers_)
draw_model_2d(fcm, test_data,'Fcm', headers)
fcm.credibilistic_recalculation()
fcm.compute_cluster_centers(test_data)

df["Cluster"] = [np.argmax(row) + 1 for row in fcm.u]
print("\n\Original data with result")
print(df)
print("\nCluster centers after credibilistic")
print(fcm.cluster_centers_)
```

Рисунок 3.10 – Лістинг коду для виведення результату у вигляді таблиць

```
def draw_model_2d(fcm, test_data, name, headers):
    rgb_spectre = fcm.u.tolist()
    for i in range(len(rgb_spectre)):
        if len(rgb_spectre) == 2:
            rgb_spectre[i].append(0)
    comb = list(combinations(range(0, len(fcm.cluster_centers_[0])), 2))
    transponate_input = test_data.transpose()
    all_combination_features = list(combinations(transponate_input, 2))
    fig, axes = plt.subplots(len(all_combination_features), 2, figsize=(16, 5 * len(all_combination_features)))
    plt.subplots_adjust(top=0.99)
    fig.suptitle(f'{name} fuzzy clustering result', fontsize=16, y=1)
    i = 0
    for features_to_compare, centers_i in zip(all_combination_features, comb):
        axes[i, 0].scatter(features_to_compare[:][0], features_to_compare[:][1], alpha=1)
        axes[i, 0].set_ylabel(headers[centers_i[0]])
        axes[i, 0].set_xlabel(headers[centers_i[1]])
        axes[i, 1].scatter(features_to_compare[:][0], features_to_compare[:][1], c = rgb_spectre, alpha=1)
        axes[i, 1].set_ylabel(headers[centers_i[0]])
        axes[i, 1].set_xlabel(headers[centers_i[1]])
        for cluster in fcm.cluster_centers_:
            axes[i, 1].scatter(cluster[centers_i[0]], cluster[centers_i[1]], marker="+", s=150, c='b')
        i += 1
    tmpfile = BytesIO()
    fig.savefig(tmpfile, format='png')
    encoded = base64.b64encode(tmpfile.getvalue()).decode('utf-8')

    html = '<img src=\'{}\'>'.format(encoded)

    with open(f'{name}.html','w') as f:
        f.write(html)
```

Рисунок 3.11 – Лістинг коду для малювання 2D графіків

3.3 Інструкція користувача

Кожен користувач має змогу запустити застосунок на своєму комп'ютері, не дивлячись на те, яка в нього операційна система. Для цього йому треба встановити декілька додатків на комп'ютер.

В першу чергу треба встановити Python. Рекомендується встановлювати версію 3.9 та вище, але будь-яка версія вища за 3 буде працювати. Сам Python можна скачати та установити з різних джерел – офіційного сайту python.org, у магазині додатків Microsoft Store, якщо операційна система Windows, або ж через додаток Anaconda. Останній варіант є найзручнішим, адже він надає інтерфейс роботи з бібліотеками Python, про які буде розказано трохи пізніше.

Як вже зазначалося у попередніх розділах, для реалізації програмного застосунку були застосовані додаткові бібліотеки, які не включені у базовий набір бібліотек мови Python. Їх можна встановити через інтерфейс додатку Anaconda, або ж через консоль. Список всіх потрібних бібліотек буде наданий у файлі requirements.txt у додатку, а також на рисунку 3.12. Лістинг коду установки усіх бібліотек, зазначених у файлі requirements.txt наданий на рисунку 3.13.

requirements.txt

```
appdirs==1.4.4
argh==0.26.2
boto==2.49.0
entrypoints==0.3
et-xmlfile==1.1.0
fonttools==4.25.0
mkl-fft==1.3.1
mkl-service==2.4.0
mpmath==1.2.1
nltk==3.6.5
numpy==1.20.3
pandas==1.3.4
pathspec==0.7.0
patsy==0.5.2
pep8==1.7.1
Pillow==8.4.0
pkginfo==1.7.1
ply==3.11
pycosat==0.6.3
pycurl==7.44.1
pytz==2021.3
PyYAML==6.0
scikit-image==0.18.3
scikit-learn==0.22.1
simplegeneric==0.8.1
unicodecsv==0.14.1
```

Рисунок 3.12 – Файл requirements.txt

```
pip install -r requirements.txt
```

Рисунок 3.13 – Лістинг коду установки усіх бібліотек, зазначених у файлі requirements.txt

Після цього треба взяти вибірку даних у форматі csv, та загрузити в каталог з програмою. Можна також узяти навчальну вибірку, яка буде зазначена у додатку. Назва файлу повинна мати розширення «.csv».

У файлі «main.py» на 10 рядку потрібно відредагувати назву файлу вибірки. Якщо була вибрана навчальна вибірка, можна залишити це за замовчуванням. Приклад правильного вигляду рядка, при умові використання файлу навчальної вибірки «top10s.csv» [30] наданий на рисунку 3.14.

