Харківський національний університет імені В.Н. Каразіна

Факультет математики і інформатики

Кафедра прикладної математики

Кваліфікаційна робота

магістра

на тему «*Використання топологічного аналізу даних для клінічних досліджень: виявлення прихованих закономірностей за допомогою машинного навчання на основі графів*»

Виконав:
студент групи МП-61
2-го курсу
спеціальність 113 – прикладна математика
освітньо-наукова програма
«Прикладна математика»
Остапчук С.А.
Науковий керівник: канд. ф.-м. наук, доц.
Стєпанова К.В., Булаєнко Д.В.
Рецензент: Квітчук С.В.

Харків – 2024 рік

Анотація

Остапчук С.А. Використання топологічного аналізу даних для клінічних досліджень: виявлення прихованих закономірностей за допомогою машинного навчання на основі графів. – Кваліфікаційна робота магістра.

Розглянуто класичний підхід до аналізу даних клінічних досліджень, а також розповсюджені сучасні методи кластерного та топологічного аналізу даних.

Проведено розрахунковий експеримент з аналізу даних реального клінічного дослідження. Цей процес включав очищення, підготовку та адаптацію даних шляхом побудови комбінованої метрики. Крім того, було проведено багатовимірне масштабування для проекування даних і виконано топологічний аналіз з використанням методу стійкої гомології. Завдяки цьому були отримані діаграми стійкості та побудовані відповідні графи, які допомогли ідентифікувати та провести порівняльний аналіз основних спільнот.

Запропоновано більш широко використовувати топологічний аналіз даних в розвідувальному аналізі клінічних досліджень, оскільки навіть базові його методи показують хороші результати.

Summary

S.A. Ostapchuk. Using of Topological Data Analysis for Clinical Research: Detecting of Hidden Patterns with Graph-based Machine Learning. – Masters qualification work.

The classical approach to analyzing clinical trial data, as well as common modern methods of cluster and topological data analysis are considered.

A computational experiment involving the analysis of a real clinical trial's data. This process included data cleaning, preparation, and adaptation through the construction of a combined metrics. Furthermore, a multidimensional scaling to project the data and performed topological analysis using persistent homology. This led to the creation of persistence diagrams and corresponding graphs, which helped identify and comparatively analyze the main communities.

The findings suggest that topological data analysis, even at its basic level, is highly effective and should be utilized more extensively in the exploratory analysis of clinical trials.

Зміст

Вступ

У сфері клінічних досліджень, де точність та надійність аналізу даних мають критичне значення, традиційні статистичні методи вже давно є основними інструментами. Вони включають регресійний аналіз, аналіз виживання, кореляційний аналіз тощо. Ці методи ефективні для виявлення лінійних залежностей та оцінки ризиків, але їх здатність до ідентифікації більш складних, нелінійних взаємозв'язків в мультиваріативних та високорозмірних даних може бути обмеженою. І тут, дослідницький розвідувальний аналіз даних може розширити можливості класичних статистичних досліджень, надаючи інструменти для глибшого розуміння складності біомедичних даних.

Топологічний аналіз даних (ТАД), заснований на принципах алгебраїчної топології, відкриває нові горизонти для математиків та дослідників у клінічних науках. ТАД дозволяє аналізувати дані на різних масштабах та вимірах, допомагаючи виявляти важливі топологічні структури, що можуть залишатись непоміченими за допомогою традиційних методів. Завдяки своїй здатності виявляти цикли, дірки та інші важливі топологічні особливості, ТАД забезпечує можливість глибшого розуміння складних даних, таких як ті, що зустрічаються в генетиці, нейронауках, епідеміології та ін.

Ця робота покликана дослідити, як топологічний аналіз може бути інтегрований у клінічні дослідження, розглядаючи як теоретичні аспекти, так і практичне застосування. Ми розглянемо базові концепти та основні методи ТАД, зокрема стійку гомологію та продемонструємо їхню ефективність на прикладі реального клінічного дослідження. Мета полягає в тому, щоб показати, як новітні математичні методи можуть вдосконалити існуючі підходи до аналізу даних, надаючи дослідникам потужні інструменти для розкриття нових біомедичних відкриттів.

1. Класичні методи аналізу даних клінічних досліджень

Головною комерційною метою класичного клінічного дослідження є зрозуміти, чи ефективні ліки, що досліджуються. Відкидаючи складову безпеки, типові статистичні гіпотези для таких досліджень можна сформулювати наступним чином:

H_0: Немає статистично значущої різниці в кінцевих показниках для групи досліджуваних, що приймали препарат та тими, що приймали плацебо.

H_1: Існує статистично значуща різниця в кінцевих показниках для групи піддослідних, що приймали препарат та тими, що приймали плацебо.

Такі гіпотези перевіряються за допомогою низки статистичних моделей лінійних, множинних, поліноміальних, логістичних та інших регресій, регресій з випадковими коефіцієнтами, а для часових рядів та аналізу подій: кривих Каплан-Майєра, регресій Кокса, log-ранг тесту та багатьох інших, обраних в залежності від мети та дизайну дослідження, кінцевих показників, та інших методів, детально описаних в книгах [1, 2]. Значну частку класичного статистичного аналізу охоплює проста описова статистика. Результати представляються у вигляді графіків та таблиць. Увесь процес, починаючи від підготовки дослідження і закінчуючи представленням його результатів, строго регулюється та регламентується місцевими регулюючими органами (наприклад керівництвом Food and Drug Administration в США [3], Pharmaceuticals and Medical Devices Agency в Японії) та загальногалузевими стандартами [4].

Проте, не дивлячись на комерційну складову дослідження, при його проведенні накопичується низка параметрів та характеристик, які мають наукову цінність як для подальших досліджень хвороби, так і для медичної науки вцілому. Додаткові дослідження, що виконуються поза межами основних досліджень називаються розвідувальними і не мають жодних обмежень та вимог щодо інструментарію, який може бути застосований для їх проведення.

Розвідувальний аналіз даних (РАД) відіграє ключову роль у виявленні шаблонів, аномалій та зв'язків у клінічних даних. Цей процес включає в себе різноманітні інструменти, зокрема графічні методи, техніки зменшення розмірності та кількісні методи, які разом допомагають визначити важливі структурні характеристики даних.

Графічні методи являють собою потужні інструменти в РАД, забезпечуючи інтуїтивне розуміння даних через візуалізацію. Вони включають створення гістограм, кругових діаграм, діаграм Парето, діаграм розсіювання та багатьох інших типів візуалізацій, які відображають різні аспекти даних. Особливо корисними є динамічні методи візуалізації, такі як гранд-тур, тур з гідом і ручний тур, що дозволяють досліджувати багатовимірні дані з різних перспектив.

Методи зменшення розмірності, такі як багатовимірне масштабування та аналіз головних компонент, дозволяють зменшити складність даних, зберігаючи при цьому їхні важливі характеристики. Ці техніки корисні для виявлення основних варіативних структур у даних та виокремлення тих, що найбільше впливають на досліджувані явища.

Кількісні методи, що застосовуються в розвідувальному аналізі можуть включати такі процедури, як медіанна поліровка, ординація та тримейн. У класичному підході розвідувальний аналіз часто включає створення регресійних моделей (лінійних, нелінійних, множинних тощо).

Розвиток методів машинного навчання сприяв широкому використанню кластерного аналізу у клінічних дослідженнях. Цей метод групує об'єкти за схожістю характеристик, формуючи кластери, де об'єкти максимально схожі один на одного і мінімально схожі на об'єкти з інших кластерів. Основні техніки, такі як k -середніх, ієрархічна кластеризація та кластеризація на основі графів, допомагають ідентифікувати та аналізувати підгрупи в даних, що можуть мати важливе клінічне значення.

Ієрархічна кластеризація ґрунтується на концепції близькості і будує модель на основі зв'язку на відстані, групуючи об'єкти в кластери на основі

відстані між ними. Визначення функції відстані (також званої відстанню) є одним з методів, що використовується для визначення схожості об'єктів. Функція відстані - це інструмент, який використовується для вимірювання схожості між точками даних, що належать до набору даних про результати. Після обчислення відстані для набору даних, набір даних може бути представлений у вигляді метричного простору, який можна вивчати за допомогою геометричних і топологічних методів. На додаток до вибору функції відстані, дослідник повинен визначити поняття кластеру та способи його ідентифікації. Алгоритми кластеризації для набору даних потрібно підбирати експериментально залежно від типу даних.

Загалом, кластерний аналіз включає вивчення даних для пошуку одного або декількох параметрів, які можна вважати факторами, що впливають на результати.

Після ідентифікації ключових груп та шаблонів у даних, наступним етапом є побудова статистичних моделей для детальнішого вивчення зв'язків і впливу окремих змінних. Потім визначається рівень значущості для параметрів (предикторів), знайдених у даних, щоб зрозуміти рівень впливу цих параметрів на статистичну модель. Після визначення значущих параметрів, які також називають значущими коваріатами, дослідник може врахувати виявлені значущі коваріати при розробці рекомендацій щодо проведення клінічного дослідження. Наприклад, дослідник може порекомендувати обмежити вхідну популяцію за віком, якщо вік був визначений як значуща коваріативна ознака.

Таким чином, такий підхід до РАД клінічного дослідження не лише розкриває приховані зв'язки між змінними, але й вказує на можливі напрямки для подальших досліджень, забезпечуючи дослідникам міцну основу для прийняття обґрунтованих клінічних рішень.

2. Аналіз даних за допомогою машинного навчання

2.1. Кластерний аналіз даних

Кластеризація, яка вважається найважливішим питанням неконтрольованого навчання, пов'язана з розбиттям структури даних на невідомі області. Повне визначення кластеризації, однак, ще не узгоджене, і класичне визначення описується наступним чином:

1) екземпляри в одному кластері повинні бути максимально схожими;
2) екземпляри в різних кластерах повинні максимально відрізнятися один від одного;
3) вимірювання подібності та відмінності повинно бути чітким та мати практичне значення.

