НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

**КУРСОВА РОБОТА**

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Кластеризація в машинному навчанні»

Студента 2 курсу групи ІТ-02

Спеціальності: 121

«Інженерія програмного забезпечення»

Підгорного Владислава Володимировича

«ПРИЙНЯВ» з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис                    Дата

Київ - 2022 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІТ-02 Семестр 4

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Підгорного Владислава |

|  |  |
| --- | --- |
| 1.Тема роботи | Кластеризація в машинному навчанні |
|  | |
|  | |

|  |  |
| --- | --- |
| 2.Строк здачі студентом закінченої роботи | 19.06.2022 |

|  |  |
| --- | --- |
| 3. Вхідні дані до роботи | методичні вказівки до курсової роботи, обрані дані з сайту |
| https://www.kaggle.com/datasets/grosvenpaul/family-income-and-expenditure | |
| https://scikit-learn.org/stable/user\_guide.html | |
| https://matplotlib.org/stable/users/index | |
|  | |

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

|  |
| --- |
| 1.Постановка задачі |
| 2.Обгрунтування вибору методів інтелектуального аналізу даними |
| 3.Застосування та порівняння ефективності методів інтелектуального аналізу даними для поставленого завдання |
|  |
|  |

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

|  |  |
| --- | --- |
|  | |
|  | |
|  | |
|  | |
| 6.Дата видачі завдання | 16.04.2022 |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Підписи керівника, студента |
| 1. | Отримання теми курсової роботи | 16.04.2022 |  |
| 2. | Визначення зовнішніх джерел даних | 24.04.2022 |  |
| 3. | Пошук та вивчення літератури з питань курсової роботи | 28.04.2022 |  |
| 4. | Розробка моделі сховища даних |  |  |
| 4. | Розробка ETL процесів |  |  |
| 5. | Обґрунтування методів інтелектуального аналізу даних | 06.05.2022 |  |
| 6. | Застосування та порівняння ефективності методів інтелектуального аналізу даних | 16.05.2022 |  |
| 7. | Підготовка пояснювальної записки | 29.05.2022 |  |
| 8. | Здача курсової роботи на перевірку |  |  |
| 9. | Захист курсової роботи | 19.06.2022 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Підгорний Владислав Володимирович |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| Керівник |  |  | доц. Ліхоузова Т.А |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |
| Керівник |  |  | доц. Олійник Ю.О. |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

"26" червня 2022 р.

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: 71 сторінка, 45 рисунків, 10 посилань.

Об’єкт дослідження: кластеризація.

Предмет дослідження: логічне порівняння алгоритмів кластеризації.

Мета роботи: обробка даних логічного порівняння роботи алгоритмів кластеризації, а також прикладне використання алгоритмів на сховищі даних.

Дана курсова робота включає в себе: аналіз даних сховища, опис використання програмного забезпечення для інтелектуального аналізу даних, їх графічне відображення та логічне порівняння моделей.

ЗМІСТ

ЗМІСТ…………………………………………………………………………2

[ВСТУП](#Вступ)………………………………………………………………………...4

1.[ПОСТАНОВКА ЗАДАЧІ](#ПОСТАНОВКА_ЗАДАЧІ)......….....………………………………………...5

1.1. [Огляд предметної області](#Огляд_предметної_області)……………………………………………….5

1.2. [Огляд доступних джерел даних](#ОГЛЯД_ДОСТУПНИХ_ДЖЕРЕЛ_ДАНИХ)………………………………………...6

1.3. [Постановка задачі](#ПОСТАНОВКА_ЗАДАЧІ)………………………………………………………..7

2.[ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИМИ](#Обгрунтування_Вибору_Методів)………………………………………………………………..8

3. [ЗАСТОСУВАННЯ ТА ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИМИ ДЛЯ ПОСТАВЛЕНОГО ЗАВДАННЯ](#Застосування_та_порівняння) …………………………………………………………………………10

3.1 [Визначення моделей та методів, що можуть бути використані](#Визначення_Моделей)……….10

3.2 [Вибір ознак, що будуть використані для аналізу](#Вибір_ознак)………………………11

3.3 [K-means](#k_means) …………………………………………………………………...17

3.4 [Hierarchal clustering](#Hierarchical_clustering) ………………………………………………………24

3.5 [Fuzzy C-means](#Fuzzy_C_means) ……………………………………………………………28

3.6 [Кластеризація з більш ніж двома параметрами](#кластер_з_більш_2)…………………………32

3.7 [Висновки щодо якості побудованих моделей](#Висновки_щодо) ………………………….44

[ВИСНОВКИ](#висн)…………………………………………………………………..48

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ](#списоквик)…………………………………….49

[ДОДАТОК А](#додаток_А)………………………………………………………………….51

ВСТУП

Кластеризація, або кластерний аналіз — це статистична процедура, задача якої полягає в розбитті вибірки об'єктів на підмножини, що не перетинаються і називаються кластерами.

Кожен кластер має складається зі схожих об'єктів, а об'єкти різних кластерів мають істотно відрізнятися один від одного.

Задача кластеризації відноситься до статистичної обробки, а також до широкого класу задач навчання без вчителя. Ще її можна описати через задачу класифікації.

Застосування кластеризації можна знайти у великій кількості доменів для неконтрольованого навчання.

Кластеризація також може бути використана для підвищення точності контрольованого алгоритму машинного навчання. Хоча це легко реалізувати, потрібно подбати про деякі критичні аспекти, як-от обробку викидів у даних та забезпечення того, щоб кожен кластер мав достатню кількість населення. Все це входить до складу сертифікації машинного навчання.

1. ПОСТАНОВКА ЗАДАЧІ
   1. Огляд предметної області

Метою роботи є аналіз роботи різних методів кластеризації на прикладі бази даних витрат сімей.

Кластерний аналіз — це завдання поділу даної вибірки об’єктів (ситуацій) на підмножини, які називаються кластерами, таким чином, щоб кожен кластер складався з подібних об’єктів, а об’єкти різних кластерів суттєво відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого кола навчальних завдань без вчителя.

Кластеризація — це метод пошуку шаблонів, які призначені для розбиття набору об’єктів на однорідні групи (кластери) або для пошуку наявних структур у даних.

Метою кластеризації є отримання нових знань. Це як «знайти скарб у власному підвалі».

Навіщо це потрібно компаніям? Щоб краще пізнати своїх клієнтів. Знайти індивідуальний підхід до кожного клієнта, а не працювати з усіма однаково Аналіз даних цієї курсової роботи дає змогу більш точно підібрати потрібний алгоритм кластеризації для своїх, або корпоративних проблем.

* 1. Огляд доступних джерел даних

Якщо проаналізувати дані в мережі Інтернет, то ви в першу чергу зустрінете декілька сайтів з інформацією та датасетами на будь-яку тему. Найбільшими та найвідомішими серед них є https://www.kaggle.com/ та https://www.drivendata.org/.

Ці сервіси збирають дані з багатьох сфер людської діяльності та дозволяють користувачу переглянути інформацію з кожного з них. Ці сервіси є досить успішними, але більше репрезентують клієнтську частину ринку задля навчання аналізу даних. Хоча ці сервіси також зберігають дані і можуть їх надавати всім бажаючим, вони не являються освітніми програмами та не навчають користувача користуватись цими даними.

З боку машинного навчання, прекрасними доступними інтернет джерелами є <https://towardsdatascience.com/> та <https://www.globaltechcouncil.org/>

* 1. Постановка задачі

Основним завданням цієї роботи є аналіз роботи різних методів кластеризації на прикладі бази даних витрат сімей.

В цій роботі буде розглянуто декілька алгоритмів кластеризації з практичним використанням на існуючому датасеті прибутків та витрат філіппінських сімей.

Ми проведемо аналіз роботи таких алгоритмів кластеризації:

1. k-means,
2. ієрархічний метод
3. fuzzy c-means

Проведений аналіз порівняння цих методів кластеризації дасть відповідь на питання швидкодії та переваги використання окремих алгоритмів в різних задачах.

Для кожного алгоритма будуть зазначені відповідні математичні обґрунтування, схеми, діаграми, дендрограми, тощо…

1. ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИМИ

В даній роботі ми будемо використовувати мову програмування Python та супутні бібліотеки. Обґрунтовано це тим, що в нинішній час це одна з найбільш популярних мов програмування та тим, що вона проста у застосуванні.

Середовищем програмування буде використаний Jupyter Notebook, а точніше його більш доступна gui - Google Collab. Це середовище дозволяє юзеру виконувати код порядково, один кусок кода за іншим.

Аналізуючи джерела інформації про кластеризацію, зокрема електронне джерело номер 7, були виявлені найбільш використовувані алгоритми кластеризації, чим і обґрунтований вибір цих алгоритмів.

В алгоритмі кластеризації, якщо ймовірність того, що приналежність однієї точки даних до кластеру може приймати тільки значення 1 або 0, це жорстка кластеризація. Границя кластера в методі жорсткої кластеризації може бути відображена як чітка границя. Протилежно, при м’якому методі кластеризації ймовірність того, що приналежності однієї точки даних у кластері можуть приймати будь-яке значення між 0 і 1, наприклад 75%, для якої межі кластера можуть бути представлені як нечітка границя.

Відобразимо графічно різницю між м’якою та твердою кластеризацією (рис. 2.1)

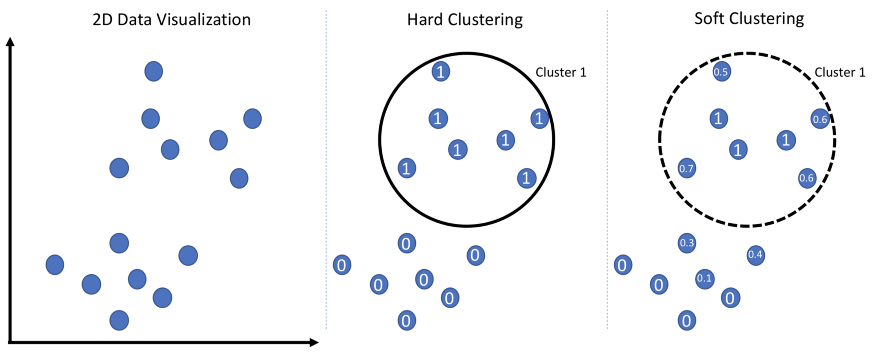
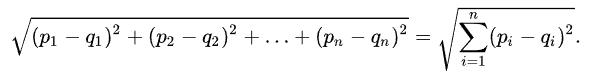


Рисунок 2.1 – Графічне відображення різниці твердої та м’якої кластеризації.

Також для порівняння двох об’єктів необхідно мати критерій, на основі якого буде відбуватися порівняння. Як правило, цим критерієм є відстань між об'єктами.

Існує багато мір відстані, давайте розглянемо деякі з них:

* **Евклідова відстань** — найпоширеніша відстань. Це геометрична відстань у багатовимірному просторі.



* **Квадрат евклідової відстані**. Іноді ви можете звести в квадрат стандартну евклідову відстань, щоб надати більшу вагу більш віддаленим об’єктам.
* **Відстань Чебишова**. Ця відстань може бути корисною, коли потрібно визначити два об’єкти як «різні», якщо вони відрізняються за однією координатою (у будь-якому одному вимірі).



* **Манхетенська відстань** - Ця відстань є просто середнім з різниць координат. У більшості випадків ця міра відстані призводить до тих же результатів, що й для звичайної евклідової відстані. Зауважте, однак, що для цієї міри вплив окремих великих відмінностей (похибок) зменшується (оскільки вони не в квадраті).



Якщо вибрати різні відстані, результати кластеризації можуть значно відрізнятися, тому в нашій роботі ми будемо використовувати евклідову відстань. Це зумовлено тим, що вона є найпоширенішою, так як є основною в евклідовій геометрії.

1. ЗАСТОСУВАННЯ ТА ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИМИ ДЛЯ ПОСТАВЛЕНОГО ЗАВДАННЯ
   1. Визначення моделей та методів, що можуть бути використані

У аналізі даних ми можемо використовувати кластерний аналіз, щоб отримати цінну інформацію з наших даних, побачивши, в які групи потрапляють точки даних, коли ми застосовуємо алгоритм кластеризації.

Існують різні типи алгоритмів кластеризації:

* **На основі щільності**. У кластеризації на основі щільності дані групуються за областями високої концентрації точок даних, оточеними областями з низькою концентрацією точок даних. В основному алгоритм знаходить місця, які щільно заповнені точками даних, і викликає ці кластери.

Чудова особливість цього полягає в тому, що кластери можуть бути будь-якої форми. Ви не обмежені очікуваними умовами.

Алгоритми кластеризації під цим типом не намагаються призначити викиди для кластерів, тому їх ігнорують.

* **На основі розподілу.** При підході до кластеризації на основі розподілу всі точки даних вважаються частинами кластера на основі ймовірності того, що вони належать даному кластеру.

Це працює так: є центральна точка, і зі збільшенням відстані точки даних від центру ймовірність того, що вона є частиною цього кластера, зменшується.