```
df = pd.read_csv('top10s.csv', sep=';')
```

Рисунок 3.14 – Лістинг коду для завантаження навчальної вибірки «top10s»

На цьому вся підготовча робота завершена і застосунок готовий до роботи. Для того щоб запустити програму, треба виконати консольну команду, знаходячись у каталозі з кодом застосунку, надану на рисунку 3.15.

```
python  main.py
```

Рисунок 3.15 – Лістинг коду консольної команди для запуску застосунку

Як вже було зазначено, результат роботи програми виводиться у двох варіантах – у порівняльних таблицях (рис. 3.16) та у графіках (рис 3.17). До того, було також вирішено виводити результати роботи методу *C*-середніх, щоб користувач міг сам побачити та оцінити ефективність методу рекурентної достовірної нечіткої кластеризації.

Графіки, як зазначалося, виводяться у файли html, для того щоб користувач міг подивитися їх на любій операційній системі.

Після закінчення роботи, застосунок генерує два файли: «Fcm.html» – графіки результатів роботи методу *C*-середніх, «Credibilistic.html» – графіки результатів роботи методу рекурентної достовірної нечіткої кластеризації.

```
Normalized data
          bpm      nrgy      dnce        dB      live       dur     acous      spch
0   -0.058252  0.816327  0.395833  0.931034 -0.783784 -0.427586 -0.616162 -0.833333
1   -0.155340  0.897959  0.562500  0.896552  0.405405 -0.110345 -0.515152 -0.041667
2    0.165049  0.714286  0.583333  0.965517 -0.216216 -0.544828 -0.797980 -0.416667
3    0.436893  0.551020  0.083333  0.862069 -0.675676 -0.372414 -0.858586 -0.833333
4    0.436893  0.693878  0.000000  0.862069  0.027027 -0.462069 -1.000000 -0.833333
..        ...       ...       ...       ...       ...       ...       ...       ...
432  0.009709  0.428571  0.229167  0.862069 -0.081081 -0.420690 -0.595960 -0.875000
433 -0.097087 -0.081633  0.458333  0.827586 -0.567568 -0.124138 -0.757576 -0.375000
434  0.475728  0.673469  0.500000  0.896552 -0.027027 -0.806897 -0.737374 -0.791667
435 -0.067961  0.632653  0.750000  0.931034 -0.837838 -0.455172 -0.676768 -0.041667
436 -0.087379  0.775510  0.541667  0.965517 -0.891892 -0.675862 -0.656566 -0.791667

[437 rows x 8 columns]

Cluster centers after FCM
[[ 0.17226439  0.51314825  0.35812975  0.88563039 -0.51815355 -0.38141819
  -0.81864087 -0.67032328]
 [ 0.17383069  0.51256546  0.35640119  0.88551856 -0.5087749  -0.38038767
  -0.81731986 -0.66755713]
 [ 0.12258767  0.14278882  0.2463457   0.83804575 -0.54947637 -0.29899695
  -0.23713688 -0.67501712]]