Роздуми про кластерний аналіз, базові поняття (наприклад функції відстаней, функції подібності) та порівняння різних методів кластеризації добре описані в статті Dongkuan Xu та Yingjie Tian [5].

2.1.1. Ієрархічна кластеризація

Ієрархічна кластеризація – це метод аналізу даних, який використовується для групування схожих об'єктів у вкладені кластери. Математично цей процес можна розділити на два основні підходи: агломеративний (знизу вгору) та дивізивний (зверху вниз).

У агломеративному підході кожен об'єкт спочатку розглядається як окремий кластер. Потім, на кожному етапі об'єднання, два найближчих кластера (за обраною метрикою відстані) зливаються разом, утворюючи новий кластер. Цей процес повторюється до тих пір, поки всі об'єкти не опиняться в одному кластері.

Дивізивний підхід починається з усіх об'єктів у одному кластері. На кожному кроці кластер з найбільшою внутрішньою неоднорідністю розділяється на два підкластери. Цей процес продовжується, поки кожен об'єкт не стане окремим кластером.

Для визначення «відстані» між кластерами використовується метрика, наприклад Евклідова відстань для кількісних даних:

$$d_{Euclidean}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots}$$

або манхеттенська відстань для категорійних:

$$d_{Manhattan}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|_1 = |x_1 - y_1| + |x_2 - y_2| + \cdots$$

Функція відстані повинна задовольняти аксіомам метричного простору, включаючи не від'ємність, симетричність та нерівність трикутника.

Приклад ієрархічної кластеризації, а саме побудови дендрограми для хмари точок наведено на рис. 2.1 [6]. В лівій частині відображено шість точок A-F, а кола 1-5 показують різні групування на основі відстаней. Ці групи утворюють неявну ієрархію. Явну ієрархію показано на правій частині.

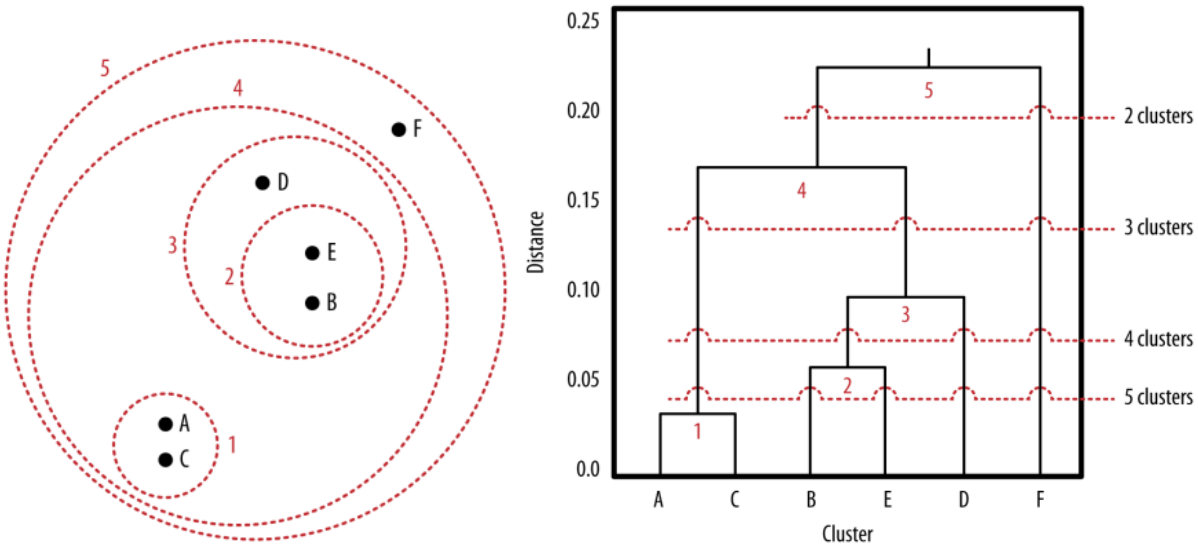


Рисунок 2.1. Ієрархічна кластеризація [6].

2.1.2. Метод k-середніх та його похідні

Ієрархічна кластеризація фокусується на схожості між окремими екземплярами і як ця схожість пов'язує їх разом. Інший спосіб мислення про кластеризацію даних це зосередитися на самих кластерах – групах екземплярів. Найпоширеніший метод зосередження на самих кластерах - це представлення кожного кластера його «кластерним центром», або центроїдою.

Скористаємось визначенням методу, що наведений в книзі [6]. Припустимо, що ми маємо набір даних $\{x_1,\ldots,x_N\}$, що складається з N спостережень випадкової D-вимірної евклідової змінної x. Слід розбити набір

даних на деяку кількість кластерів K. Вважаємо, що значення K задано. Введемо набір D- вимірних векторів μ_k, де $k = 1,\ldots,K$. Кінцевою метою методу є знайти такий розподіл точок по кластеру, щоб сума квадратів відстаней кожної точки до найближчого до неї вектора μ_k була мінімальною.

Для кожної точки x_n введемо відповідний набір бінарних індикаторів змінних $r_{nk} \in \{0, 1\}$, де $k = 1,\ldots,K$, що описують, до якого з кластерів відноситься точка x_n. Тоді точка x_n відноситься до кластеру k, то $r_{nk} = 1$, а $r_{nj} = 0$ для $j \neq k$. Це можна описати функцією міри спотворення:

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|x_n - \mu_k\|^2,$$

яка є сумою сумою квадратів відстаней кожної точки даних до відповідного вектору μ_k. Мета алгоритму k-середніх - знайти такі значення $\{r_{nk}\}$ та $\{\mu_k\}$, щоб мінімізувати J:

$$J = \begin{cases} 1, & x \text{ якщо } k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0, & \text{в іншому випадку} \end{cases}.$$

Розглянемо оптимізацію μ_k при фіксованому значенні r_{nk}. Цільова функція J є квадратичною функцією від μ_k, і її можна мінімізувати, взявши її похідну за μ_k і прирівнявши до нуля:

$$2\sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0.$$

Вирішивши відносно μ_k отримаємо:

$$\mu_k = \frac{\sum_n r_{nk}x_n}{\sum_n r_{nk}}.$$

Знаменник у цьому виразі дорівнює кількості точок, віднесених до кластера k, і тому цей результат має просту інтерпретацію, а саме: встановити μ_k рівним середньому значенню всіх точок даних x_n, віднесених до кластера k. З цієї причини процедура відома як алгоритм k-середніх.

Для вибору параметра k, тобто застосовної кількості кластерів, можна використовувати різні критерії. Зазвичай такими критеріями є деякі функції, які дають оцінку якості кластеризації залежно від заданого k. Значення k, яке

оптимізує значення такої функції (наприклад, максимальне або мінімальне значення), обирається як оптимальний параметр методу.

На рис. 2.2 проілюстровано алгоритм k-середніх для хмари точок. Зелені точки позначають набір даних у двовимірному евклідовому просторі. Початковий вибір центрів μ_1 та μ_2 показано червоним та синім хрестиками відповідно (a). На початковому кроці кожна точка даних відноситься або до червоного, або до синього кластера, відповідно до того, до якого центру кластера вона знаходиться ближче. Це еквівалентно класифікації точок відповідно до того, по який бік від бісектриси перпендикуляра (показаного пурпуровою лінією) до двох центрів кластерів вони лежать (b). На наступному кроці кожен центр кластера повторно обчислюється як середнє значення точок, віднесених до відповідного кластера (c). Далі алгоритм повторюється до остаточної збіжності (d-i) [6].

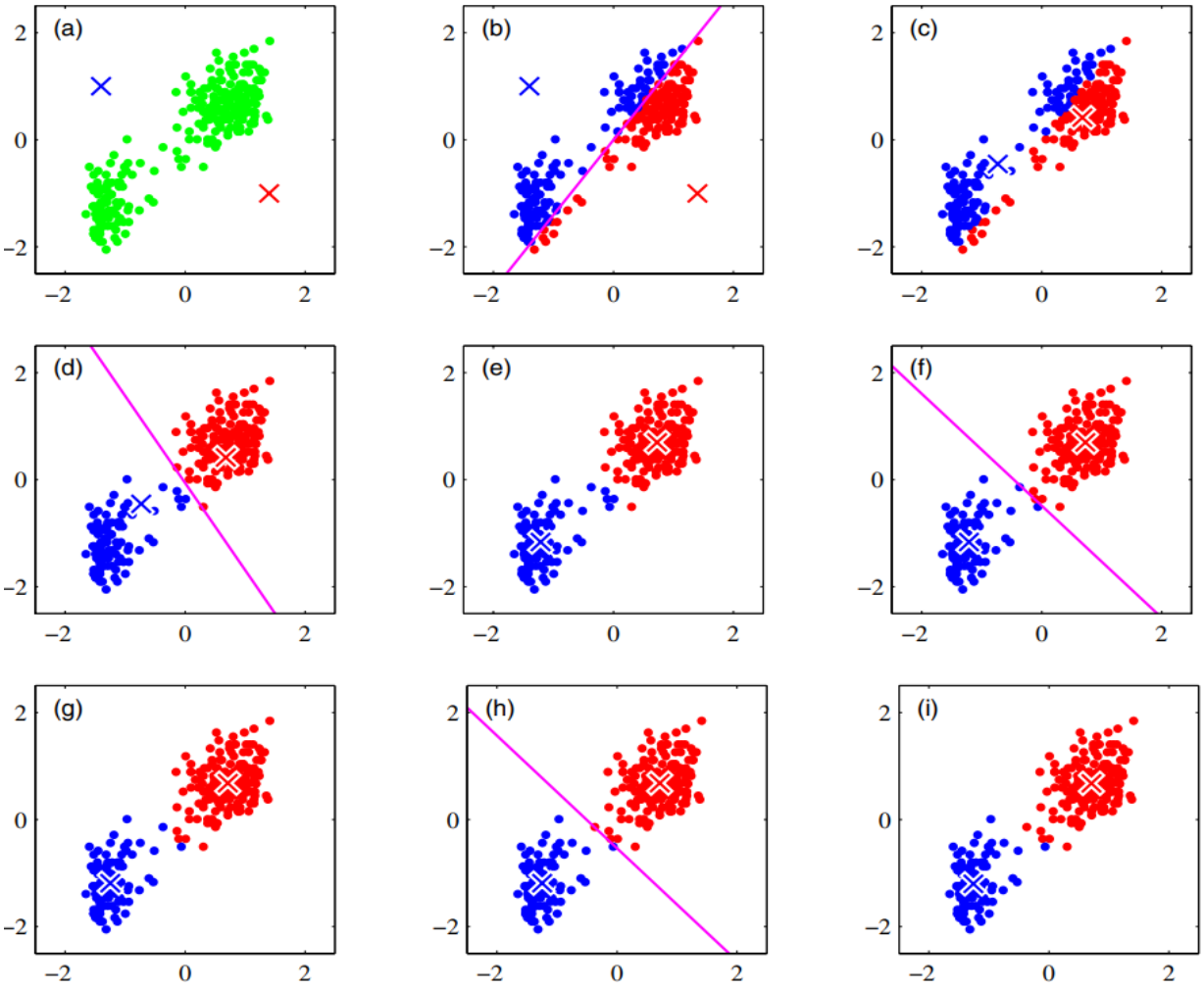


Рисунок 2.2. Робота алгоритму k-середніх [6].