Якщо ми не впевнені, яким може бути розподіл у ваших даних, вам слід розглянути інший тип алгоритму.

* **На основі центроїди.** Напевно, найчастіше використовується кластеризація на основі центроїд. Він трохи чутливіший до початкових параметрів, які ви йому надаєте, але він швидкий та ефективний. Ці типи алгоритмів розділяють точки даних на основі кількох центроїдів у даних. Кожна точка даних призначається кластеру на основі його квадратної відстані від центроїда. Це найпоширеніший тип кластеризації.
* **На основі ієрархії.** Кластеризація, заснована на ієрархії, зазвичай використовується для ієрархічних даних, як, наприклад, з бази даних компанії або таксономій. Він створює дерево кластерів, щоб все було організовано зверху вниз. Це більш обмежувальне рішення, ніж інші типи кластеризації, але воно ідеально підходить для певних типів наборів даних.

Як ви могли зрозуміти, існує багато типів кластеризації, у цій роботі ми розглянемо найпопулярніші з них за інформацією електронного ресурса номер 7:

1. K-means
2. Hierarchical algorithm
3. Fuzzy c-means
   1. Вибір ознак, що будуть використані для аналізу.

Сам датасет містить більше 60-ти ознак. В цілому вони описують типи витрат та надходжень у сімей. Є також категоріальні значення, які нам не знадобляться при аналізі та для кластеризації.

Спершу дізнаємося чи є в датасеті нульові поля за допомогою функції isnull() (рис 3.1).

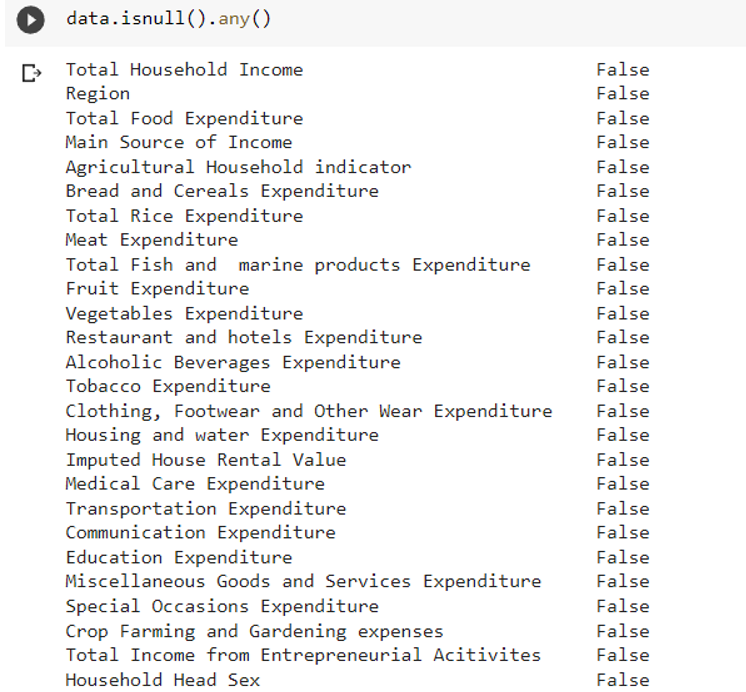


Рисунок 3.1 – Результат роботи функції isnull().

Бачимо що більшість полів не мають нульових значень.

Далі треба зрозуміти межі доступних даних. Для цього використаємо метод .describe() (рис 3.2)

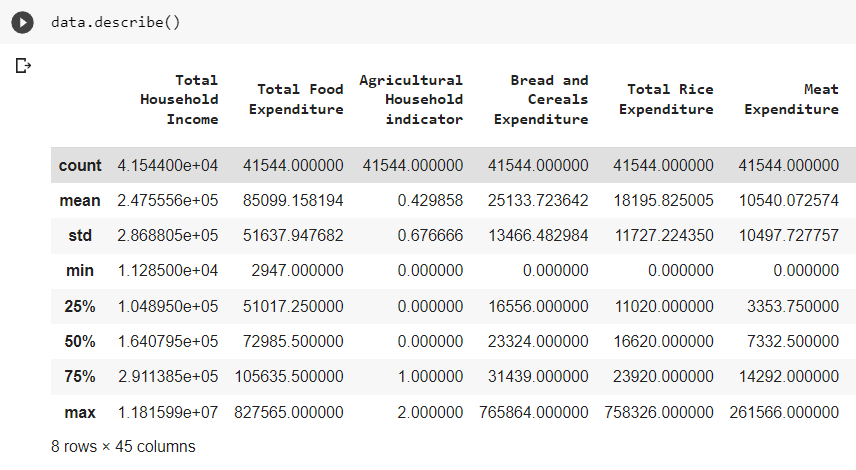


Рисунок 3.2 – Результат роботи функції .describe().

Як ми можемо бачити, перші два показники “Total Household Income” та “Total Food Expenditure” мають цілком сприятливі показники. Також хотілось би зауважити що обчислення ведуться в Філіппінських пєсо (1грн = 0.53 Філіппінських пєсо).

Приступимо до візуалізації даних. Скористаємося бібліотекою matplotlib. Спочатку візуалізуємо колонку прибутків за допомогою функції .hist() (рис. 3.3)

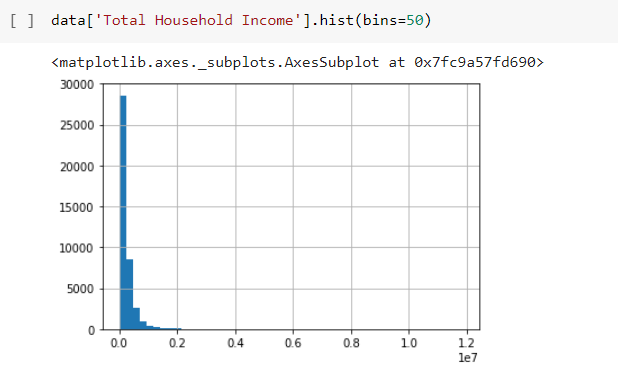


Рисунок 3.3 – Результат роботи функції .hist()

По осі х – прибуток домогосподарства.

По осі у – кількість таких домогосподарств

Виглядає так, ніби щось пішло не так, але насправді ця візуалізація цілком правдива. Відсортуємо дані у зворотному порядку за допомогою функції .sort(reverse= True) (рис.3.4).

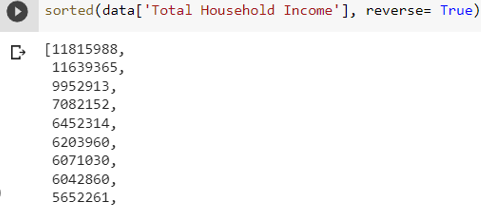


Рисунок 3.4 – Результат роботи функції .sort(reverse= True)

Як ми можемо бачити, в цьому датасеті існують як малозабезпечені, так і дуже багаті сім’ї з річними надходженнями більше 11 млн. п’єсо.

Для того щоби краще орієнтуватись в даних менш забезпечених сімей, подивимось глибше до показників х<500000 (рис. 3.5).

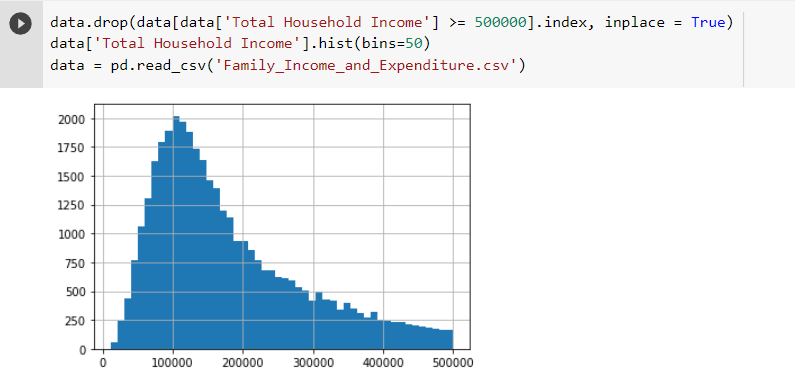


Рисунок 3.5 – Гістограма прибутків менш забезпечених сімей.

З даного графіка ми можемо зробити висновок що більшість сімей заробляють біля 100,000 Філіппінських п’єсо. Розподіл більш схожий на нормальний, але таким не є.

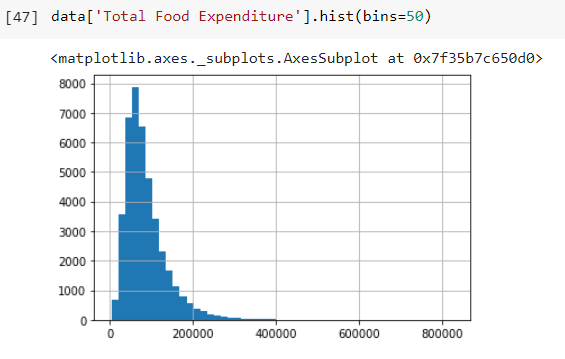
Наступною ознакою провізуалізуємо витрати сімей на їжу (рис. 3.6) 

Рисунок 3.6 – Гістограма витрат сімей на їжу.

Логічно, що сім’ї які заробляють більше – тратять на їжу більше грошей, тому звузимо нашу гістограму до х<250000 для того щоб подивитися глибше на витрати малозабезпечених сімей (рис.3.7).

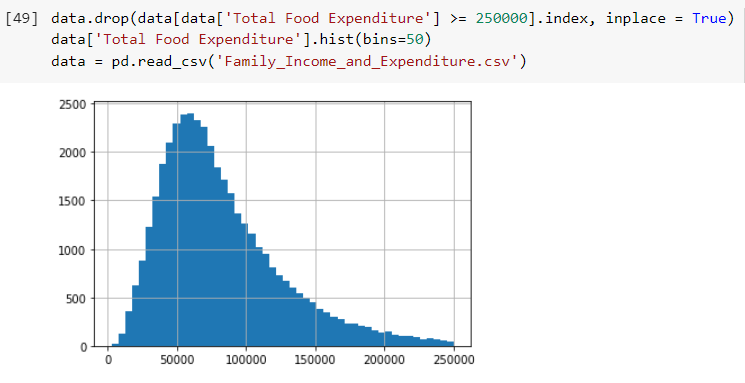


Рисунок 3.7 – Гістограма витрат менш забезпечених сімей на їжу.

З цього графіка ми можемо зробити висновок що більшість сімей витрачають на їжу біля 50000 Філіппінських п’єсо.

Для кластеризації з більш ніж двома параметрами підберемо ще один стовпець. Нехай це будуть витрати на м’ясо. Провізуалізуємо стовпець у вигляді гістограми (рис. 3.8).

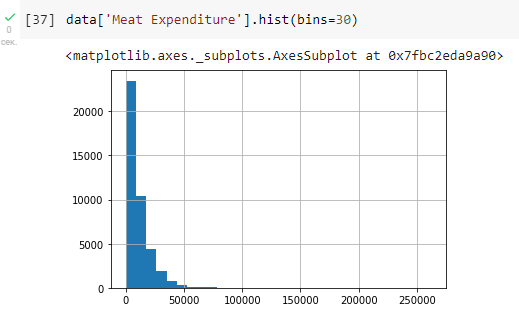


Рисунок 3.8 – Гістограма витрат сімей на м’ясо.

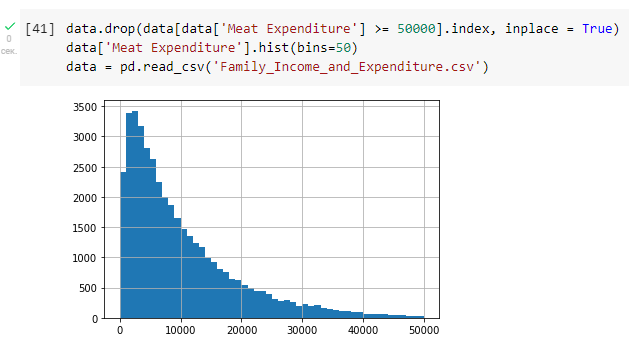
Ситуація така ж як і в минулих гістограмах – бідних сімей набагато більше ніж багатих, тому звузимо графік до значень >50000 (рис. 3.9). 

Рисунок 3.9 – Гістограма витрат на м’ясо > 50000.

* 1. K-means

Розглянемо перший алгоритм кластеризації – метод К-середніх.

Як же працює цей алгоритм? Розберемо покроково:

Крок 1: По-перше, нам потрібно вказати кількість кластерів K, які необхідно створити за допомогою цього алгоритму.

Крок 2: Коротко класифікуємо дані на основі кількості точок даних.