\Original data with result
                                          title           artist        top genre  bpm  nrgy  dnce  dB  live  dur  acous  spch  Cluster
0                                Hey, Soul Sister            Train      neo mellow   97    89    67  -4     8  217     19     4        1
1                            Love The Way You Lie           Eminem  detroit hip hop   87    93    75  -5    52  263     24    23        2
2                                         TiK ToK            Kesha        dance pop  120    84    76  -3    29  200     10    14        2
3                                         Secrets       OneRepublic        dance pop  148    76    52  -6    12  225      7     4        1
4                                          Animal        Neon Trees        indie pop  148    83    48  -6    38  212      0     4        2
..                                            ...               ...              ...  ...   ...   ...  ..   ...  ...    ...   ...      ...
432                              Call You Mine  The Chainsmokers       electropop  104    70    59  -6    34  218     20     3        2
433                No Guidance (feat. Drake)       Chris Brown        dance pop   93    45    70  -7    16  261     12    15        3
434                Antisocial (with Travis Scott)       Ed Sheeran              pop  152    82    72  -5    36  162     13     5        2
435  Taki Taki (feat. Selena Gomez, Ozuna & Cardi B)          DJ Snake  electronic trap   96    80    84  -4     6  213     16    23        1
436                           Con Calma - Remix      Daddy Yankee            latin   94    87    74  -3     4  181     17     5        1

[437 rows x 12 columns]