Однією із модифікацій алгоритму k-середніх є алгоритм k- медоїдів. У кожному потенційному кластері метод k-медоїдів обирає «найбільш центрального» представника в кластері; ця центральна точка називається медоїдом. Після цього метод намагається мінімізувати суму розбіжностей від медоїда до інших точок відповідного кластера. На відміну від алгоритму k-середніх, алгоритм k-медоїдів обирає фактичні точки даних як медоїди, а не середні точки кластерів, як в алгоритмі k-середніх.

2.1.3. DBSCAN

Згідно з визначення, що запропоноване в статті [5], DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — це алгоритм кластеризації, який відрізняється від багатьох інших методів тим, що він може виявляти кластери довільної форми і ефективно обробляти шумові точки даних. В ході його роботи точки даних, що лежать в області з високою щільністю хмари точок даних, вважаються такими, що належать до одного кластера.

Щільність об'єкта *O* можна виміряти за кількістю об'єктів, близьких до об’єкта *O*. DBSCAN знаходить основні об'єкти, тобто об'єкти, які мають щільні околи та з'єднує основні об'єкти та їхні околи, щоб сформувати щільні регіони у вигляді кластерів. Для визначення радіуса околу, який ми розглядаємо для кожного об'єкта, використовується заданий користувачем параметр $\epsilon > 0$ (сусідство). Для даного об’єкта *O,* ϵ - множина точок, які знаходяться на відстані ϵ від *O*, тобто $N_\epsilon(O) = \{Q | \text{dist}(O, Q) \leq \epsilon\}$, де dist(*O, Q*) – відстань між об’єктами *O* та *Q*. Тобто, окіл об'єкта *O* - це простір в радіусі з центром в *O*.

Завдяки фіксованому розміру околу, який задається параметром ϵ, щільність околу можна визначити кількістю об'єктів в ньому. Щоб визначити, чи є окіл щільним, DBSCAN використовує ще один параметр, що задається користувачем – *MinPts*, який визначає поріг щільності для щільних областей. Об'єкт є основним об'єктом, якщо ϵ-окіл об'єкта містить принаймні *MinPts* об'єктів: $|N_\epsilon(O)| \geq MinPts$.

Маючи множину D об'єктів, ми можемо визначити всі основні об'єкти за заданими параметрами та *MinPts*. Таким чином, задача кластеризації зводиться до використання основних об'єктів та їх околів для формування щільних областей, де щільні області є кластерами.

DBSCAN алгоритм можна описати так:

1) Всі точки маркуються як необроблені;
2) Для кожної ще не обробленої точки, визначається її ϵ, і якщо вона виявляється основною точкою, починається формування кластеру. Якщо точка не є основною, точка маркується як шум (але це може змінитися).
3) Якщо точка O є основною, додаються всі точки з $N_\epsilon(O)$ та ітеративно повторюється процес для кожної з цих точок.

Графічно основні поняття, що використовує алгоритм показано на рис. 2.3, де червоні точки є основними точками через те, що їх ϵ околи містять в собі як мінімум три точки (*MinPts* = 3). Жовті точки не є основними точками, але є досяжними з основних точок. Сині точки вважається шумом/викидом, оскільки не є ні основною, ні досяжною.

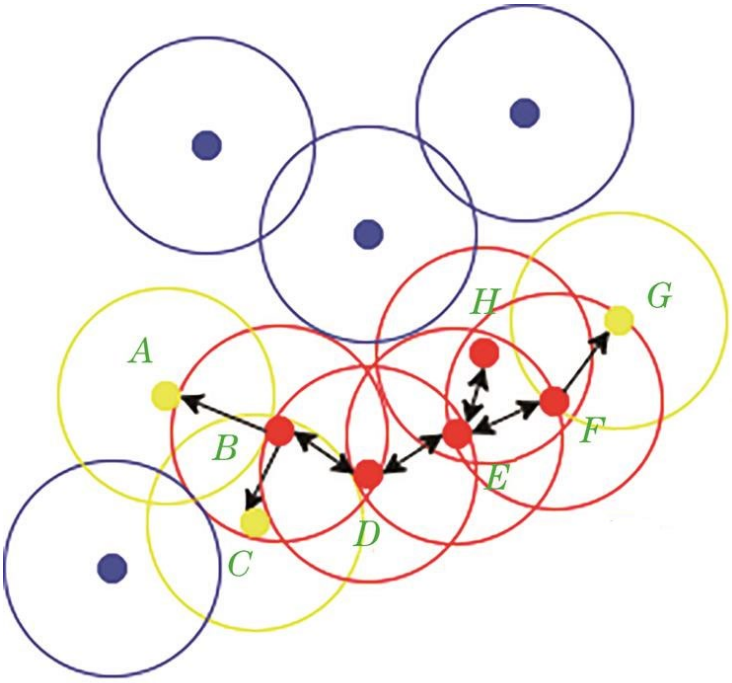


Рисунок 2.3. DBSCAN алгоритм [7].

На відміну від методів k-середніх та його похідних, DBSCAN не вимагає заздалегідь визначати кількості кластерів у даних.

2.2. Топологічний аналіз даних

Топологічний аналіз даних (ТАД) використовує ідеї з алгебраїчної топології для вивчення форми даних. Цей метод дозволяє виявити «форми» у даних на різних масштабах, ідентифікувати кластери, цикли, дірки та інші топологічні особливості. Основні концепції ТАД включають стійку гомологію, комплекси В'єторіса-Ріпса та мапування багато вимірних величин.

Також поширеними методами пошуку спільнот у графах є алгоритм Гірвана–Ньюмена, що описаний в статті одноіменних авторів [10] та метод клікової перколяції що добре описано в [11].

2.2.1. Стійка гомологія

Стійка (персистентна) гомологія — це метод, який дозволяє аналізувати структуру даних на різних масштабних рівнях і визначити, які топологічні особливості зберігаються через різні масштаби. Це робиться шляхом побудови серії вкладених підмножин (фільтрація) із даних та розрахунку їх гомологій.

Користуючись поясненнями, наведеними в книзі [7] визначимо поняття фільтрації та діаграми стійкої гомології. Розглянемо симплікаційний комплекс K і функцію $f \to \mathbb{R}$. Вимагається, щоб f була монотонною (не спадає вздовж зростаючих ланцюжків граней), тобто $f(\sigma) \leq f(\tau)$, якщо τ э границею σ. Монотонність означає, що підмножина підрівнів $K(a) = f^{-1}(-\infty, a]$ є підкомплексом K для кожного $a \in \mathbb{R}$. Нехай m – кількість симплексів у симплікаційному комплексі K, тоді можна отримати $n + 1 \leq m + 1$ різних підкомплексів, які можна влаштувати в порядку їх збільшення.

$$\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K$$

Тобто, якщо $a_1 < a_2 < \ldots < a_n$ це значення функції симплекса в K і $a_0 = -\infty$, то $K_i = K(a_i)$ для кожного i. Така послідовність комплексів

називається фільтрацією f і може розглядатись, як конструкція, до якої щоразу додаються шматочки у симплексів.

Якщо уявити набір стійких чисел Бетті $\beta_p^{k,l} = \sum_{i \le k} \sum_{j > l} \mu_p^{i,j}$, малюючи точки в двох вимірах, то деякі з цих точок можуть мати нескінченні координати, а деякі можуть мати однакові, тому мова йде про множину точок на розширеній дійсній площині $\overline{\mathbb{R}}^2$. Припустимо $\mu_p^{i,j}$ – кількість p-вимірних класів, народжених в K_i і таких, що загинули переходячи в K_j, тоді

$$\mu_p^{i,j} = \left(\beta_p^{i,j-1} - \beta_p^{i,j}\right) - \left(\beta_p^{i-1,j-1} - \beta_p^{i-1,j}\right),$$

для всіх $i < j$ і для всіх p. Перша різниця в правій частині рахує класи, які народжуються під час чи перед K_i та помирають переходячи до K_j. Намалювавши кожну точку (a_i, a_j) з кратністю $\mu_p^{i,j}$, отримаємо p-ту діаграму стійкої гомології.

Приклад народження і зникнення комплексів запозичено з [8] та представлено на рис. 2.4 .