Крок 3: Тепер обчислюємо центроїди кластерів. (рис. 3.10)

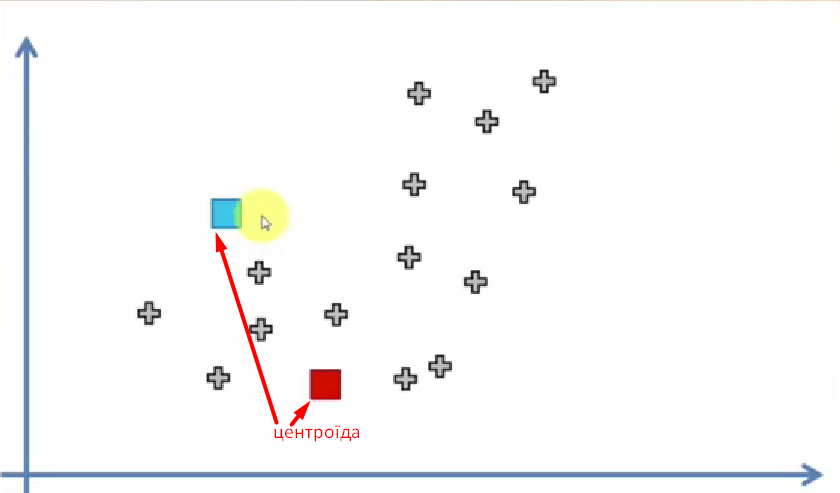


Рисунок 3.10 – Приклад для пояснення кроків 1-3.

Крок 4: Повторюємо наведені нижче кроки, поки не знайдемо ідеальну центроїду, яка є призначенням точок даних кластерам, які не змінюються:

- Спочатку обчислюємо суму квадратів відстаней між точками даних і центроїдами. В нашому випадку це буде евклідова відстань між точками.

- На цьому етапі нам потрібно виділити кожну точку даних кластеру, яка є найближчою до інших (центроїдою).

- Нарешті, обчислюємо центроїди для кластерів, усереднюючи всі точки даних кластера. (рис 3.11 – 3.12)

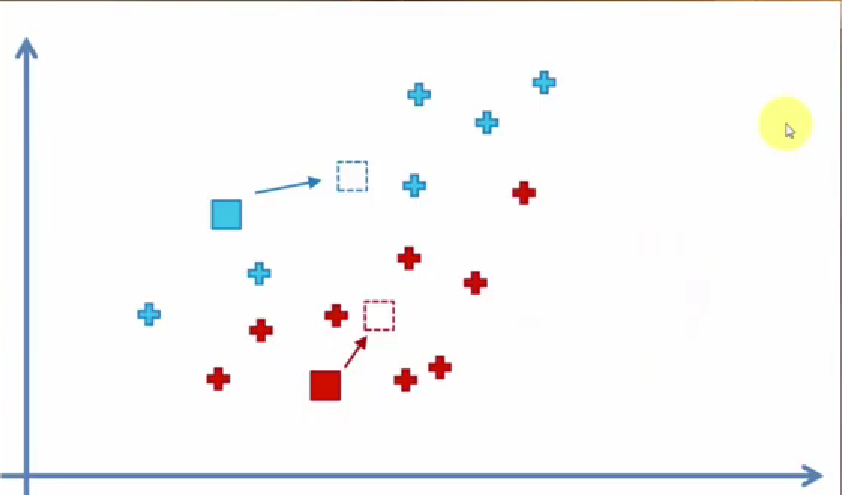


Рисунок 3.11 – Графічний приклад роботи алгоритму К-середніх.

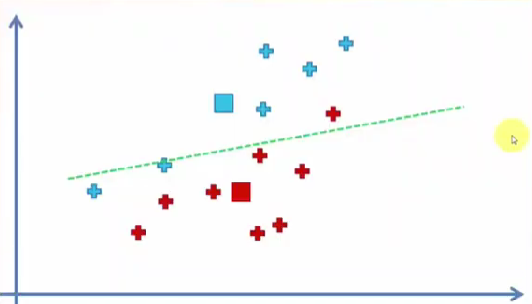


Рисунок 3.12 – Графічний приклад роботи алгоритму К-середніх

K-means реалізує стратегію Expectation-Maximization для вирішення проблеми. Крок очікування використовується для призначення точок даних найближчому кластеру, а крок максимізації використовується для обчислення центроїди кожного кластера.

Перейдемо до реалізації моделей на нашому датасеті.

По-перше, для цього алгоритму треба визначитися з кількістю кластерів. Зробимо це за методом ліктя.

Для проблем кластеризації існує метрика WCSS (Within clusters some of squares) або ж сума квадратів всередині кластерів.

Вираховується вона по такій формулі:

Тож, наш алгоритм для кожного кластера вираховує суму квадратів відстаней точок (які належать цьому кластеру) до його центроїди.

Роздивимось більш наглядно. Якщо ми взяли би один кластер, і порахували би відстань від центроїди цього кластеру до всіх точок, то ця метрика була би дуже великою (рис. 3.13).

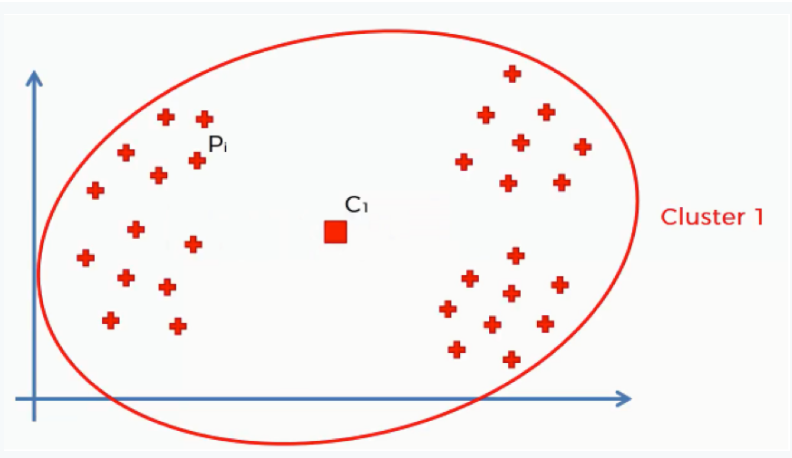


Рисунок 3.13 – Візуалізація моделі з одним кластером.

Але якщо ми візьмемо більше число кластерів, тоді відстань до центроїд буде більш оптимальною, відповідно метрика буде менша (рис 3.14).

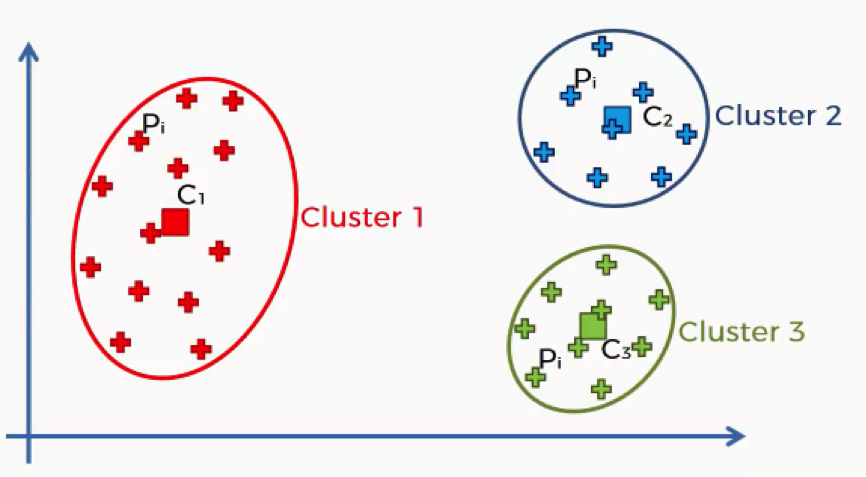


Рисунок 3.14 – Візуалізація моделі з трьома кластерами.

Для визначення оптимальної кількості кластерів використовують метод ліктя.

У кластерному аналізі метод ліктя є евристичним методом, який використовується для визначення кількості кластерів у наборі даних. Метод складається з побудови графіка пояснення варіації як функції від кількості кластерів і вибору ліктя кривої як кількості кластерів для використання.

Використаємо метод ліктя на нашому датасеті. Для цього використаємо бібліотеку sklearn. (рис. 3.15)

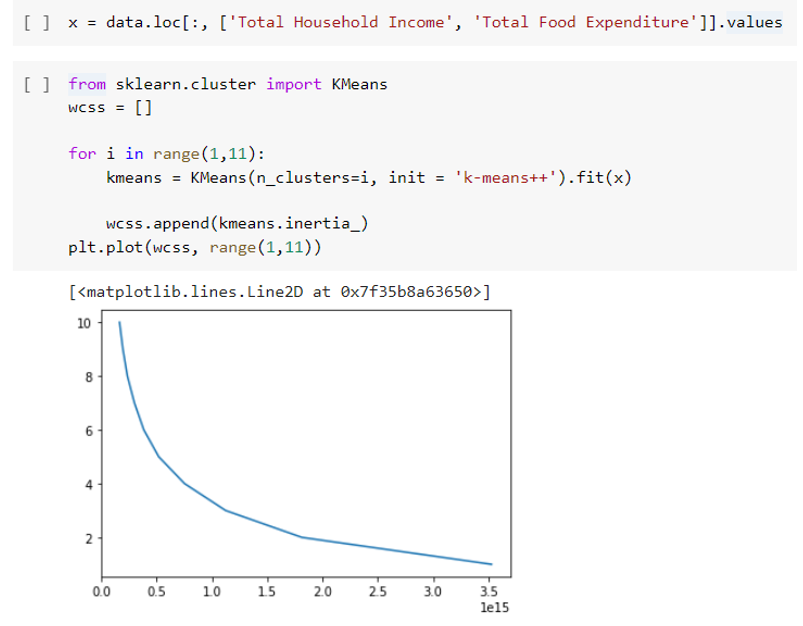


Рисунок 3.15 – Застосування та візуалізація методу Ліктя

З графіка бачимо що оптимальна кількість кластерів це 4 або 5, так як саме на цих позначках графік має прийнятні показники.

Перейдемо до тренування моделі (рис. 3.16).

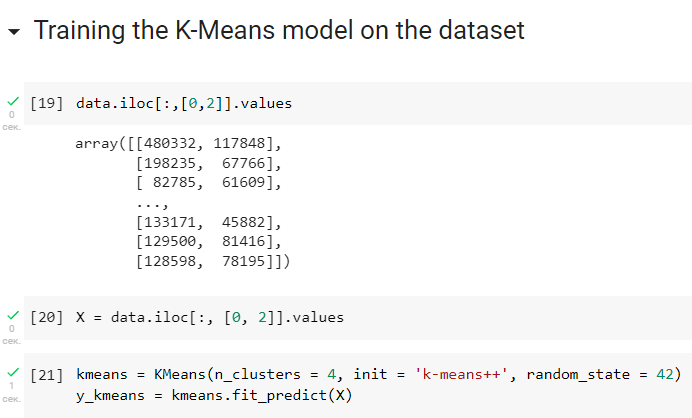


Рисунок 3.16 – Тренування моделі.

Далі зробимо візуалізацію отриманих даних. (рис. 3.17 – 3.18).

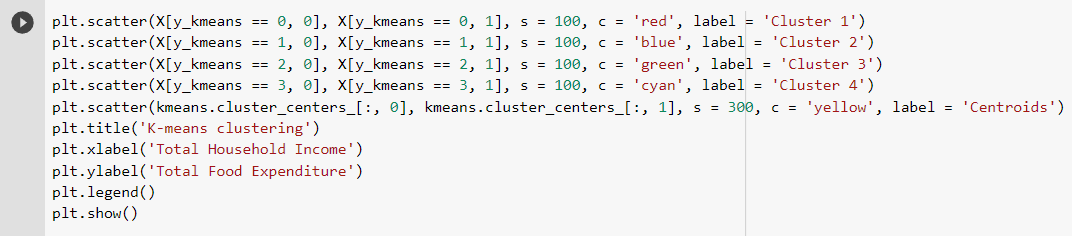


Рис. 3.17 – Код для візуалізації кластеризації.

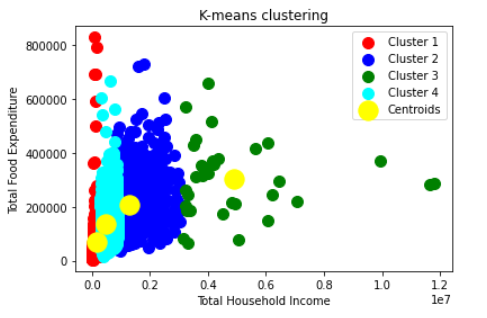


Рисунок 3.18 – Візуалізація результату кластеризації алгоритмом К-середніх.

* 1. Hierarchical clustering

Ієрархічна кластеризація — це алгоритм, який групує подібні об’єкти в групи, які називаються кластерами. Кінцева точка - це набір кластерів, де кожен кластер відрізняється один від одного, а об'єкти в кожному кластері в цілому схожі один на одного.

Ієрархічна кластеризація може бути виконана або з матрицею відстаней, або з необробленими даними. Коли надаються вихідні дані, програмне забезпечення автоматично обчислить матрицю відстані у фоновому режимі.

Ієрархічні алгоритми діляться на агломеративні і дівізімні.

Агломеративні алгоритми - це алгоритми, які починають своє виконання з того, що кожен об'єкт заносять в свій власний кластер і в міру виконання об'єднують кластери, до тих пір, поки в кінці не отримує один кластер, що включає в себе всі об'єкти набору.