Cluster centers after credibilistic
[[ 0.17091053  0.49725112  0.35263083  0.88337237 -0.51568674 -0.3770543
  -0.79547431 -0.66704275]
 [ 0.17206487  0.49667135  0.35128339  0.88326745 -0.50868011 -0.37624242
  -0.7942554  -0.66495077]
 [ 0.13693287  0.23320809  0.27164265  0.84895284 -0.53128038 -0.31656841
  -0.38512601 -0.6629137 ]]
```

Рисунок 3.16 – Приклад виведення результатів у табличному вигляді

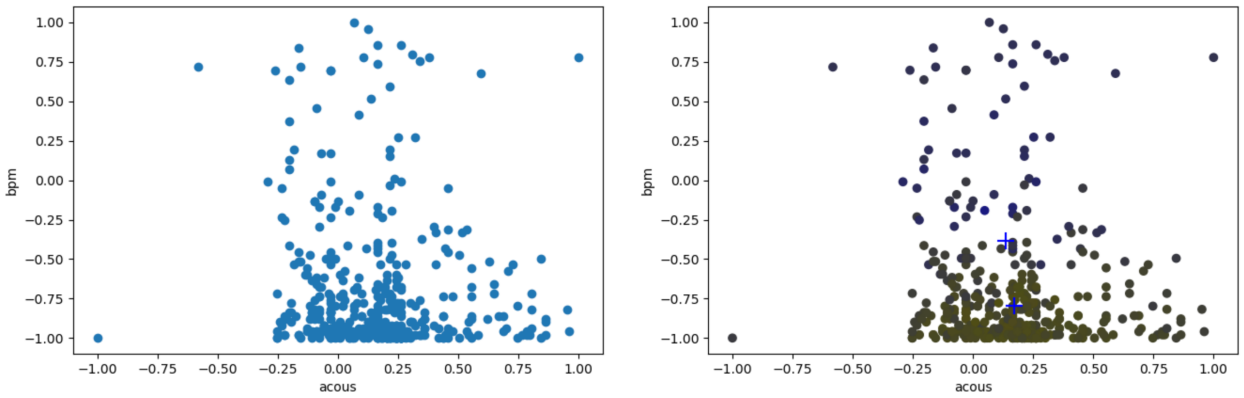

Рисунок 3.17 – Приклад виведення результатів у графічному вигляді

3.4 Тестування розробленої моделі

Для того щоб перевірити ефективність, оцінити працездатність та спроможність якісно кластеризувати велику кількість даних запропонованого методу, було проведено експериментальне дослідження на двох різних наборах даних.

Перший набір даних – це мабуть найпопулярніший набір для кластеризації Iris. Інший набір [30] сильно більший та складніший за кількістю характеристик – набір найпопулярніших пісень останнього десятиріччя з їх технічними характеристиками.

Був проведений порівняльний аналіз якості за основними характеристиками кластеризації, такими як: коефіцієнт розподілу(PC), індекс розподілу(SC), індекс Ксі-Бені(XB).

Також «конкуренти» були обрані серед найпопулярніших алгоритмів кластеризації, а саме: метод нечітких *C*-середніх(FCM), алгоритм Густафсона-Кесселя(GK), алгоритм Гата-Геви.

Результати експериментального дослідження та порівняльного аналізу зазначених алгоритмів із зазначеними характеристиками з набором Iris наведені у таблиці 3.1, а з набором пісень у таблиці 3.2 відповідно.

Таблиця 3.1 – Порівняльна оцінка якості кластеризації нечітких методів кластеризації з використанням набору даних Iris

Методи кластеризації даних	PC	SC	XB
FCM	0,5	1,62	0,19
Густафсон-Кессель	0,27	1,66	1,62
Гат-Гева	0,25	1,54	1,35
Рекурентна нечітка достовірна кластеризація	0,21	1,13	0,01

Таблиця 3.2 – Порівняльна оцінка якості кластеризації нечітких методів кластеризації з використанням набору даних «top10s»

Методи кластеризації даних	PC	SC	XB
FCM	0,5	1,62	0,18
Густафсон-Кессель	0,27	1,66	1,61
Гат-Гева	0,25	1,54	1,35
Рекурентна нечітка достовірна кластеризація	0,23	1,22	0,01

Після аналізу та оцінки отриманих результатів експериментального дослідження можна зробити висновки, що запропонований метод рекурентної достовірної нечіткої кластеризації має гарну ефективність та досить точні результати кластеризації, що підтверджується експериментально.

На рисунках 3.18 та 3.19 більш наочно продемонстровані результати вище проведеного експерименту.

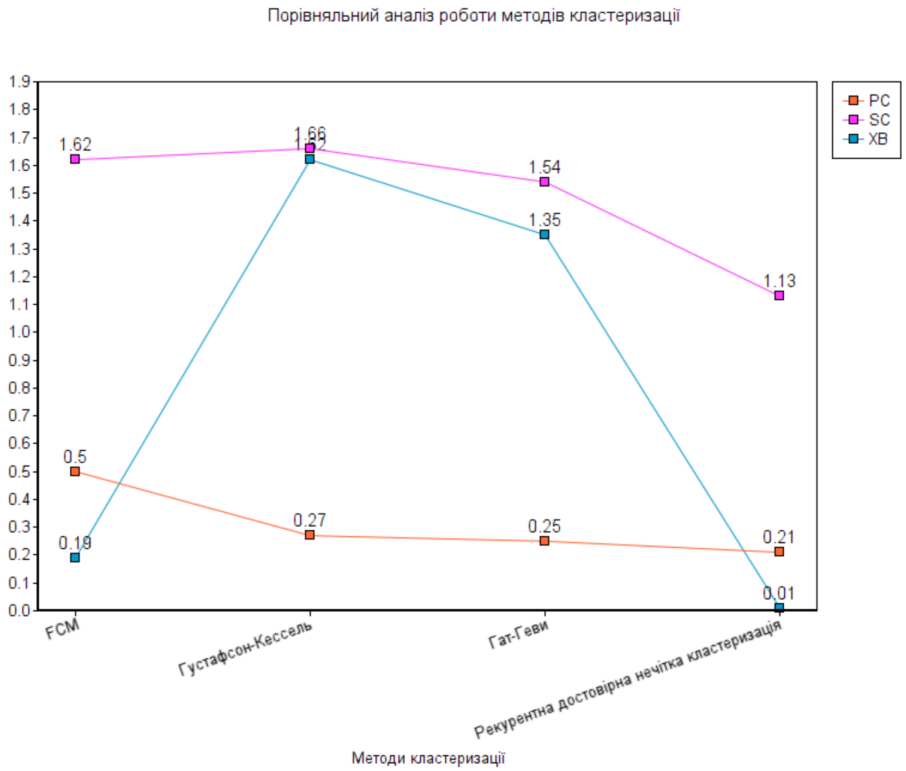

Рисунок 3.18 – Графік порівняльних оцінок якості кластеризації нечітких методів кластеризації з використанням набору даних Iris

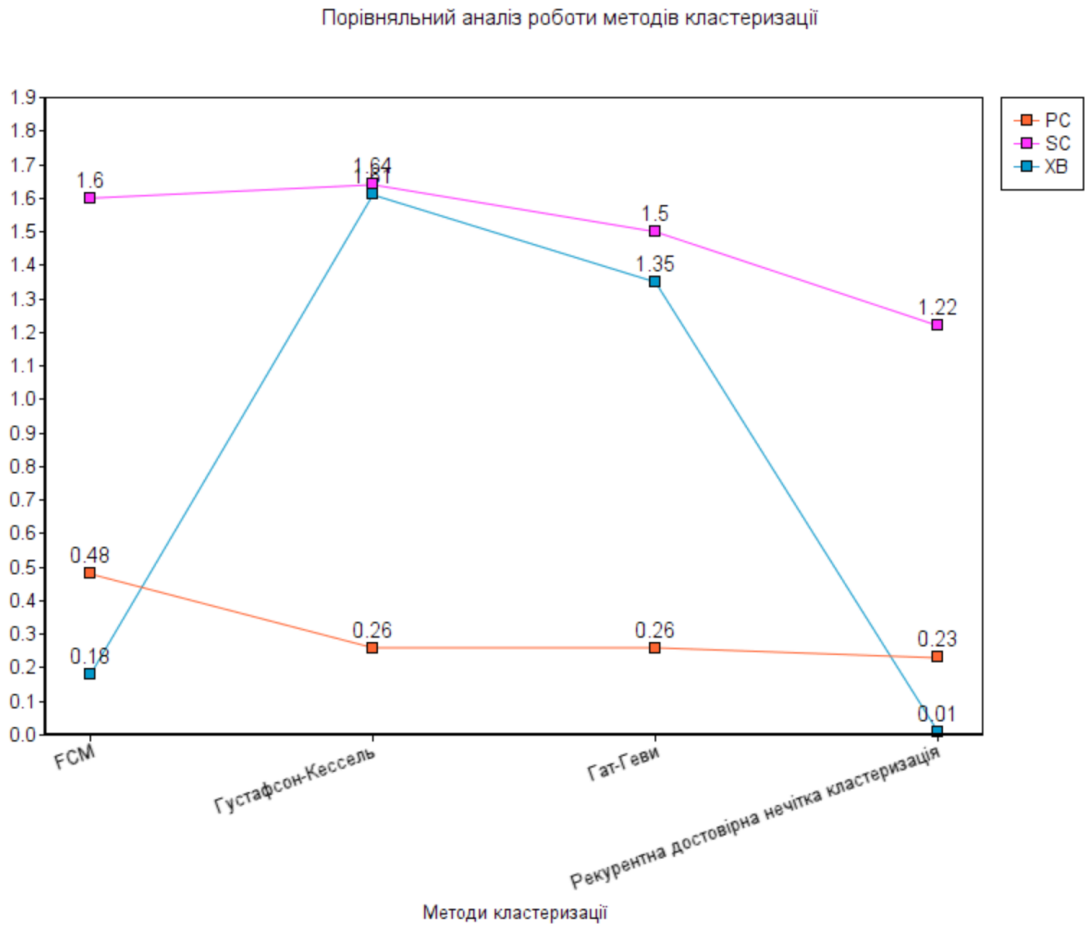

Рисунок 3.19 – Графік порівняльних оцінок якості кластеризації нечітких методів кластеризації з використанням набору «top10s»