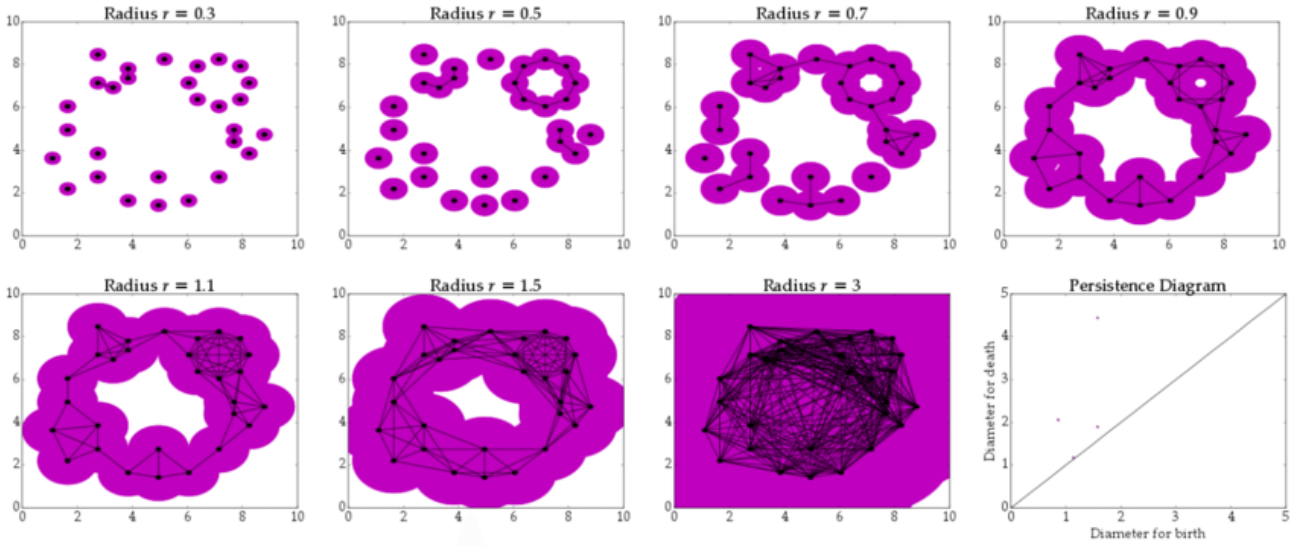


Рисунок 2.4 Алгоритм дослідження стійкої гомології та діаграма стікості [8]

Видно, що при r = 0.3 народжується низка класів, більшість яких складаються з одного вузла, вже при r = 0.9 усі старі комплекси помирають і створюється один комплекс, який живе до r=3.

2.2.2. Алгоритм Гірвана–Ньюмена

Алгоритм Гірвана–Ньюмена [9] використовує поняття «міжкластерної сили ребра»

$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}},$$

де σ_{st} – кількість найкоротших шляхів від вершини s до вершини t, а $\sigma_{st}(e)$ – кількість тих шляхів, що проходять через ребро e. Цей показник вимірюється через кількість найкоротших шляхів, що проходять через кожне ребро графа та використовується для визначення «важливості» ребра у структурі графа.

Алгоритм Гірвана–Ньюмена може бути описаний наступними кроками:

1) Для кожного ребра в графі обчислити C_b;
2) Видалити ребро з найбільшою C_b;
3) Обчислити C_b для кожного ребра, що залишилося;
4) Повторювати 2 і 3 доки не задовільниться умова $C_b = \max(C_b)$.

Графічно основні поняття, які використовує алгоритм показано на рис. 2.5, на якому видно, що ребрам призначаються C_b на основі кількості найкоротших шляхів, що проходять через них. Алгоритм усуває ребро між вузлами C і D, оскільки воно має найбільшу міжкластерну силу.

Рисунок 2.5. Алгоритм Гірвана–Ньюмена

2.2.3. k-клікова перколяція

Згідно визначенням, що представлене в [10], клікова перколяція (просочування) використовує поняття з теорії графів – кліка. Кліка – це підмножина вершин графа, така що кожна вершина з'єднана з кожною іншою вершиною в цій підмножині. Розмір кліки — це кількість вершин у ній. Цей метод базується на тому, що внутрішні ребра всередині спільноти утворюють

k-кліки (тобто підграфи з *k* вузлами, в яких кожна пара вершин з'єднана ребром), а ребра, які пролягають між спільнотами, навряд чи утворюють кліки.

Використання методу клікових перколяцій ґрунтується на припущенні, що якщо кліка може «переміщатися» в графі, то вона опиниться в пастці всередині спільноти і не зможе пройти між двома спільнотами через брак сполучних шляхів. У цьому методі одна кліка може бути «переміщена» в іншу, якщо вони мають спільні вершини, крім однієї, а спільнота визначається як максимальний зв'язний підграф вихідного графа так, що кожна вершина цього графа належить деякій *k*-тій кліці, яка повністю лежить у цьому підграфі.

Алгоритм працює наступним чином:

1) Знайти всі кліки розміру *k* в графі;
2) Створити граф, де вузли є кліками розміру *k*;
3) Додати ребра, якщо два нових вузли(кліки) мають *k*-1 спільних вузлів;
4) кожен зв'язаний компонент є спільнотою.

Візуалізація 3-клікової перколяції наведена на рис. 2.5, де видно, що початковий граф має наступні кліки: {1, 2, 3} {1, 3, 4} {4, 5, 6} {5, 6, 7} {5, 6, 8} {5, 7, 8} {6, 7, 8}. Кожна кліка представлятиме вузол у графі клік, і ці вузли з’єднані один з одним, якщо вони мають спільні *k*-1 (у цьому випадку 2) вузли (a). У результаті граф кліків представляє два зв’язані компоненти, які містять {1,2,3,4} та {4,5,6,7,8} (b). Вузол 4 належить обом спільнотам. Тобто, граф містить дві співтовариства, що перекриваються.

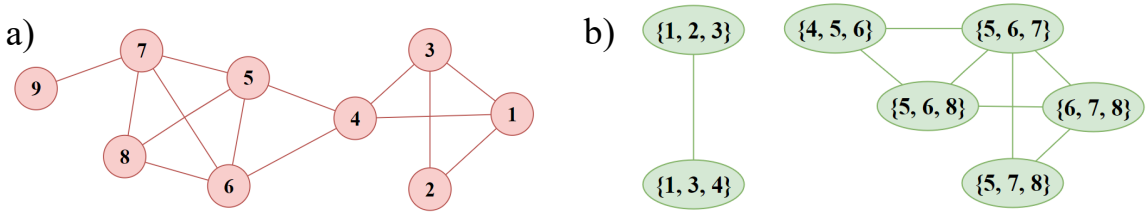


Рисунок 2.5. Візуалізація 3-клікової перколяції

Описані методи Гірвана–Ньюмена та клікової перколяції є особливо корисними у біологічних, соціальних та комп'ютерних мережах, де структура спільноти має важливе значення для аналізу мережевих взаємодій, а сама мережа не є надто великою, адже ці методи є обчислювально затратними. В рамках даної роботи вирішено застосувати інструменти стійкої гомології, оскільки вони надають змогу зрозуміти принципи побудови симплікаційних комплексів та візуалізувати граф для різних варіацій параметрів.

3. Розрахунковий експеримент

3.1. Дизайн дослідження. Дані та підготовка

Набір даних для розрахункового експерименту було отримано з дослідження National Institute on Drug Abuse CENIC-P1S1 щодо нікотинових курців [11]. Це подвійно сліпе, паралельне, рандомізоване клінічне випробування, проведене з червня 2013 року по липень 2014 року в десяти місцях. Умови участі включали вік від 18 років, щоденне куріння п'яти або більше сигарет і зацікавленість у відмові від куріння. Учасникам випадковим чином було запропоновано курити протягом шести тижнів або їхні звичайні, або один з шести типів дослідних сигарет, наданих безкоштовно.

Первинним результатом була кількість викурених за день сигарет (далі – CPD) протягом 6-го тижня. Метою дослідження було оцінити зв'язок між виходом нікотину в сигаретах з дуже низьким вмістом нікотину, викурених за день, впливом нікотину, дискомфортом/дисфункцією та іншими поведінковими проявами, пов'язаними зі здоров'ям, нікотиновою/тютюновою залежністю, біомаркерами впливу тютюну, наміром кинути курити, компенсаторним курінням, вживанням інших видів тютюну, характеристиками сигарет, когнітивною функцією, функціями серця і судинної системи, а також оціненим ризиком.

У дослідженні протягом 6 тижнів приймали участь 839 учасників, якому передував збір базових даних, а завершилося воно подальшим додатковим дослідженням.

На рис. 3.1 показана спрощена схема дизайну дослідження.

Результати оригінального дослідження [12] показали, що протягом 6-го тижня середня кількість викурених за день сигарет була (достовірно) меншою в учасників, випадково розподілених на сигарети з вмістом нікотину 2,4, 1,3 або 0,4 мг/г тютюну, ніж в учасників, випадково розподілених на сигарети їхньої звичайної марки або на сигарети з вмістом нікотину 15,8 /г тютюну.

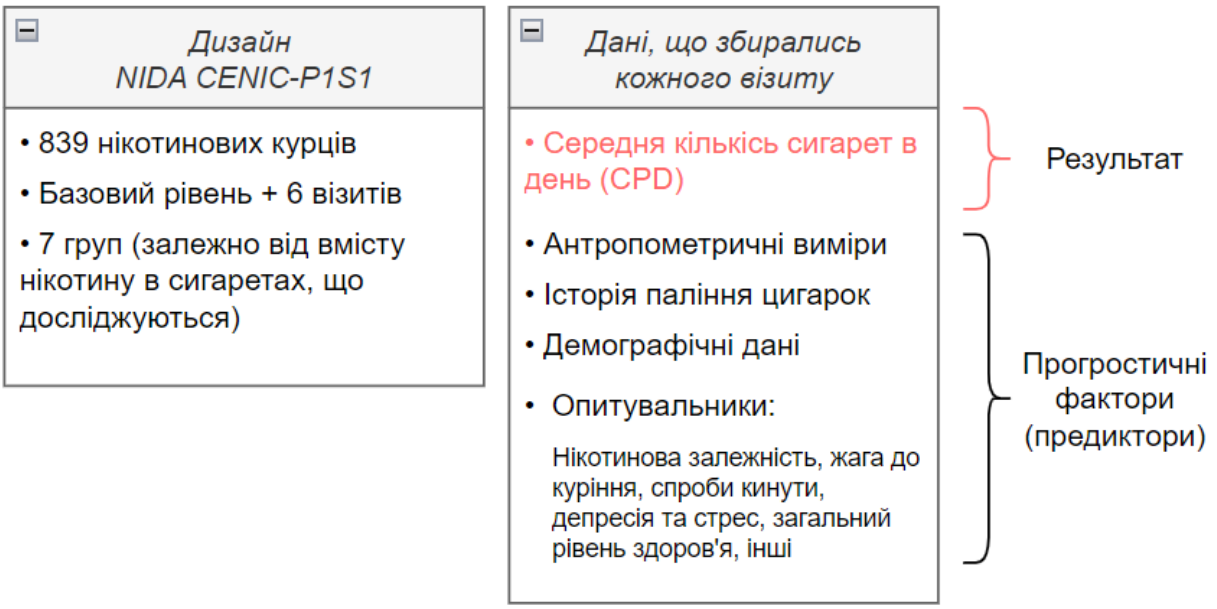


Рисунок 3.1 Схема дизайну дослідження для розрахункового ескперименту

3.1.1. Попередня підготовка даних

Файли отримані у вигляді 6 наборів даних [11], що містять в собі текстові дані, розділені комою, що вимагають їх конвертації.