Дивізімні алгоритми, навпаки, спочатку відносять всі об'єкти в один кластер і потім розділяють цей кластер до тих пір, поки кожен об'єкт не виявиться в своєму власному кластері (рис. 4.19).

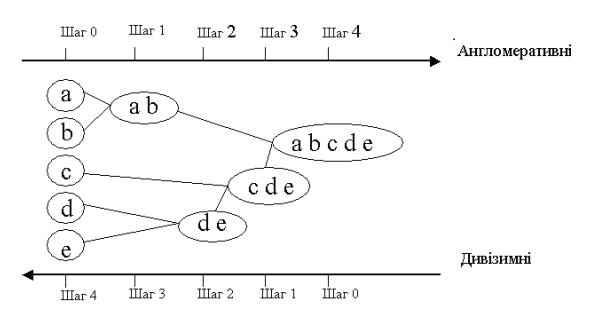


Рисунок 3.19 – Різниця між англомеративними та дивізимними алгоритмами

Тож яким чином працює ієрархічна кластеризація?

Ієрархічна кластеризація починається з обробки кожного спостереження як окремого кластера. Потім він багаторазово виконує наступні два кроки:

1) визначає два кластери, які знаходяться найближче один до одного

2) об’єднує два найбільш схожі кластери.

Цей ітераційний процес триває, поки всі кластери не будуть об’єднані разом. Це показано на діаграмах нижче (рис. 3.20).

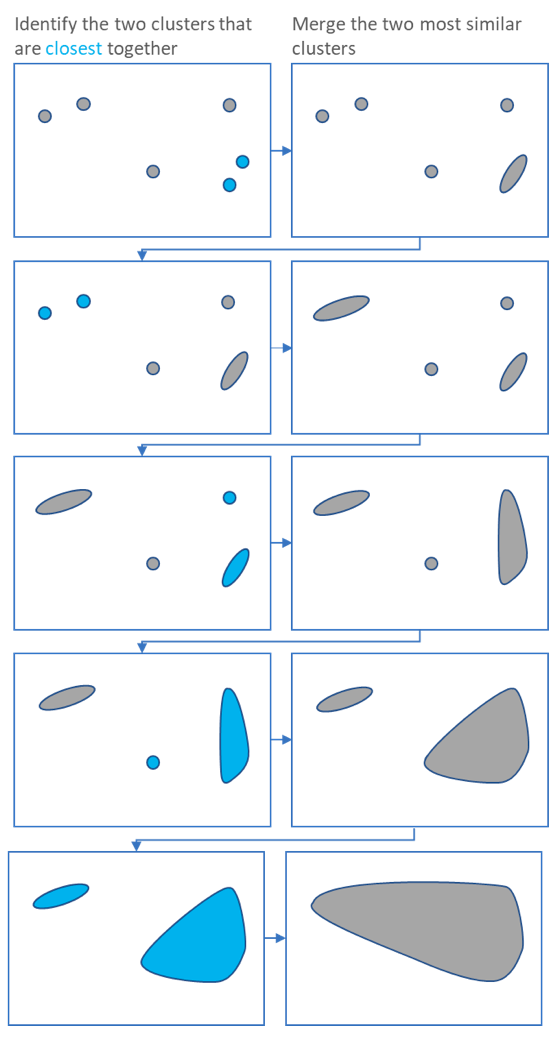


Рисунок 3.20 – Процес роботи Ієрархічного Алгоритму.

Основним результатом ієрархічної кластеризації є дендрограма, яка показує ієрархічні відносини між кластерами. Також за допомогою дендрограми можно визначити оптимальну кількість кластерів. Використаємо це на нашому датасеті (рис. 3.21).

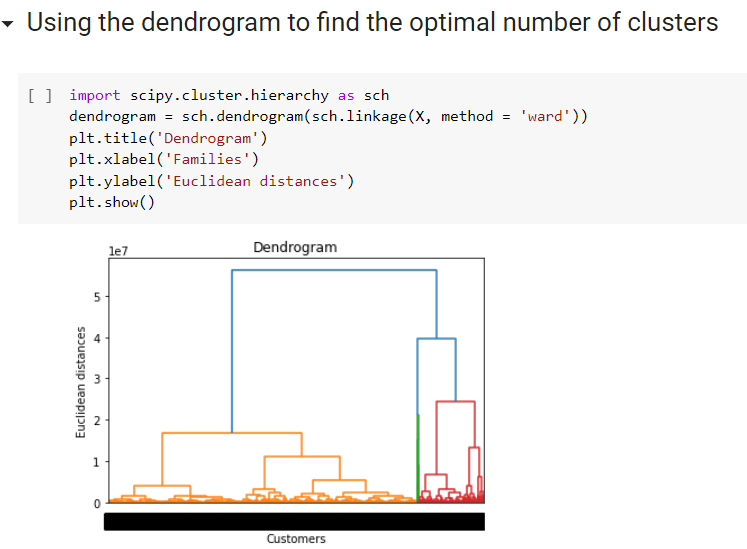


Рисунок 3.21 - Код для візуалізації дендрограми.

З даної дендрограми можемо зробити висновок, що оптимальна кількість кластерів це 4.

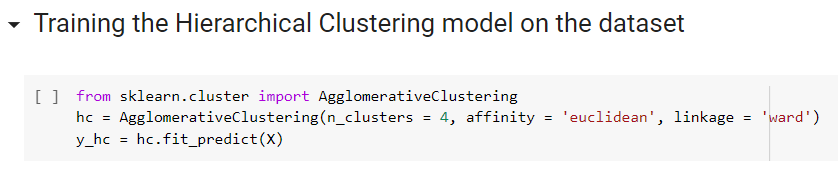
Перейдемо до тренування ієрархічного алгоритму на нашому датасеті. У якості метрики використаємо евклідову відстань. (рис 3.22). 

Рисунок 3.22 – Код для тренування Ієрархічного алгоритму.

Провізуалізуємо результат (рис. 3.23).

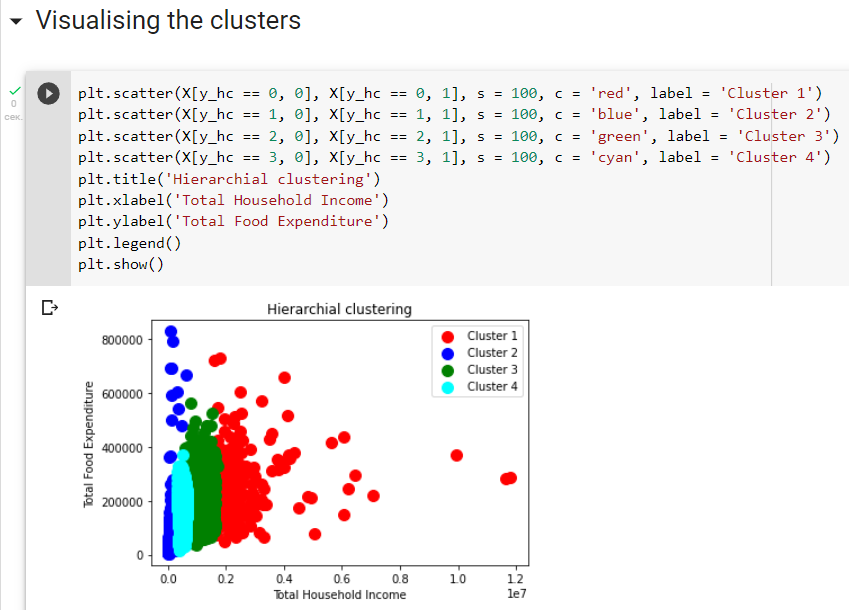


Рисунок 3.23 – Код для візуалізації результатів Ієрархічної кластеризації.

* 1. Fuzzy C-means

Алгоритм нечітких с-середніх це алгоритм м’якої кластеризації.

У методі кластеризації нечітких c-середніх (FCM) ми маємо два параметри, μ\_ij і c\_i, і один гіперпараметр m.

μ\_ij, значення приналежності, — це ймовірність того, що j-та точка даних належить i-му кластеру, і вона обмежена тим, що сума μ\_ij над центрами кластера C дорівнює 1 для кожної точки даних j.

c\_i — центр i-го кластера (такий самий вимір із X).

І гіперпараметр m контролює, наскільки нечіткою має бути межа кластера.

Роздивимося на прикладі (рис. 3.24).

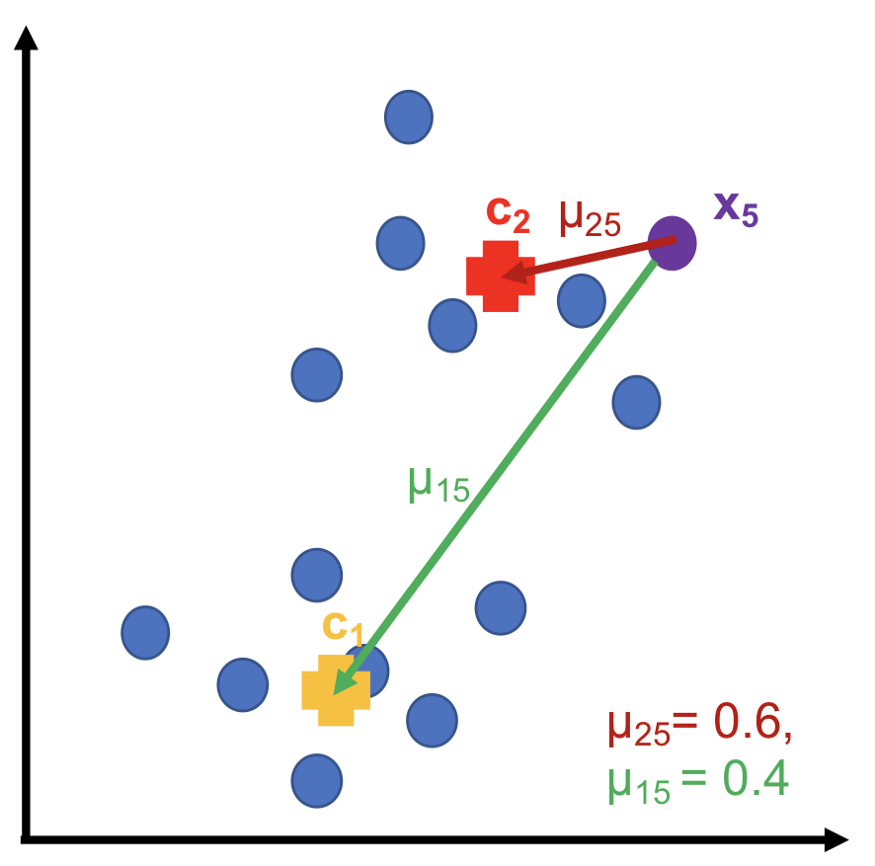
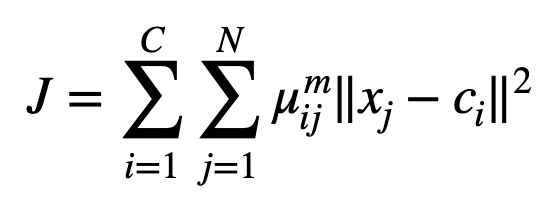


Рисунок 3.24 – Графічний приклад роботи алгоритму нечітких С-середніх.

У наведеному вище графіку ми дивимося на 5-ту точку даних X\_5 і припустимо, що ми знаємо, що є лише два кластери, а поточні центри кластерів — c\_1 і c\_2. μ\_25 – ймовірність того, що 5-а точка даних належить до 2-го кластеру, а μ\_15 – ймовірність того, що 5-а точка даних належить 1-му кластеру. Тоді ми бачимо, що 5-а точка даних набагато ближча до c\_2, ніж c\_1, тому μ\_25 (0,6) більше, ніж μ\_15 (0,4). І вони відповідають обмеженню, що сума μ для кожної точки даних дорівнює 1, де μ\_15 + μ\_25 = 1.

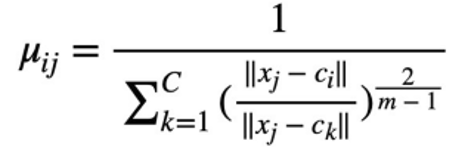
Об’єктивна функція цього алгоритму:



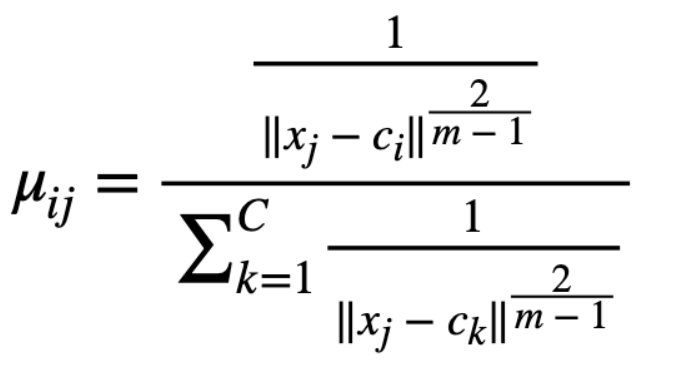
Формально алгоритм FCM можна описати так:

1. Випадково розкидаємо центри кластерів.
2. Повторюємо ці два пункти поки зміна U між двома ітераціями не стала дуже мала (заздалегідь визначене відсічення).