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод нечіткої кластеризації, що базуються на використанні міри подібності, параметри якої визначаються автоматично у процесі самонавчання.

У ході роботи було розглянуто задачу нечіткої кластеризації на основі ймовірнісного, можливісного і достовірного підходів на основі пакетного і online режимів надходження і обробки інформації.

Введена рекурентна версія достовірного алгоритму, що є за суттю процедурою градієнтної оптимізації прийнятого критерія нечіткої достовірної кластеризації.

Введена модифікація функції належності, що є по сутті мірою подібності та узагальненням відомих раніше функцій.

Запропонований метод рекурентної достовірної нечіткої кластеризації великих даних з використанням функції належності спеціального типу є доволі простим в чисельній реалізації і призначений для вирішення завдань, що виникають у рамках інтелектуального аналізу великих даних.

Розроблений застосунок, який реалізує запропонований метод рекурентної достовірної нечіткої кластеризації і наочно демонструє його ефективність перед конкурентами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Xu, R., & Wunsch, D. C. (2009). II, Clustering. Hoboken. NJ: Wiley/IEEE Press, 6, 583-617. Aggarwal C.C. Data Mining: Text Book. Springer, 2015.

2. Bezdek, J. C. (2013). Pattern recognition with fuzzy objective function algorithms. Springer Science & Business Media.

3. Кластерні процедури класифікації. URL: https://posibniki.com.ua/post-klasterni-proceduri-klasifikaciyi-ierarhichni-klaster-proceduri (дата звернення 06.05.2022).

4. Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. Computers & geosciences, 10(2-3), 191-203.

5. Höppner, F., & Klawonn, F. (2013). Fuzzy-Clusteranalyse: Verfahren für die Bilderkennung, Klassifizierung und Datenanalyse. Springer-Verlag.

6. 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.

7. Алгоритм Густафсона-Кесселя. URL: https://studref.com/616260/informatika/algoritm_gustafsona_kesselya (дата звернення 05.05.2022).

8. Gath, I., & Geva, A. B. (1989). Unsupervised optimal fuzzy clustering. IEEE Transactions on pattern analysis and machine intelligence, 11(7), 773-780.

9. Xie, X. L., & Beni, G. (1991). A validity measure for fuzzy clustering. IEEE Transactions on pattern analysis and machine intelligence, 13(8), 841-847.

10. Krishnapuram, R., & Keller, J. M. (1993). A possibilistic approach to clustering. IEEE transactions on fuzzy systems, 1(2), 98-110.

11. Chintalapudi, K. K., & Kam, M. (1998, May). A noise-resistant fuzzy c means algorithm for clustering. In 1998 IEEE international conference on fuzzy systems proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36228) (Vol. 2, pp. 1458-1463). IEEE.