Дані містять в собі пусті значення і значення, які виглядають неправдоподібно (CDP > 1000), тому суб'єкти з такими записами, для щонайменше хоча б одного візиту, були вилучені з набору даних для аналізу.

В якості кінцевого показника обрано CDP, що оцінювався на кожному із семи візитів.

Таким чином після попередньої обробки оригінальних даних в набір даних для розрахункового експерименту було включено 767 суб'єкта.

Попередню підготовку було виконано за допомогою мови програмування SAS. Код програми попередньої підготовки даних наведено у Додатку 1.

3.1.2. Підготовка до ТАД

Приймаючи до уваги кількість візитів та обраний кінцевий показник, для кожного суб'єкта було визначено 7-вимірний вектор, що характеризує поведінку суб'єкта щодо куріння:

$$v = (CPD_0, CPD_1, \dots, CPD_6)$$

Щоб акцентувати увагу лише на поведінкових характеристиках суб'єктів і спростити аналіз, різниця між учасниками вимірювалась за допомогою комбінованої метрики, яка складається з кореляційної відстані та абсолютного значення різниці між значеннями CPD:

$$d_{cm}(u, v) = d_c(u, v) + d_m(u, v),$$

де d_c – кореляційна відстань, d_m – абсолютне величина різниці між значеннями CPD. В свою чергу,

$$d_c(u, v) = 1 - \frac{(u - \overline{u}) \cdot (v - \overline{v})}{\|u - \overline{u}\|_2 \|v - \overline{v}\|_2},$$

де $\overline{v}$ – середні значення елементів вектора *v.* $\|\boldsymbol{u} - \overline{\boldsymbol{u}}\|_\mathbf{2}$–стандартна евклідова відстань. Кореляційна відстань коливається від 0 до 2 і вимірює близькість напрямків поведінкових трендів щодо CPD. Вона є чутливою до нахилів часового ряду. Однак вона відцентрова, тому не чутлива до зсувів і не може розрізнити початкові точки трендів (базовий показник CPD), а отже, і середні значення CPD під час візитів. Таким чином, побудувавши хмару точок за отриманою матрицею кореляційних відстаней, отримаємо H_0 комплекс, який народжується та існує майже увесь час фільтрації, оскільки щільність точок буде велика, а їх розподіл буде відцентровим від 0. Останнє пояснює необхідність використання абсолютної величини різниці між значеннями CPD:

$$d_m(u, v) = 2 \left| \frac{(\overline{u} - \overline{v})}{max_{a,i}(a_i)} \right|,$$

де максимальне значення береться по усім координатах a_i усіх векторів a набору даних (максимальне CPD, що спостерігалось).

Рис. 3.2 відображає візуалізацію набору даних, що містить комбіновані метрики використовуючи проекцію багатовимірного масштабування (далі – MDS). Це дозволяє спроектувати багатовимірний набір даних на 2-вимірну площину намагаючись зберегти відстані між точками.

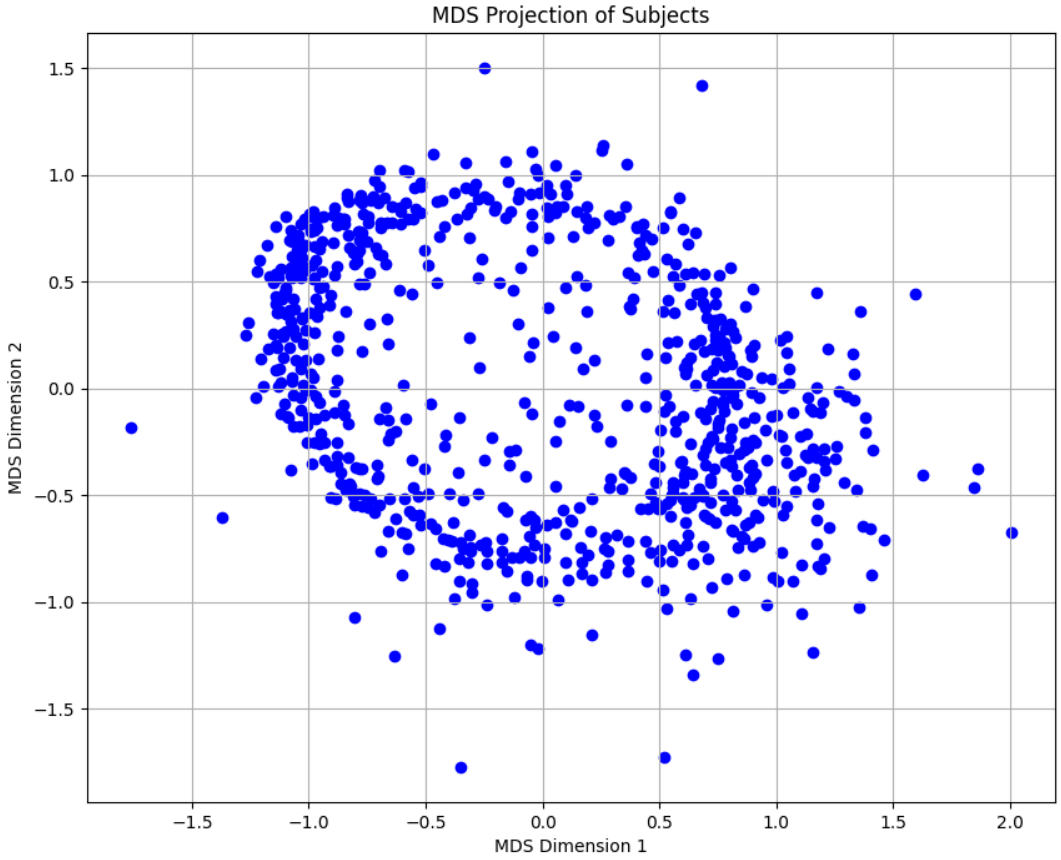


Рисунок 3.2. Результат MDS проектування комбінованих метрик кожного суб’єкта

Розрахувавши зміну CPD для налаштування кольорової шкали від мінімального до максимального значення зміни CPD можна отримати діаграму MDS з шкалою, що показуватиме збільшення чи зменшення CPD для кожного суб’єкта (див. рис. 3.3).

Отримані комбіновані метрики та проекції, що базуються на щільності точок, з урахуванням кінцевих показників, можуть бути використані в побудові топологічної моделі.

Підготовку набору даних для ТАД було виконано за допомогою мови програмування Python. Код програми наведено у Додатку 2.

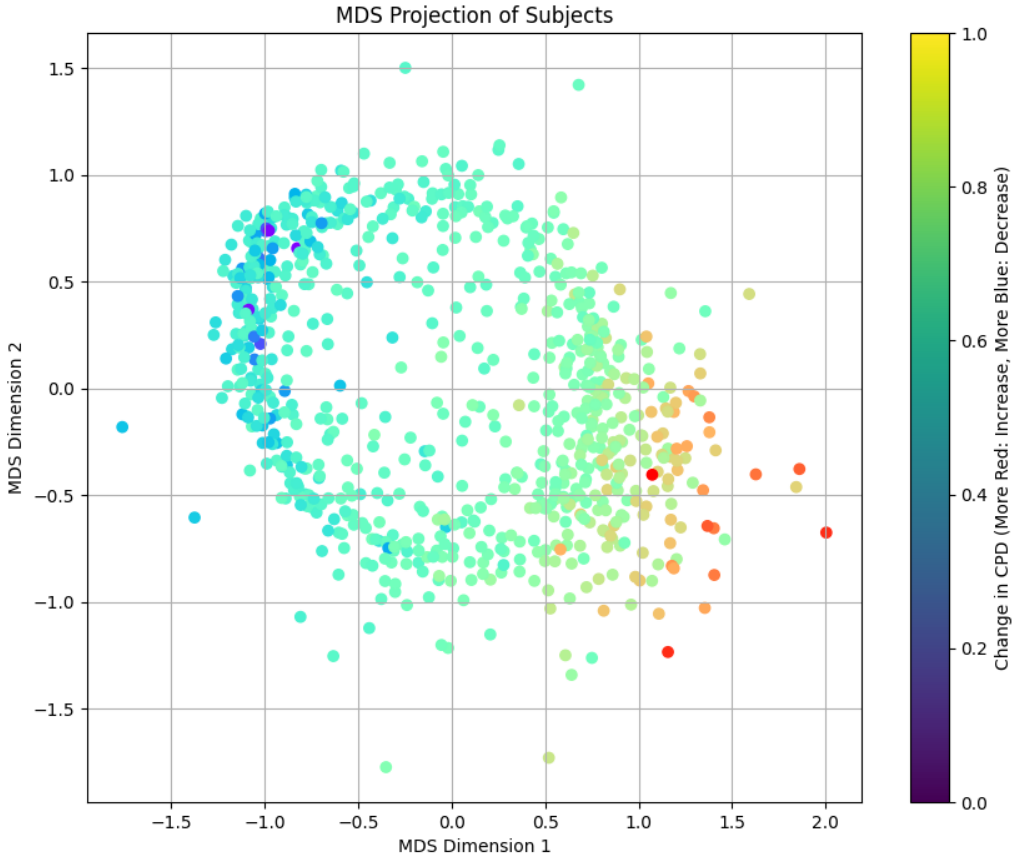


Рисунок 3.3 Результат MDS проектування комбінованих метрик кожного суб’єкта з додаванням параметру зміни CPD за часом

3.2. Аналіз та застосування ТАД

Почнемо дослідження з аналізу стійкої гомології, для чого побудуємо діаграми стійкості для 0-вимірної (H_0) та 1-вимірної (H_1) гомологій. Отримані діаграми показані на рис. 3.4.