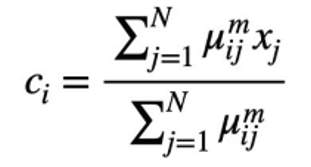
2.1) Обчислюємо та оновлюємо нечітку належність  для кожних даних у кластері за формулою:



Або ж:



2.2) Обчислюємо та оновлюємо центроїду  для кожного кластеру за формулою:



Перейдемо до реалізації алгоритму нечітких С-середніх на нашому датасеті (рис. 3.25 – 3.26).

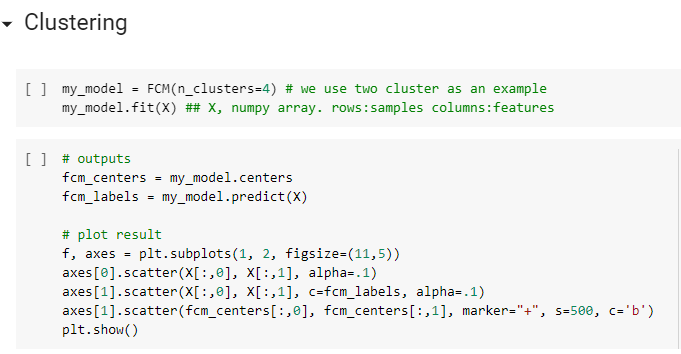


Рисунок 3.25 – Код для тренування та візуалізації алгоритму нечітких С-середніх.

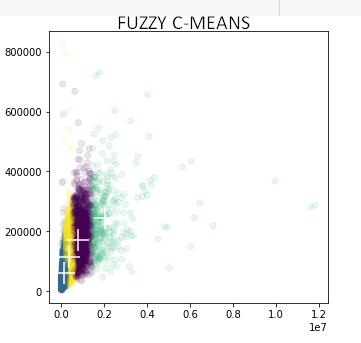


Рисунок 3.26 – Візуалізація результату роботи алгоритму нечітких С-середніх.

* 1. Кластеризація з більш ніж двома параметрами.

Іноді в задачі кластеризації може виникнути необхідність враховувати більш ніж 2 параметра. В такому випадку доцільно використовувати алгоритм К-середніх.

По-перше, так як ми записуємо в датафрейм три показники замість двух, треба реалізувати метод ліктя для нових даних (рис. 3.27)

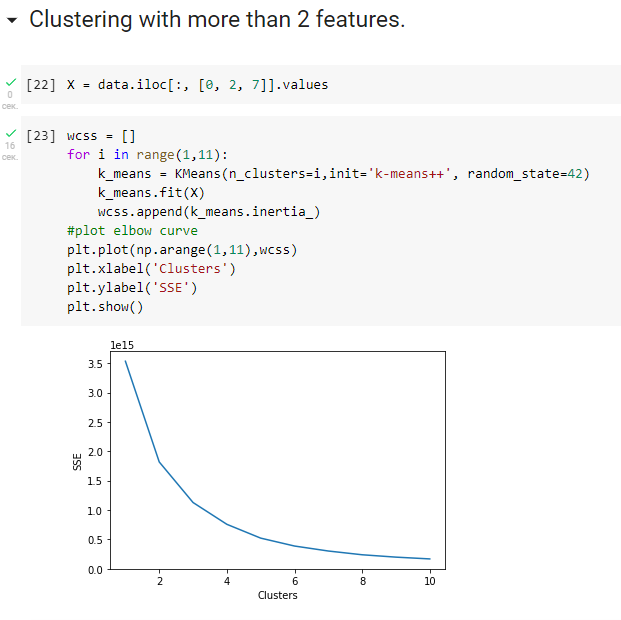


Рисунок 3.27 – Метод ліктя для нового датафрейму.

Тепер із кривої видно, що оптимальна кількість кластерів, тобто n\_clusters, дорівнює 4. Далі застосуємо оптимальну кластеризацію k-середніх, щоб знайти номер кластера для кожної ноди даних (рис. 3.28).

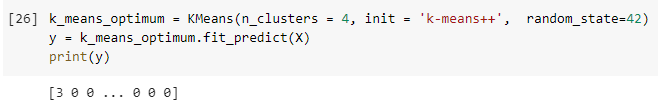


Рисунок 3.28 – Застосування кластеризації К-середніх.

Тепер ми можемо додати додатковий стовпець у кінці під назвою «кластер», щоб вказати, до якого кластера належать дані. Як зазвичай, нумерація кластерів починається з 0 (рис. 3.29).



Рисунок 3.29 – Код для додавання нового стовпця.

Тепер ми можемо розділити таблицю даних на чотири частини. Одна містить дані з кластером 0 (перший кластер), а інша містить дані з кластером 1 (другий кластер) ітд… Назвемо дані1, дані2, ітд… відповідно (рис.4.30) .

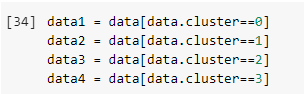


Рисунок 4.30 – Код для розділення таблиць на частини.

На цьому етапі ми маємо необхідні дані, розділені на кластери. Тепер замість 2D-графіка ми використовуємо 3D-графік, оскільки тут працюємо з 3 параметрами. З кожним новим параметром, потрібно збільшувати кількість вимірів на 1. Призначаємо кожному кластеру окремий колір та візуалізуємо графік (рис. 3.31 - 3.32)

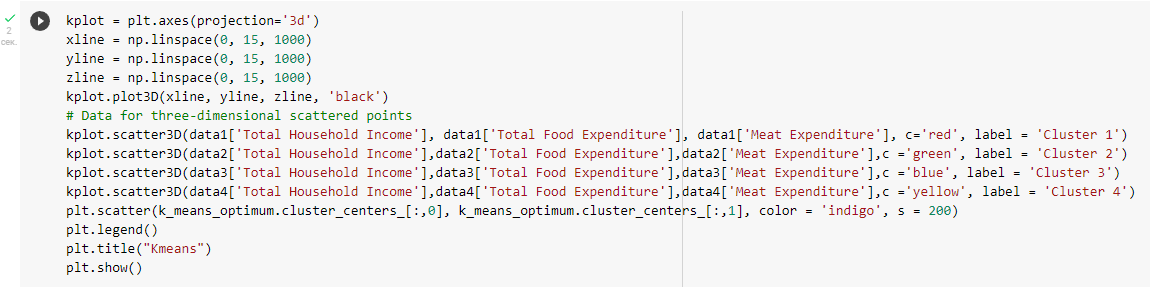


Рисунок 3.31 – Код для побудови графіка кластеризації.

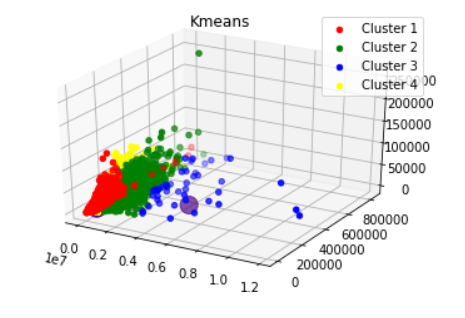


Рисунок 3.32 – Візуалізація кластеризації з 3 параметрами.

Але які методи візуалізації використовувати коли параметрів більше ніж 3? Розглянемо декілька з них. Нашим першим методом візуалізації буде аналіз основних компонентів (PCA).

PCA — це алгоритм, який використовується для зменшення розмірності, тобто неофіційно, що він може приймати датафрейм з багатьма стовпцями і повертати датафрейм зі зменшеною кількістю стовпців, який все ще зберігає велику частину інформації зі стовпців вихідного датафрейму. Стовпці датафрейму, отримані за допомогою процедури PCA, називаються основними компонентами. Ми будемо використовувати ці основні компоненти, щоб допомогти нам візуалізувати наші кластери в 1-D, 2-D і 3-D просторах, оскільки ми не можемо легко візуалізувати дані, які ми маємо, у більш високих вимірах. Наприклад, ми можемо використовувати два головні компоненти для візуалізації кластерів у 2-D просторі або три головні компоненти для візуалізації кластерів у 3-D просторі. Але спочатку ми імпортуємо бібліотеки, визначимо основний датафрейм та створимо окремий, менший датафрейм - plotX, щоб побудувати наші дані (рис. 3.33). Причина, чому ми створюємо менший датафрейм, полягає в тому, щоб ми могли швидше наносити наші дані на графік і щоб наші графіки не виглядали занадто брудними або переповненими.

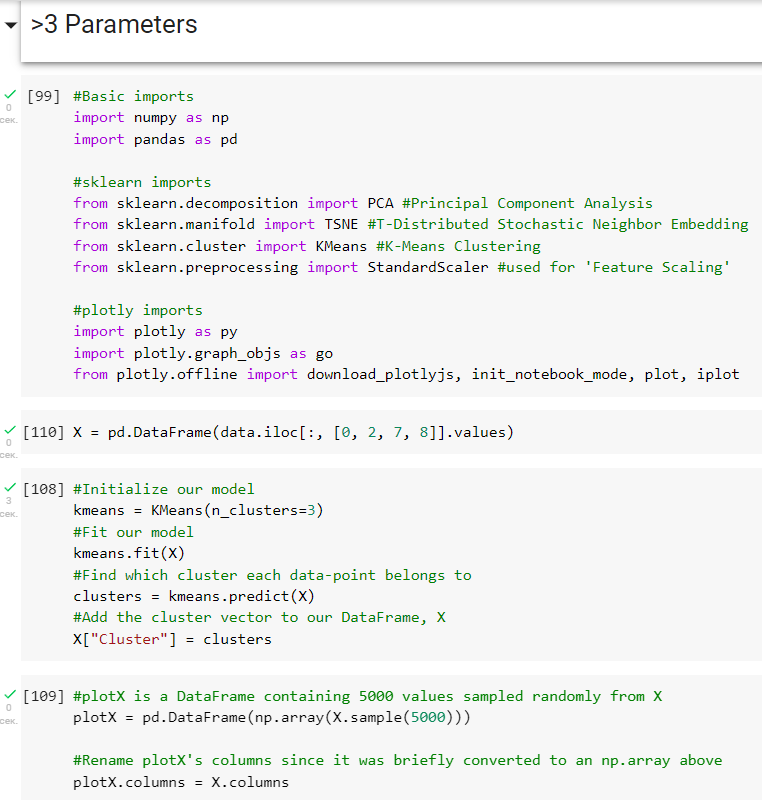


Рисунок 3.33 – Код

Тепер, щоб візуалізувати наші дані, ми побудуємо три датафрейму з plotX за допомогою алгоритму «PCA».

Перший датафрейм буде містити результати алгоритму PCA лише з одним головним компонентом. Цей датафрейм буде використовуватися для візуалізації наших кластерів в одному вимірі (1-D).

Другий датафрейм буде містити два основних компоненти, повернуті алгоритмом PCA з n\_components=2. Цей датафрейм допоможе нам у візуалізації цих кластерів у двох вимірах (2-D).

А третій датафрейм буде містити результати алгоритму PCA, який повертає три основні компоненти. Цей датафрейм дозволить нам візуалізувати кластери в тривимірному просторі (3-D). Ініціалізуємо наші PCA моделі (рис. 3.34) Зауважимо, що вище ми виконували наші PCA для даних, які виключали змінну Cluster.

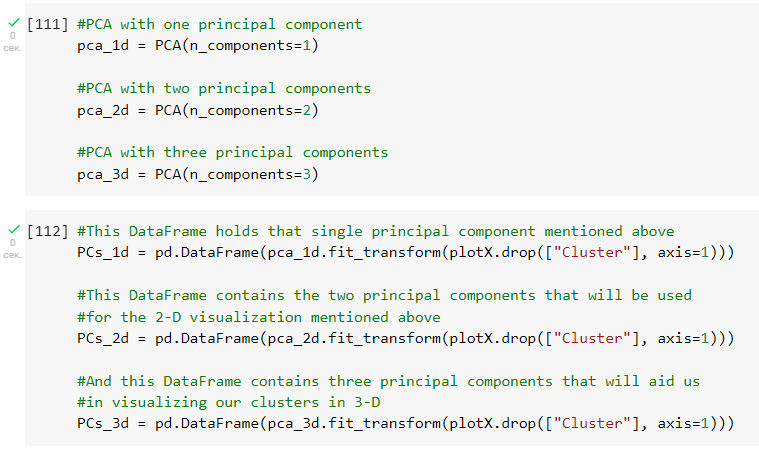


Рисунок 3.34 – Код ініціалізації PCA моделей.

Перейменуемо стовпці цих новостворених датафреймів та об’єднаємо ці нещодавно створені датафрейми до plotX, щоб plotX їх міг використовувати як стовпці. Також створюємо один новий стовпець для plotX, щоб ми могли використовувати його для одновимірної візуалізації. (рис 3.35).