12. Zhou, J., Wang, Q., Hung, C. C., & Yi, X. (2015). Credibilistic clustering: the model and algorithms. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 23(04), 545-564.

13. Zhou, J., Wang, Q., Hung, C. C., & Yang, F. (2017). Credibilistic clustering algorithms via alternating cluster estimation. Journal of Intelligent Manufacturing, 28(3), 727-738.

14. Liu, B., & Liu, Y. K. (2002). Expected value of fuzzy variable and fuzzy expected value models. IEEE transactions on Fuzzy Systems, 10(4), 445-450.

15. Liu, B. (2006). A survey of credibility theory. Fuzzy optimization and decision making, 5(4), 387-408.

16. Park, D. C., & Dagher, I. (1994, June). Gradient based fuzzy c-means (GBFCM) algorithm. In Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94) (Vol. 3, pp. 1626-1631). IEEE.

17. Bodyanskiy Ye, Shafronenko A., Mashtalir S., Online robust fuzzy clustering of data with omissions using similarity measure of special type - Lecture Notes in Computational Intelligence and Decision Making-Cham: Springer, 2020-P.637-646.

18. Shafronenko, A., Bodyanskiy, Y., Pliss, I., & Popov, S. (2020, September). Evolving neo-fuzzy system for distorted data online processing. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 352-355). IEEE.

19. Shafronenko, A., & Bodyanskiy, Y. V. (2020). Adaptive fuzzy clustering approach based on evolutionary cat swarm optimization. In CMIS (pp. 832-842).

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

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

22. Chung, F. L., & Lee, T. (1994). Fuzzy competitive learning. Neural Networks, 7(3), 539-551.

23. Zhou, J., & Hung, C. C. (2007). A generalized approach to possibilistic clustering algorithms. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 15(supp02), 117-138.

24. Young F.W., & Hamer R.M. (1994). Theory and Applications of Multidimensional Scaling-Hillsdale, N.J.: Erlbaum.

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

26. Shafronenko, A. Y., & Rudenko, D. A. (2020). ONLINE RECURRENT METHOD OF CREDIBILISTIC FUZZY CLUSTERING. BBK 91, 37.

27. Bodyanskiy, Y. V., Shafronenko, A. Y., Rudenko, D. A., & Klymova, I. N. (2020). Online Recurrent Method Of Credibilistic Fuzzy Clustering.

28. Bodyanskiy, Y. V., Shafronenko, A. Y., & Klymova, I. N. (2021). ONLINE FUZZY CLUSTERING OF INCOMPLETE DATA USING CREDIBILISTIC APPROACH AND SIMILARITY MEASURE OF SPECIAL TYPE. Radio Electronics. Computer Science, Control, 1(1), 97-104.

29. Shafronenko, A., Bodyanskiy, Y., & Rudenko, D. (2020). Neuro-fuzzy clustering of Distorted Data Using Cat Swarm Optimization. LAP LAMBERT Academic Publishing.

30. Датасет з найпопулярнішими піснями минулого десятиріччя. URL: https://www.kaggle.com/datasets/leonardopena/top-spotify-songs-from-20102019-by-year?select=top10s.csv (дата звернення 28.04.2022).