H_0 відстежує з'єднані компоненти в наборі даних. Кожна точка (або точка) на діаграмі представляє групу тісно пов’язаних точок даних. Значення на осі абсцис, де з’являється точка, вказує, коли в процесі фільтрації «народжується» новий зв’язаний компонент. Значення осі ординат вказує, коли цей компонент зливається з іншим компонентом, таким чином «вмираючи». Точки вздовж осі ординат, які виділяються від інших довгим існуванням представляють істотні ізольовані кластери або компоненти в даних.

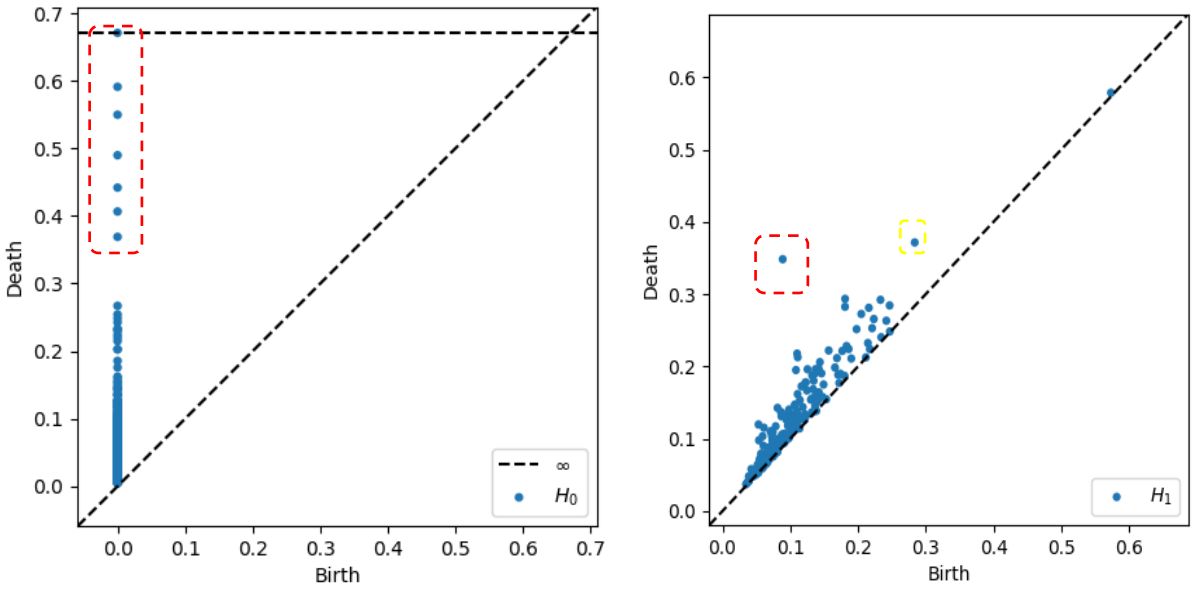


Рисунок 3.4. Зліва – діаграма стійкості для 0-вимірних (H_0) гомології, справа – діаграма стійкості для 1-вимірних (H_1) гомології.

Таким чином, кожну значущу точку на діаграмі H_0 (особливо ті, що знаходяться далеко від осі народження або зберігаються задовго до злиття) можна розглядати як вузол на графіку. Ці вузли можна вважати центральними або ізольованими кластерами в аналізі графа.

H_1 відстежує петлі або діри в даних. Це прогалини, оточені даними, але які самі по собі не містять даних. Ті точки, що знаходяться далеко від діагоналі (зберігаються в більш широкому діапазоні масштабів), є більш значущими, що свідчить про значні отвори або петлі.

Дослідивши особливості стійкої гомології можна побудувати графи для ТАД з урахуванням усіх значущих точок.

Таким чином, маніпулюючи порогом побудови вузлів графа (опираючись на діаграму H_0) та його ребер (опираючись на діаграму H_1) можна прослідкувати за формою графа та виявити пов’язані суб’єкти. Код програми фільтрації, побудови діаграми стійкості гомології та графа для аналізу наведений у Додатку 2. Візуалізація графів з різними пороговими значеннями показані в Додатку 3.

Таким чином аналітик може візуально визначити спільноти взаємопов’язаних суб’єктів, які підлягають подальшому детальному

дослідженню. Щоб виявити особливості окремих спільнот доцільно використати методи кластеризації, які зокрема описані в Розділі 2.1, для кожної спільноти окремо. Для виділення спільнот з щільного графу, альтернативою дослідженню діаграм стійкості гомології та підбору параметрів фільтрації для дослідження стійкої гомології, можуть слугувати, зокрема методи ТАД описані в Розділах 2.2.2 та 2.2.3, проте ці методи досить затратні в обчислювальній потужності.

Як кластеризація, так і виявлення спільнот в графі потребують методів оцінки точності їх результатів. Огляд методів оцінювання можна знайти в [5] для кластеризації та в [13, 14] для алгоритмів виявлення спільнот.

3.3. Результати

Для порогового значення за H_0: 0,055 та H_1: 0,132 візуалізація графу показана на рис. 3.5.

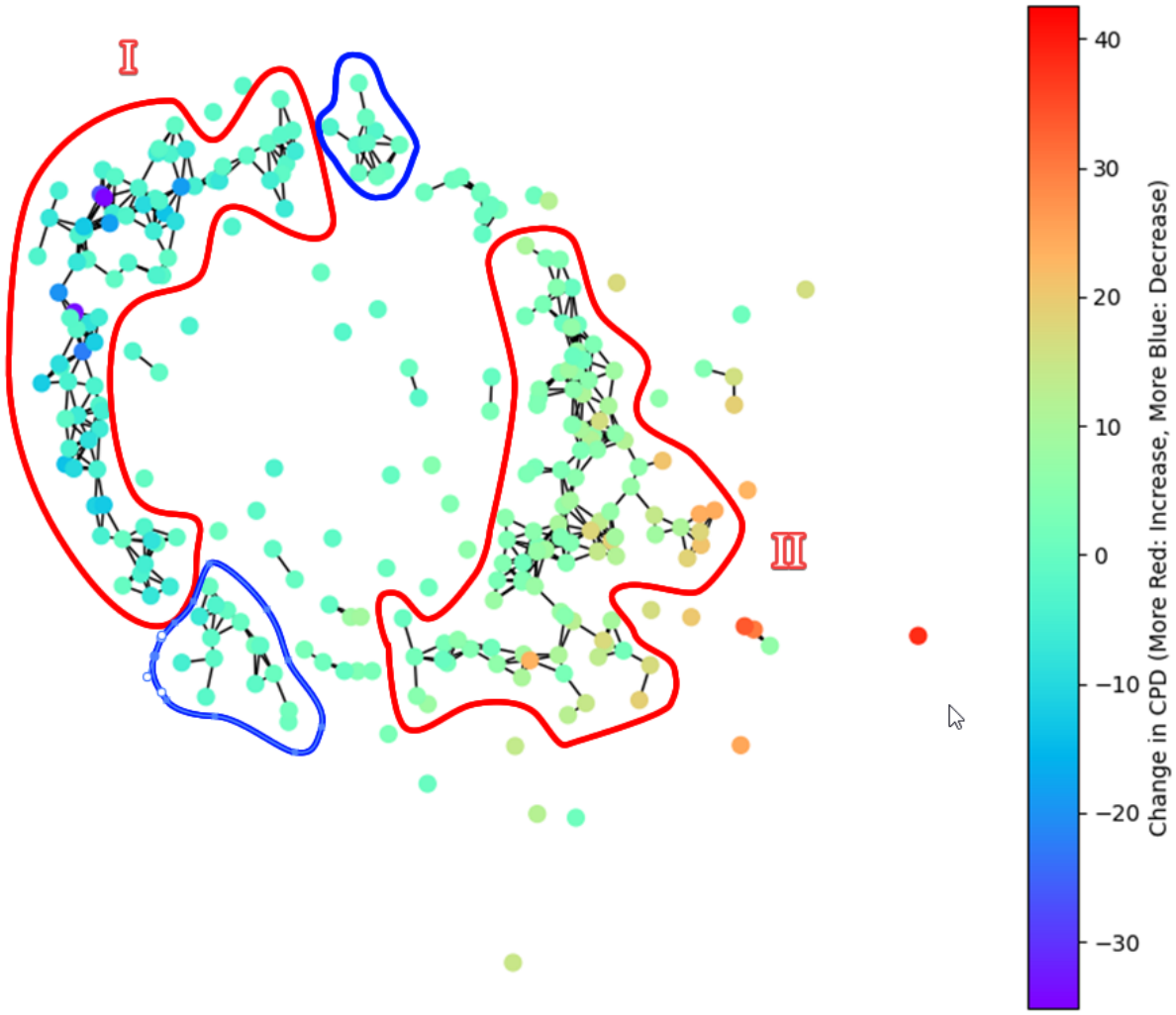


Рисунок 3.5. Граф обраний за допомогою ТАД розрахункового експерименту

На візуалізації графа чітко виділяються дві великі спільноти суб’єктів, що мають множинні зв’язки (червоним), а також декілька менших ізольованих спільнот (синім). З точки зору мети дослідження варто їх розглянути окремо. Наприклад, видно, що для менших синіх спільнот не має тенденції до збільшення чи зменшення CPD. Таким чином слід перевірити, чи входять представники цієї спільноти до групи, які не змінювали дозу нікотину в дослідженні, а курили свої звичні цигарки. У випадку, якщо в спільноту попадають представники різних підгруп, слід виявити інші спільні предиктори, наприклад шляхом кластерного аналізу підгрупи. З огляду на низку предикторів (28) в дизайні дослідження аналогічним чином варто проаналізувати інші виявлені спільноти, що схильні до збільшення CPD (червоні II) та зменшення CPD (червоні I) та перевірити чи дійсно основним фактором для зміни CPD є саме концентрація нікотину в сигаретах, як виявлено в результатах оригінального дослідження.