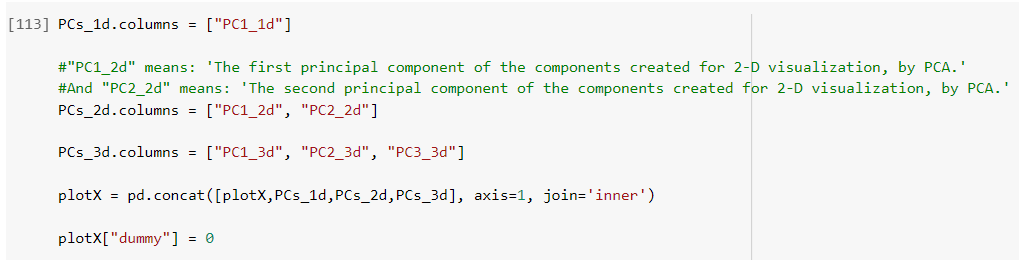


Рисунок 3.35 – Код.

Тепер ми розділимо наш датафрейм, plotX, на три нових датафрейму. Кожен із цих нових фреймів даних буде містити всі значення, що містяться в одному з кластерів. Тобто, усі значення, що містяться в cluster0, належатимуть до «кластеру 0», а всі значення, що містяться в cluster1, належатимуть «кластеру 1» тощо (рис 3.36).

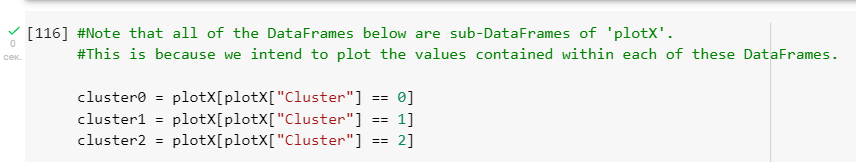


Рисунок 3.36 – Код.

На графіку нижче показано наші три оригінальні кластери на єдиному головному компоненті, створеному для одновимірної візуалізації (рис. 3.37)

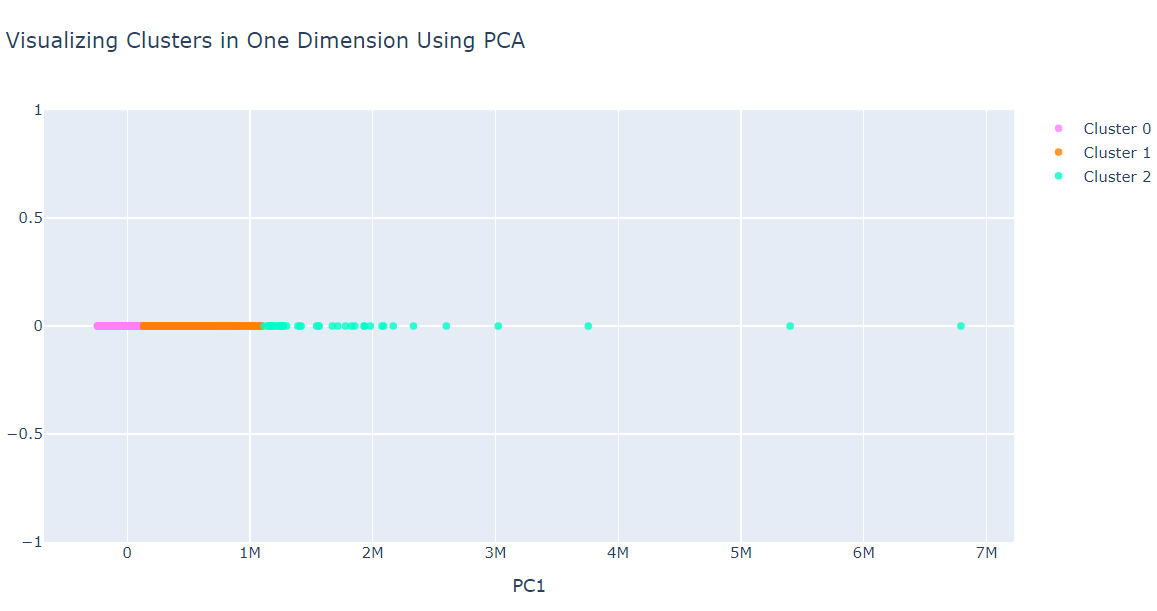


Рисунок 3.37

Наступний графік відображає три кластери на двох основних компонентах, створених для двовимірної візуалізації (рис. 3.38)

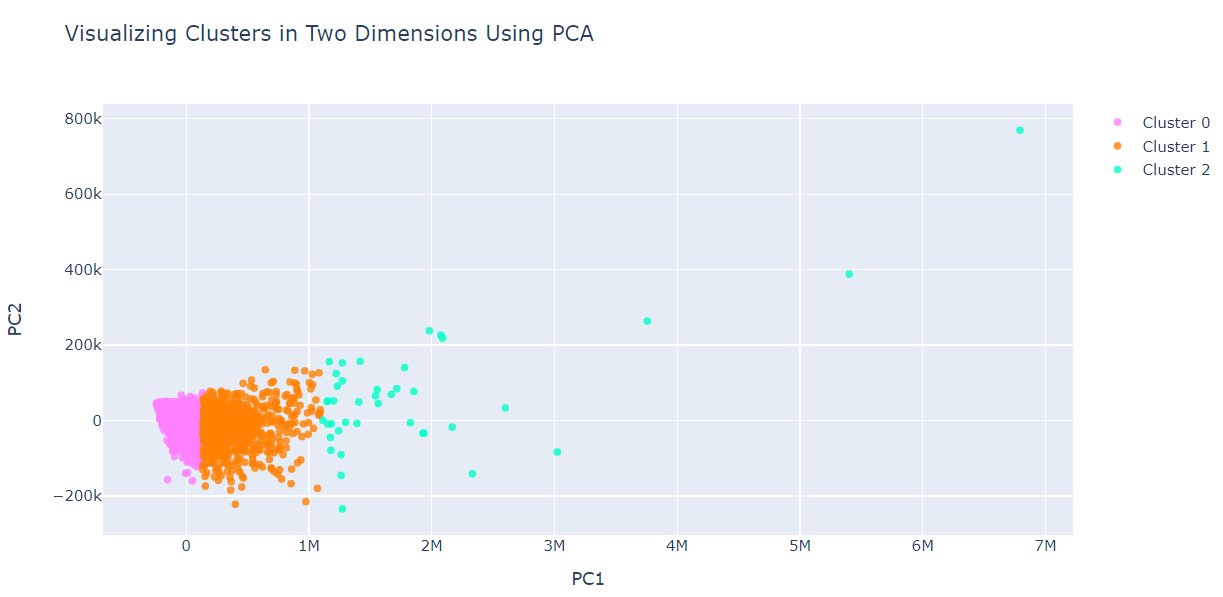


Рисунок 3.38

Цей останній графік нижче відображає наші кластери на трьох основних компонентах, створених для 3-D візуалізації (рис. 3.39)

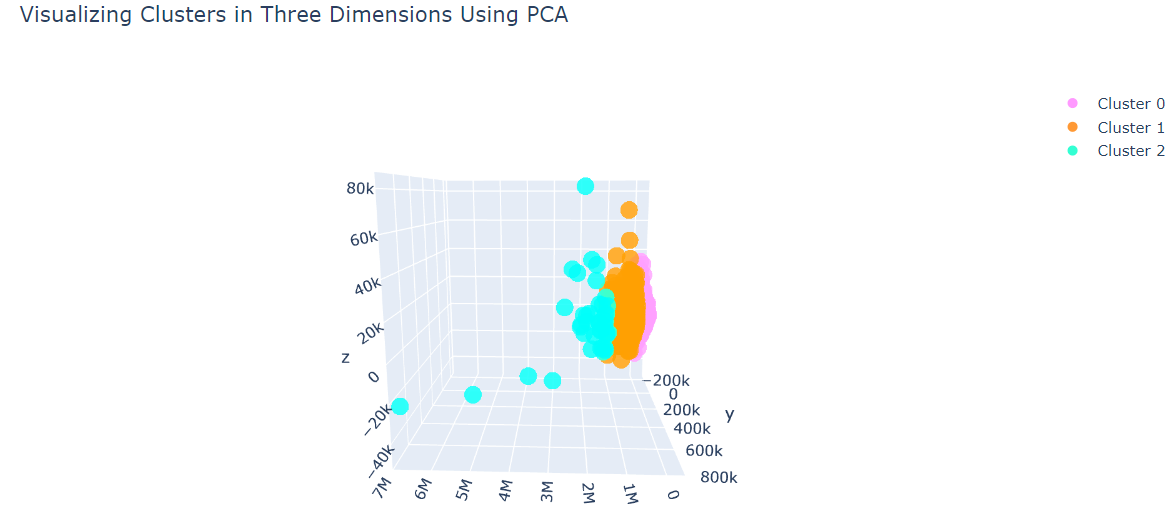


Рисунок 3.39

Як ми бачимо з наведених вище графіків: якщо у вас є дані, які дуже кластеризуються, то PCA — це досить хороший спосіб переглянути кластери, сформовані на вихідних даних. Крім того, здавалося б, що візуалізація кластерів ефективніша, коли кластери візуалізуються з використанням більшої кількості основних компонентів, а не меншої. Наприклад, 2-вимірний графік краще забезпечував чітке візуальне представлення кластерів, ніж 1-вимірний графік.

Наступним методом візуалізації наших кластерів є T-Distributed Stochastic Neighbor Embedding (T-SNE).

T-SNE став дуже популярним методом візуалізації даних великої розмірності. Хоча t-SNE є технікою зменшення розмірності, вона в основному використовується для візуалізації, а не для попередньої обробки даних (як PCA). Коротше кажучи, T-SNE — це цікавий і складний алгоритм машинного навчання, який може допомогти нам візуалізувати дані великої розмірності. Це метод для виконання зменшення розмірності, і саме з цієї причини ми можемо використовувати його, щоб допомогти нам візуалізувати наші три кластери, які були побудовані на даних високої розмірності.

Ще раз ми створюємо суб-датафрейм під назвою plotX, який буде містити вибірку даних із X для візуалізації. Далі ми повинні вирішити, який рівень заплутаності ми хотіли б використовувати для нашого алгоритму T-SNE. Заплутаність — це гіперпараметр, що використовується в алгоритмі T-SNE, який значною мірою визначає, як розподіляються дані, повернуті з алгоритму. Нехай заплутаність буде дорівнювати 50, але ймовірно, існує більш ідеальне значення. Тож, ми ініціалізуємо наші моделі T-SNE та створюємо наші нові датафрейми, щоб допомогти нам візуалізувати наші дані в 1-D, 2-D і 3-D просторі (рис. 3.40)



Рисунок 3.40 – Код.

Тепер перейменуймо стовпці цих новостворених фреймів даних, об’єднаємо ці нещодавно створені датафрейми до plotX, щоб він міг їх використовувати як стовпці. Також створюємо один новий стовпець для plotX, щоб ми могли використовувати його для одновимірної візуалізації та розділимо plotX, на три нові датафрейми. Кожен із цих нових фреймів даних буде містити всі значення, що містяться в одному з кластерів. Наприклад, усі значення, що містяться в датафреймі, cluster0, належатимуть до «кластеру 0», а всі значення, що містяться в датафреймі, cluster1, належатимуть «кластеру 1» тощо (рис. 3.41).

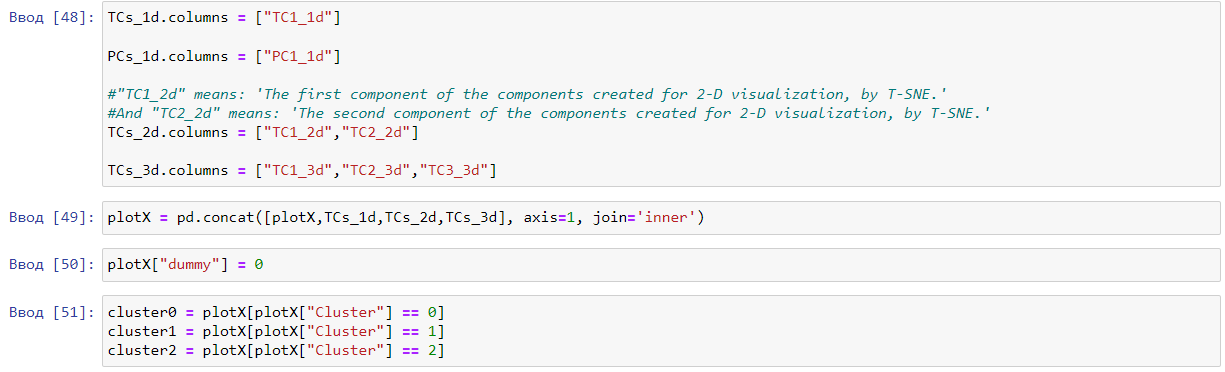


Рисунок 3.41 – Код.

На графіку нижче показано наші три оригінальні кластери в єдиному вимірі, створеному T-SNE для одновимірної візуалізації (рис. 3.42)

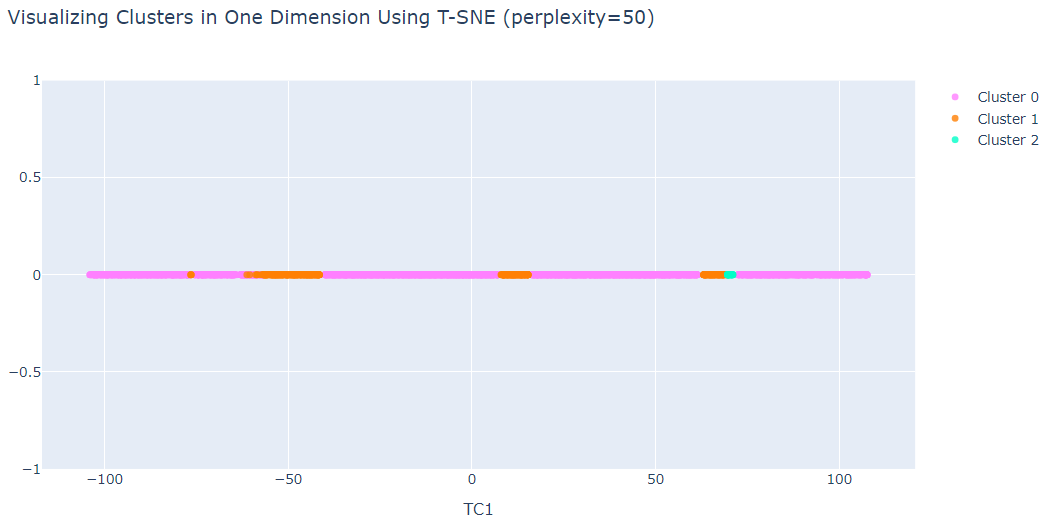


Рисунок 3.42

Наступний графік відображає три кластери у двох вимірах, створених T-SNE для двовимірної візуалізації (рис. 3.43).

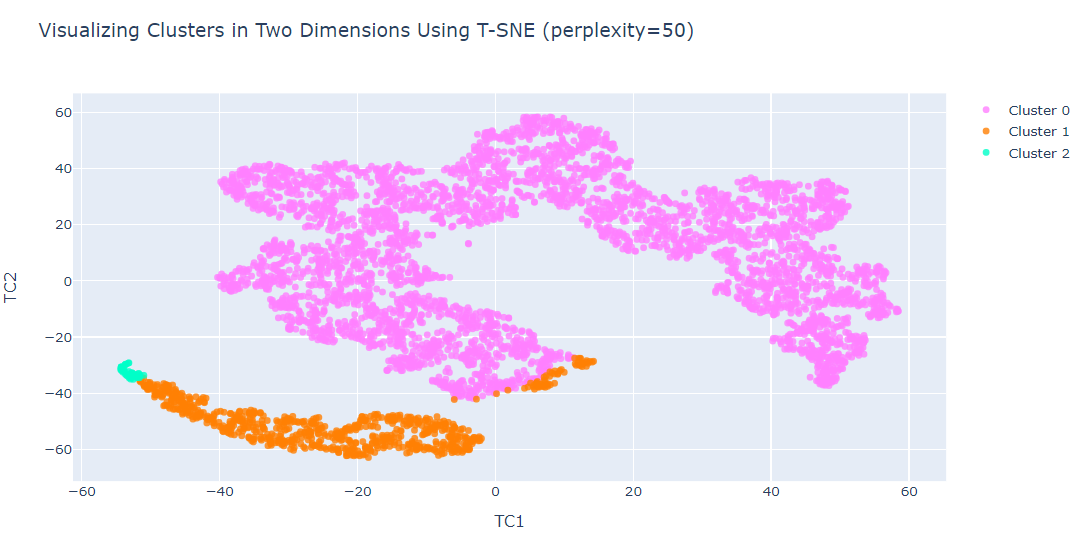


Рисунок 3.43

Цей останній графік нижче відображає наші кластери у трьох вимірах, створених T-SNE для 3-D візуалізації (рис 3.44)

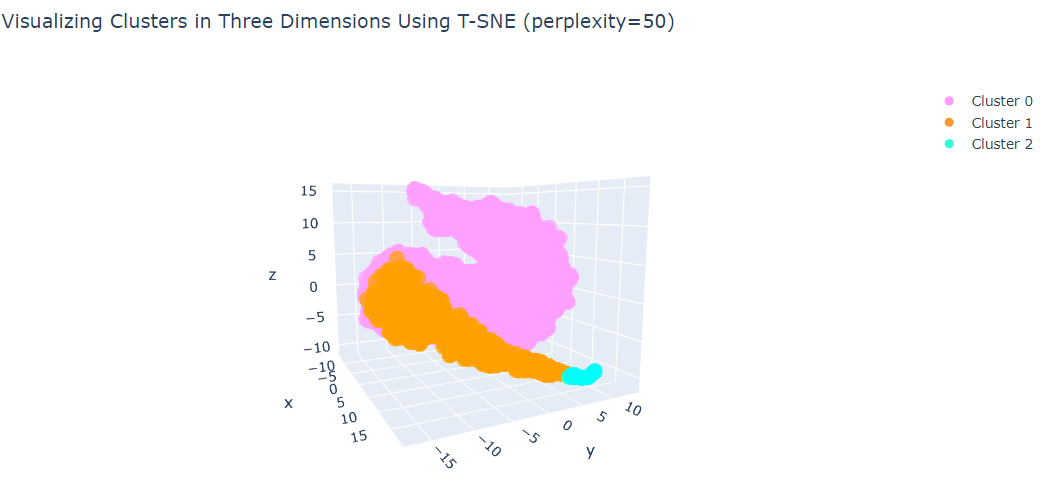


Рисунок 3.44.

Алгоритм T-SNE також зробив досить гідну роботу з візуалізацією кластерів. Але було кілька помітних відмінностей під час порівняння отриманих графіків із результатами PCA.

* 1. Висновки щодо якості побудованих моделей

Отже, візуалізація результатів роботи алгоритмів виглядає так (рис. 3.16, 3.21, 3.24)

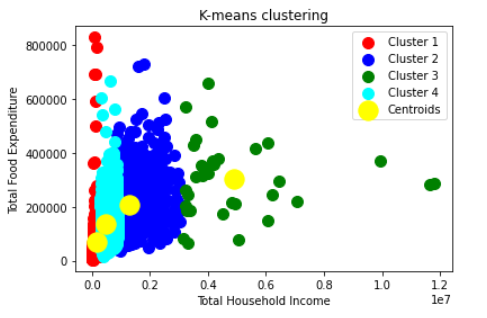


Рисунок 3.18 – Візуалізація результату кластеризації алгоритмом К-середніх.

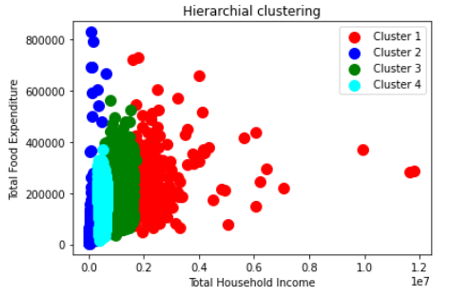


Рисунок 3.23 – Код для візуалізації результатів Ієрархічної кластеризації.

Як ми знаємо, кластеризація — це суб’єктивний статистичний аналіз, і для кожного набору даних і типу проблеми існує більше одного відповідного алгоритму. Отже, яка практична відмінність між K-середнім та ієрархічним алгоритмом?

Якщо в наборі даних є певна кількість кластерів, але група, до якої вони належать, невідома, краще обрати алгоритм K-середніх.

Якщо відмінності засновані на попередніх переконаннях, для визначення кількості кластерів слід використовувати ієрархічну кластеризацію.

Завдяки великій кількості змінних K-середні обчислюються швидше. Результат K-середніх є неструктурованим, в той час як ієрархічний результат є більш інтерпретованим та інформативним.

Для алгоритму К-середніх визначити кількість кластерів легше за методом ліктя. Для ієрархічної - легше за дендрограмою.

Ієрархічна кластеризація не може добре обробляти великі дані, але кластеризація K-Means може. Це пояснюється тим, що часова складність K-Means є лінійною, тобто O(n), тоді як ієрархічна кластеризація є квадратичною, тобто O(n2).

У кластеризації K-Means, якщо ми починаємо з випадкового вибору кластерів, результати, отримані при багаторазовому запуску алгоритму, можуть відрізнятися. Також алгоритм K-Середніх добре працює, коли форма кластерів є гіперсферичною (наприклад, коло в 2D, сфера в 3D).

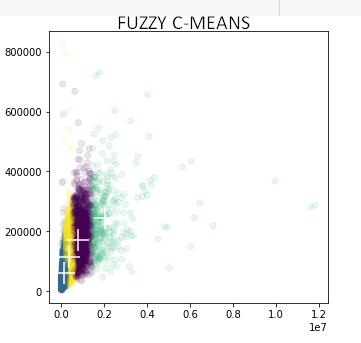


Рисунок 3.26 – Візуалізація результату роботи алгоритму нечітких С-середніх.

Алгоритм нечітких С-середніх кардинально відрізняється від алгоритму К-середніх та ієрархічного алгоритму, так як відноситься до алгоритмів «м’якої» кластеризації. Кластеризацію нечітких c-середніх можна вважати кращим алгоритмом порівняно з алгоритмом k-середніх. На відміну від алгоритму k-середніх, де точки даних належать виключно одному кластеру, у випадку нечіткого алгоритму c-середніх точка даних може належати більш ніж одному кластеру за імовірністю. Кластеризація нечітких c-середніх дає порівняно кращі результати для окремих наборів даних.

Щодо кластеризації з багатьма параметрами, то одна з основних відмінностей між графіками, створеними PCA і T-SNE, полягає в тому, що на графіках T-SNE, здавалося, кластери накладаються один на одного більше, ніж на графіках PCA. Наприклад, якщо ви подивитеся на 2-вимірну діаграму, сформовану з PCA, ви побачите окремі секції точок даних зі строгими видимими межами. Тоді як, якщо ви подивитеся на двовимірну діаграму, сформовану з T-SNE, ви, знову ж таки, побачите секції, сформовані всередині точок даних, але цього разу точки даних між кожним кластером, здається, «змішуються» і більше перекриваються.

Іншою основною відмінністю між графіками, створеними PCA, і графіками, створеними T-SNE, є форма. Оскільки і PCA, і T-SNE виконують зменшення розмірності дуже різними способами (і з різними цілями), результуюча форма або розподіл точок, створених алгоритмами, майже завжди будуть дуже різними.

Майте на увазі, що графіки, отримані в результаті алгоритму T-SNE, досить різноманітні, оскільки вони дуже сильно залежать від значення, вибраного для гіперпараметру заплутаності.

ВИСНОВКИ

Отже, ми провели кластерний аналіз датасету прибутків та витрат на їжу філіппінських сімей. Кластеризація проводилась окремо трьома різними алгоритмами.

В алгоритмі k-means ми використали метод ліктя для визначення оптимальної кількості кластерів, так як цей алгоритм дуже чутливий до вхідних параметрів. В результаті роботи алгоритму, провізуалізувавши схематично, ми отримали сегментну кластеризацію, за своєю основою яка більше залежить від прибутку сімейства, ніж до його витрат.

В ієрархічному алгоритмі ми використали дендрограму для визначення оптимальної кількості кластерів, та отримали такий же результат як і у попередньому алгоритмі. В результаті роботи алгоритму, ми отримали таку ж кластеризацію, за своєю основою, як і в алгоритмі k-means, але схематично, вони все-ж відрізняються.

В алгоритмі нечітких с-середніх ми використали окрему бібліотеку «fcmeans» для імплементації та виконання самого алгоритму. Провізуалізували його схематично. Так як цей метод належить до м’якої кластеризації, доволі тяжко зробити логічний висновок щодо його роботи до алгоритмів твердої кластеризації, але в цілому цей алгоритм може використовуватися для ширшого кола задач.

Також було виконано кластеризацію з більш ніж двома показниками за алгоритмом K-means. Застосовані практично методи PCA та t-SNE. Описана практична відмінність цих методів.