За результатами додаткового дослідження, було виявлено, що червоні спільноти мають достовірну різницю (*xi*-квадрат тест, p - значення $\leq 0{,}05$) у розподілі за групами лікування порівняно з рештою набору даних. Так, наприклад I-а червона спільнота складається здебільшого з учасників, які зменшили споживання CPD згідно з рис. 3.5 і згідно з рисунком 3.6 їм було призначено сигарети з меншим вмістом нікотину, що підтверджує результати оригінального дослідження. У II-й червоній спільноті спостерігається значний зсув до більш високого вмісту нікотину (рис. 3.7) разом зі збільшенням CPD.

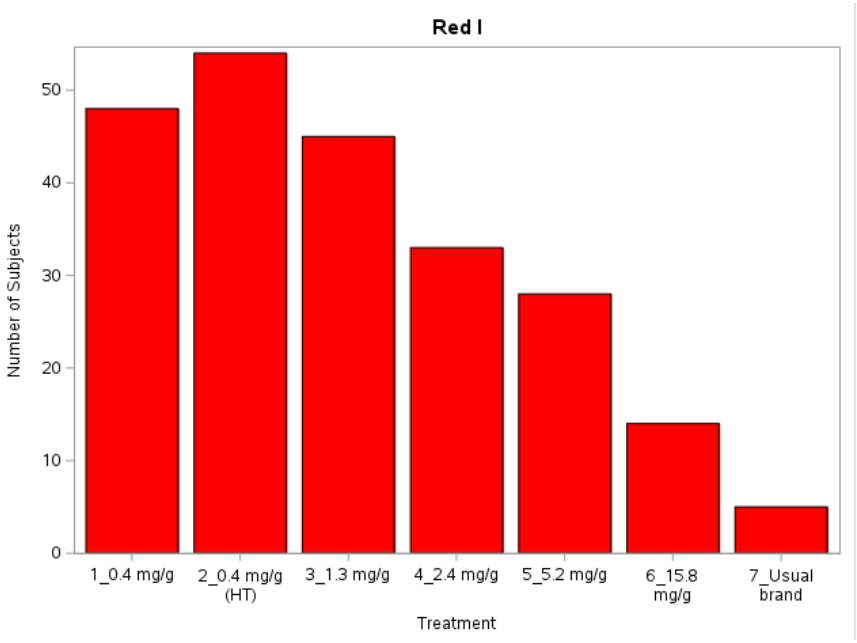


Рисунок 3.6 Розподіл пацієнтів за вмістом нікотину в сигаретах для I червоної спільноти.

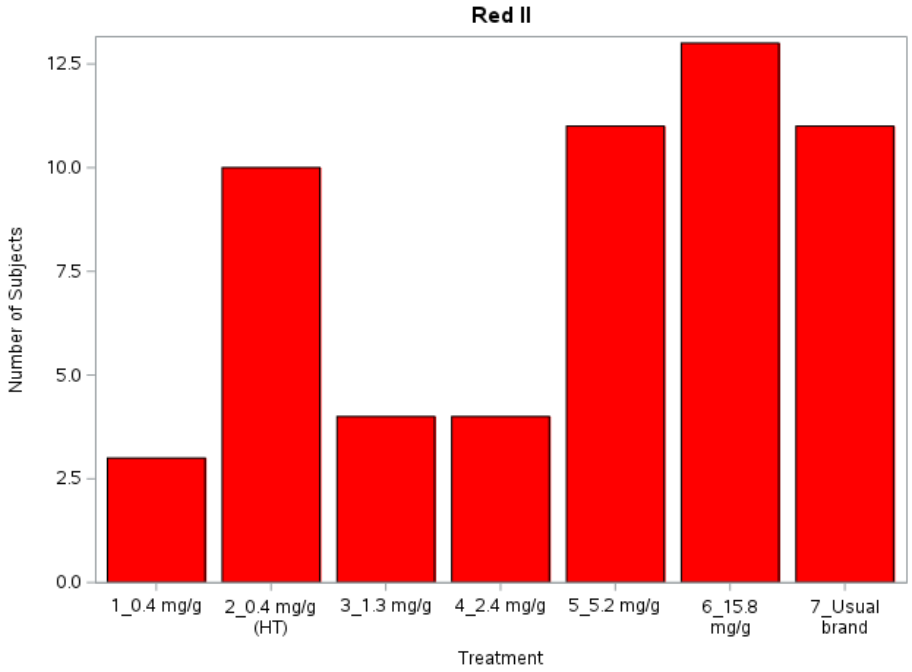


Рисунок 3.7 Розподіл пацієнтів за вмістом нікотину в сигаретах для II червоної спільноти.

4. Висновки

Як і кластеризація, визначення спільнот також є досить нечіткою проблемою, адже чітке визначення того, що становить кластер чи спільноту, відсутнє. Визначення спільнот може розглядатися як спеціальний випадок кластеризації, де групування здійснюється за даними, що пов’язані певними відносинами. Для таких даних використовуються графи, де спільноти представлені як групи вузлів з сильними внутрішніми та слабкими зовнішніми зв'язками.

Як кластеризація, так і виявлення спільнот допомагають ідентифікувати в даних групи зі спільними характеристиками чи ролями. Хоча алгоритми кластеризації базуються на метричних властивостях даних, виявлення спільнот включає використання топологічних властивостей графа, що дозволяє виявити не лише взаємозв'язки між кластерами, але й визначити роль конкретної точки усередині кластера. Так, вузли з центральним положенням у спільноті мають багато зв'язків із іншими членами спільноти, підкреслюючи їх важливу роль у стабільності спільноти, тоді як вузли на межах між спільнотами можуть відігравати ключову роль у посередництві та обміні між різними групами.

В даній роботі ТАД досліджувався шляхом застосування методу стійкої гомології для реальних даних клінічного дослідження. Виявлено декілька спільнот, які підлягають подальшому більш ізольованому та детальному дослідженню.

Слід зазначити, що в ході даної роботи не були використані індикатори для оцінки якості алгоритмів з виявлення спільнот, оскільки оптимальні параметри побудови графу обирались експериментально на основі діаграм стійкості гомології.

В подальших дослідженнях слід використати повний спектр можливостей неконтрольованого навчання із застосуванням інших сучасних алгоритмів пошуку спільнот, та порівняти їх ефективність і якість за наявними індикаторами.

ТАД широко і успішно використовується в біології, проте не часто в клінічних дослідженнях. Проте, існують успішні і багатообіцяючі приклади використання ТАД в медицині, наприклад за допомогою побудови графа було виявлено генотип раку, який повністю піддається лікуванню і його спорідненість з іншими небезпечними типами раку (див. рис. 4.1 [15]).

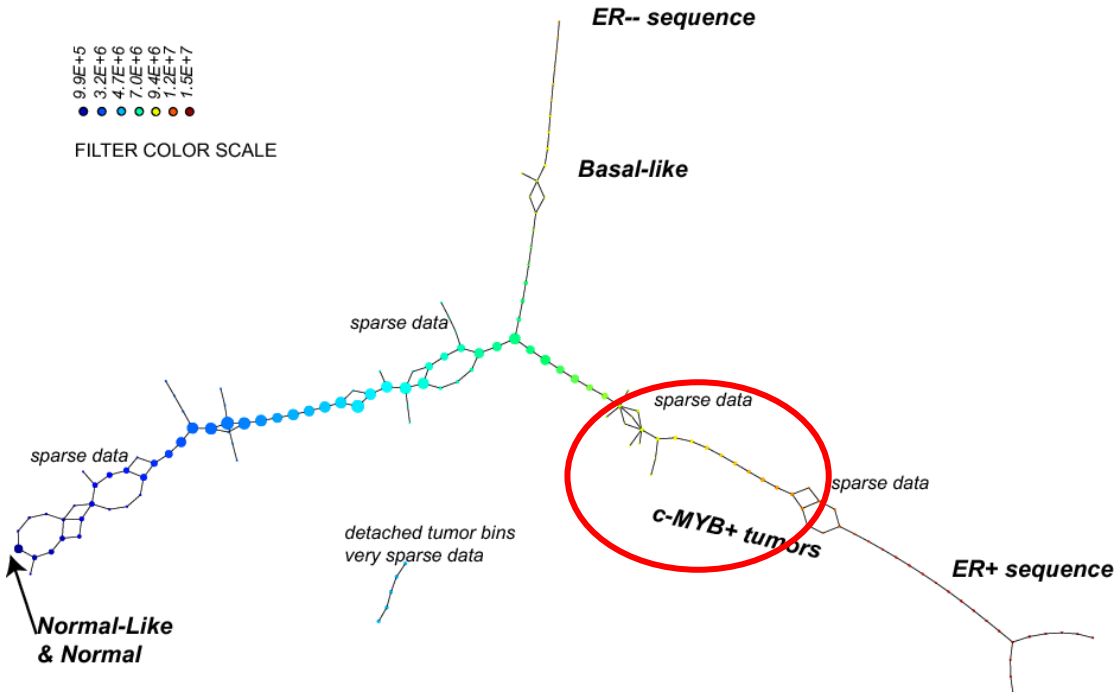


Рисунок 4.1. Презентація результатів дослідження у вигляді графу [15].