В кінці були винесені висновки щодо якості побудованих моделей, описана практична відмінність алгоритмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основні концепції кластеризації:

[Електронний ресурс] – Режим доступу:

<https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>

1. Основні концепції кластеризації:

[Електронний ресурс] – Режим доступу:

<https://habr.com/ru/post/67078/>

1. Порівняння К-середніх та С-середніх:

[Електронний ресурс] – Режим доступу:

<https://towardsdatascience.com/fuzzy-c-means-clustering-is-it-better-than-k-means-clustering-448a0aba1ee7>

1. Порівняння К-середніх та Ієрархічного алгоритму:

[Електронний ресурс] – Режим доступу:

<https://www.globaltechcouncil.org/clustering/k-means-clustering-vs-hierarchical-clustering/>

1. Документація бібліотеки scikitfuzzy:

[Електронний ресурс] – Режим доступу:

<https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_cmeans.html>

1. Приклади роботи алгоритму К-середніх:

[Електронний ресурс] – Режим доступу:

<https://www.superdatascience.com/blogs/self-organizing-maps-soms-extra-k-means-clustering-part-3>

1. Найпопулярніші алгоритми кластеризації:

[Електронний ресурс] – Режим доступу:

<https://insights.daffodilsw.com/blog/top-5-clustering-algorithms-in-machine-learning>

1. T-SNE:

[Електронний ресурс] – Режим доступу:

<https://towardsdatascience.com/why-you-are-using-t-sne-wrong-502412aab0c0>

1. PCA

[Електронний ресурс] – Режим доступу:

<https://www.kaggle.com/code/minc33/visualizing-high-dimensional-clusters/notebook>

1. Fuzzy C-means

[Електронний ресурс] – Режим доступу:

https://towardsdatascience.com/fuzzy-c-means-clustering-with-python-f4908c714081

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Файл K-means\_clustering.ipynb

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv('Family\_Income\_and\_Expenditure.csv')

data.head()

data.isnull().any()

data.describe()

data.count()

data['Total Household Income'].hist(bins=50)

data.drop(data[data['Total Household Income'] >= 500000].index, inplace = True)

data['Total Household Income'].hist(bins=50)

data = pd.read\_csv('Family\_Income\_and\_Expenditure.csv')

sorted(data['Total Household Income'], reverse= True)

data['Total Food Expenditure'].hist(bins=50)

data.drop(data[data['Total Food Expenditure'] >= 250000].index, inplace = True)

data['Total Food Expenditure'].hist(bins=50)

data = pd.read\_csv('Family\_Income\_and\_Expenditure.csv')

data['Meat Expenditure'].hist(bins=30)

data.drop(data[data['Meat Expenditure'] >= 50000].index, inplace = True)

data['Meat Expenditure'].hist(bins=50)

data = pd.read\_csv('Family\_Income\_and\_Expenditure.csv')

sorted(data['Household Head Age'], reverse= True)

x = data.loc[:, ['Total Household Income', 'Total Food Expenditure']].values

from sklearn.cluster import KMeans

wcss = []

for i in range(1,11):

kmeans = KMeans(n\_clusters=i, init = 'k-means++').fit(x)

wcss.append(kmeans.inertia\_)

plt.plot(wcss, range(1,11))

data.iloc[:,[0,2]].values

X = data.iloc[:, [0, 2]].values

kmeans = KMeans(n\_clusters = 4, init = 'k-means++', random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 300, c = 'yellow', label = 'Centroids')

plt.title('K-means clustering')

plt.xlabel('Total Household Income')

plt.ylabel('Total Food Expenditure')

plt.legend()

plt.show()

X = data.iloc[:, [0, 2, 7]].values

wcss = []

for i in range(1,11):

k\_means = KMeans(n\_clusters=i,init='k-means++', random\_state=42)

k\_means.fit(X)

wcss.append(k\_means.inertia\_)

#plot elbow curve

plt.plot(np.arange(1,11),wcss)

plt.xlabel('Clusters')

plt.ylabel('SSE')

plt.show()

k\_means\_optimum = KMeans(n\_clusters = 4, init = 'k-means++', random\_state=42)

y = k\_means\_optimum.fit\_predict(X)

print(y)

data['cluster'] = y

# the above step adds extra column indicating the cluster number for each row

data1 = data[data.cluster==0]

data2 = data[data.cluster==1]

data3 = data[data.cluster==2]

data4 = data[data.cluster==3]

kplot = plt.axes(projection='3d')

xline = np.linspace(0, 15, 1000)

yline = np.linspace(0, 15, 1000)

zline = np.linspace(0, 15, 1000)

kplot.plot3D(xline, yline, zline, 'black')

# Data for three-dimensional scattered points

kplot.scatter3D(data1['Total Household Income'], data1['Total Food Expenditure'], data1['Meat Expenditure'], c='red', label = 'Cluster 1')

kplot.scatter3D(data2['Total Household Income'],data2['Total Food Expenditure'],data2['Meat Expenditure'],c ='green', label = 'Cluster 2')

kplot.scatter3D(data3['Total Household Income'],data3['Total Food Expenditure'],data3['Meat Expenditure'],c ='blue', label = 'Cluster 3')

kplot.scatter3D(data4['Total Household Income'],data4['Total Food Expenditure'],data4['Meat Expenditure'],c ='yellow', label = 'Cluster 4')

plt.scatter(k\_means\_optimum.cluster\_centers\_[:,0], k\_means\_optimum.cluster\_centers\_[:,1], color = 'indigo', s = 200)

plt.legend()

plt.title("Kmeans")

plt.show()

#Basic imports

import numpy as np

import pandas as pd

#sklearn imports

from sklearn.decomposition import PCA #Principal Component Analysis

from sklearn.manifold import TSNE #T-Distributed Stochastic Neighbor Embedding

from sklearn.cluster import KMeans #K-Means Clustering

from sklearn.preprocessing import StandardScaler #used for 'Feature Scaling'

#plotly imports

import plotly as py

import plotly.graph\_objs as go

from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot

X = pd.DataFrame(data.iloc[:, [0, 2, 7, 8]].values)

#Initialize our model

kmeans = KMeans(n\_clusters=3)

#Fit our model

kmeans.fit(X)

#Find which cluster each data-point belongs to

clusters = kmeans.predict(X)

#Add the cluster vector to our DataFrame, X

X["Cluster"] = clusters

#plotX is a DataFrame containing 5000 values sampled randomly from X

plotX = pd.DataFrame(np.array(X.sample(5000)))

#Rename plotX's columns since it was briefly converted to an np.array above

plotX.columns = X.columns

#PCA with one principal component

pca\_1d = PCA(n\_components=1)

#PCA with two principal components

pca\_2d = PCA(n\_components=2)

#PCA with three principal components

pca\_3d = PCA(n\_components=3)

#This DataFrame holds that single principal component mentioned above

PCs\_1d = pd.DataFrame(pca\_1d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

#This DataFrame contains the two principal components that will be used

#for the 2-D visualization mentioned above

PCs\_2d = pd.DataFrame(pca\_2d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

#And this DataFrame contains three principal components that will aid us

#in visualizing our clusters in 3-D

PCs\_3d = pd.DataFrame(pca\_3d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

PCs\_1d.columns = ["PC1\_1d"]

#"PC1\_2d" means: 'The first principal component of the components created for 2-D visualization, by PCA.'

#And "PC2\_2d" means: 'The second principal component of the components created for 2-D visualization, by PCA.'

PCs\_2d.columns = ["PC1\_2d", "PC2\_2d"]

PCs\_3d.columns = ["PC1\_3d", "PC2\_3d", "PC3\_3d"]

plotX = pd.concat([plotX,PCs\_1d,PCs\_2d,PCs\_3d], axis=1, join='inner')

plotX["dummy"] = 0

#Note that all of the DataFrames below are sub-DataFrames of 'plotX'.

#This is because we intend to plot the values contained within each of these DataFrames.

cluster0 = plotX[plotX["Cluster"] == 0]

cluster1 = plotX[plotX["Cluster"] == 1]

cluster2 = plotX[plotX["Cluster"] == 2]

#This is needed so we can display plotly plots properly

init\_notebook\_mode(connected=True)

#Instructions for building the 1-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter(

x = cluster0["PC1\_1d"],

y = cluster0["dummy"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter(

x = cluster1["PC1\_1d"],

y = cluster1["dummy"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter(

x = cluster2["PC1\_1d"],

y = cluster2["dummy"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in One Dimension Using PCA"

layout = dict(title = title,

xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),

yaxis= dict(title= '',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

#Instructions for building the 2-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter(

x = cluster0["PC1\_2d"],

y = cluster0["PC2\_2d"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter(

x = cluster1["PC1\_2d"],

y = cluster1["PC2\_2d"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter(

x = cluster2["PC1\_2d"],

y = cluster2["PC2\_2d"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in Two Dimensions Using PCA"

layout = dict(title = title,

xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),

yaxis= dict(title= 'PC2',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

#Instructions for building the 3-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter3d(

x = cluster0["PC1\_3d"],

y = cluster0["PC2\_3d"],

z = cluster0["PC3\_3d"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter3d(

x = cluster1["PC1\_3d"],

y = cluster1["PC2\_3d"],

z = cluster1["PC3\_3d"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter3d(

x = cluster2["PC1\_3d"],

y = cluster2["PC2\_3d"],

z = cluster2["PC3\_3d"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in Three Dimensions Using PCA"

layout = dict(title = title,

xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),

yaxis= dict(title= 'PC2',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

#plotX will hold the values we wish to plot

plotX = pd.DataFrame(np.array(X.sample(5000)))

plotX.columns = X.columns

#Set our perplexity

perplexity = 50

#T-SNE with one dimension

tsne\_1d = TSNE(n\_components=1, perplexity=perplexity)

#T-SNE with two dimensions

tsne\_2d = TSNE(n\_components=2, perplexity=perplexity)

#T-SNE with three dimensions

tsne\_3d = TSNE(n\_components=3, perplexity=perplexity)

#This DataFrame holds a single dimension,built by T-SNE

TCs\_1d = pd.DataFrame(tsne\_1d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

#This DataFrame contains two dimensions, built by T-SNE

TCs\_2d = pd.DataFrame(tsne\_2d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

#And this DataFrame contains three dimensions, built by T-SNE

TCs\_3d = pd.DataFrame(tsne\_3d.fit\_transform(plotX.drop(["Cluster"], axis=1)))

TCs\_1d.columns = ["TC1\_1d"]

PCs\_1d.columns = ["PC1\_1d"]

#"TC1\_2d" means: 'The first component of the components created for 2-D visualization, by T-SNE.'

#And "TC2\_2d" means: 'The second component of the components created for 2-D visualization, by T-SNE.'

TCs\_2d.columns = ["TC1\_2d","TC2\_2d"]

TCs\_3d.columns = ["TC1\_3d","TC2\_3d","TC3\_3d"]

plotX = pd.concat([plotX,TCs\_1d,TCs\_2d,TCs\_3d], axis=1, join='inner')

plotX["dummy"] = 0

cluster0 = plotX[plotX["Cluster"] == 0]

cluster1 = plotX[plotX["Cluster"] == 1]

cluster2 = plotX[plotX["Cluster"] == 2]

#Instructions for building the 1-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter(

x = cluster0["TC1\_1d"],

y = cluster0["dummy"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter(

x = cluster1["TC1\_1d"],

y = cluster1["dummy"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter(

x = cluster2["TC1\_1d"],

y = cluster2["dummy"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in One Dimension Using T-SNE (perplexity=" + str(perplexity) + ")"

layout = dict(title = title,

xaxis= dict(title= 'TC1',ticklen= 5,zeroline= False),

yaxis= dict(title= '',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

#Instructions for building the 2-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter(

x = cluster0["TC1\_2d"],

y = cluster0["TC2\_2d"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter(

x = cluster1["TC1\_2d"],

y = cluster1["TC2\_2d"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter(

x = cluster2["TC1\_2d"],

y = cluster2["TC2\_2d"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in Two Dimensions Using T-SNE (perplexity=" + str(perplexity) + ")"

layout = dict(title = title,

xaxis= dict(title= 'TC1',ticklen= 5,zeroline= False),

yaxis= dict(title= 'TC2',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

#Instructions for building the 3-D plot

#trace1 is for 'Cluster 0'

trace1 = go.Scatter3d(

x = cluster0["TC1\_3d"],

y = cluster0["TC2\_3d"],

z = cluster0["TC3\_3d"],

mode = "markers",

name = "Cluster 0",

marker = dict(color = 'rgba(255, 128, 255, 0.8)'),

text = None)

#trace2 is for 'Cluster 1'

trace2 = go.Scatter3d(

x = cluster1["TC1\_3d"],

y = cluster1["TC2\_3d"],

z = cluster1["TC3\_3d"],

mode = "markers",

name = "Cluster 1",

marker = dict(color = 'rgba(255, 128, 2, 0.8)'),

text = None)

#trace3 is for 'Cluster 2'

trace3 = go.Scatter3d(

x = cluster2["TC1\_3d"],

y = cluster2["TC2\_3d"],

z = cluster2["TC3\_3d"],

mode = "markers",

name = "Cluster 2",

marker = dict(color = 'rgba(0, 255, 200, 0.8)'),

text = None)

data = [trace1, trace2, trace3]

title = "Visualizing Clusters in Three Dimensions Using T-SNE (perplexity=" + str(perplexity) + ")"

layout = dict(title = title,

xaxis= dict(title= 'TC1',ticklen= 5,zeroline= False),

yaxis= dict(title= 'TC2',ticklen= 5,zeroline= False)

)

fig = dict(data = data, layout = layout)

iplot(fig)

Інші документи та ноутбуки перелічені в github. Посилання: <https://github.com/vladislavister2/Clustering_AD_CourseWork>