В той час, коли класичний ієрархічний кластерний аналіз не показав вагомих результатів у цьому напрямку (див. рис. 4.2).

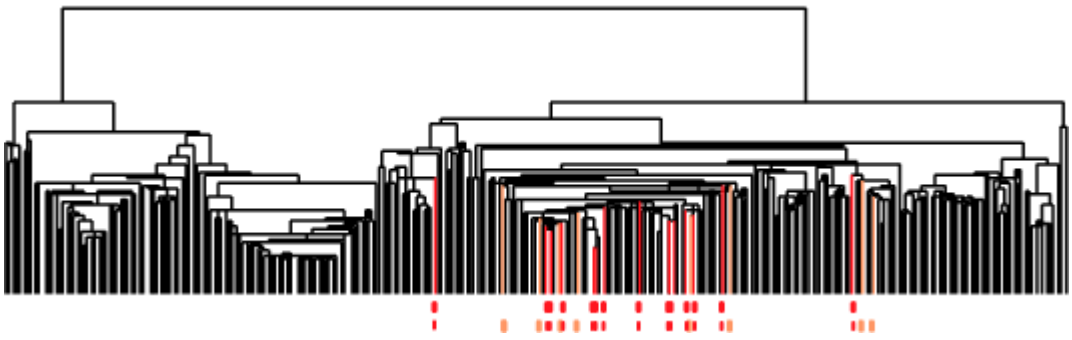


Рисунок 4.2. Презентація результатів дослідження дендрограмою [15]

Також слід прийняти до уваги те, що ТАД є мало чутливим до шумів та пропусків в спостереженнях, оскільки форма графа зберігається (див. рис. 4.3, 4.4), а шуми добре виявляються та фільтруються (див. рис. 4.5). Програмний код прикладів на рисунках наведено у Додату 4.

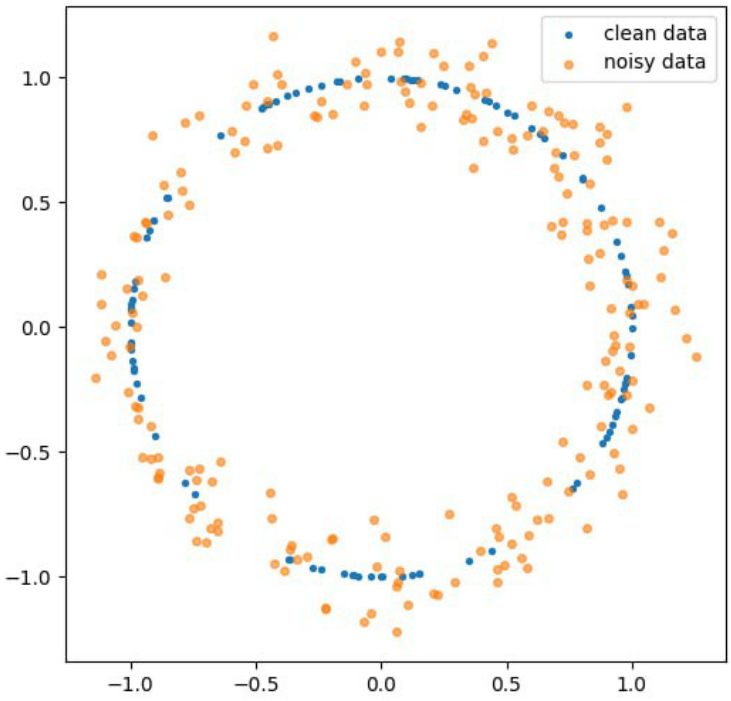


Рисунок 4.3. Діаграма чіткої множини точок (сині) та розсіяної множини точок (жовті) у формі кола.

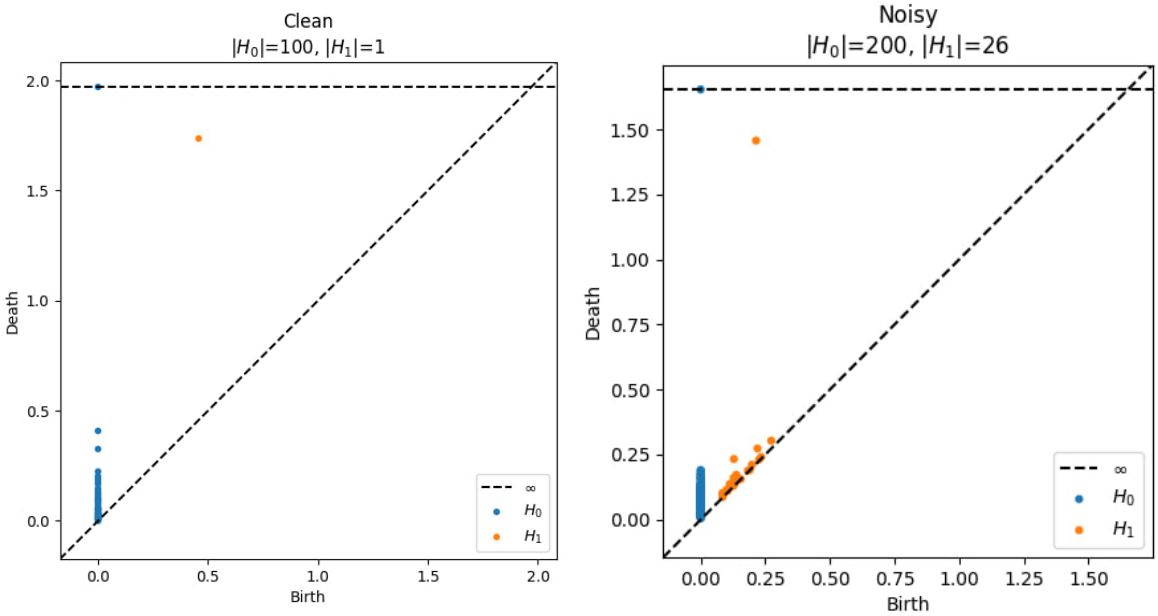


Рисунок 4.4. Діаграма стійкості для 0-вимірної (H_0) гомології (зліва) та діаграма стійкості для 1-вимірної (H_1) гомології (справа).

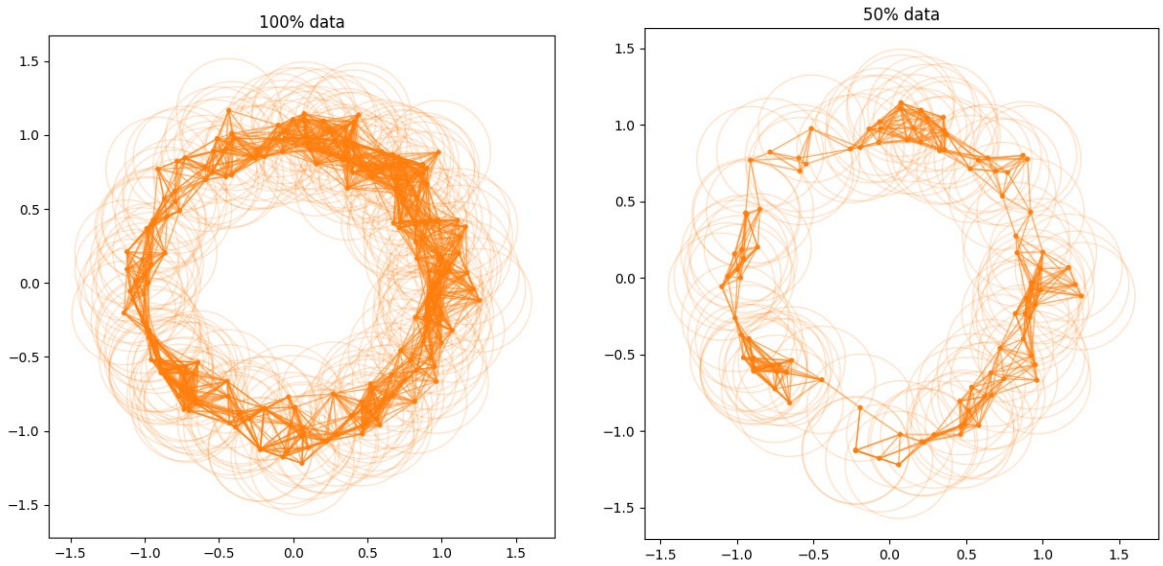


Рисунок 4.5. Комплекси В’єторіса-Ріпса для повного набору даних (зліва) та для набору, з якого видалено половину даних (справа).

Таким чином, існує актуальна доцільність впровадження ТАД як повсякденного інструменту для, принаймні, розвідувального аналізу у клінічних дослідженнях, оскільки навіть найбільш базові його методи застосовані в обчислювальному експерименті показали хорошу схожість результатів з оригінальним дослідженням.

Список використаних джерел

1. M. Bland. *An Introduction to Medical Statistics*. Oxford University Press. 2015.

2. T. Brody. *Clinical Trials: Study Design, Endpoints and Biomarkers, Drug Safety, and FDA and ICH Guidelines.* 2015.

3. FDA-2020-D-0957. Guidance for Industry and FDA Staff. *Statistical Guidance on Reporting Results from Studies Evaluating Diagnostic Tests*. U.S. Department of Health and Human Services Food and Drug Administration. 2007.

4. ICH harmonised tripartite guideline. E9. *Statistical principles for clinical trials.* International conference on harmonisation of technical requirements for registration of pharmaceuticals for human use. 1998.

5. D. Xu, Y. Tian. *A comprehensive survey of clustering algorithms*. Annals of Data Science. 2015. V.2. P. 165-193.

6. F. Provost, T. Fawcett. *Data Science for Business.* O'Reilly Media, Inc. Sebastopol. 2013.

7. X. Deng. *A novel fast classification filtering algorithm for LiDAR point clouds based on small grid density clustering*. Geodesy and Geodynamics. 2022. V.13. P.38-49.

8. Herbert Edelsbrunner, John Harer. *Computational Topology: An Introduction.* Duke University. 2010.

9. Elizabeth Munch. *A User's Guide to Topological Data Analysis*. Journal of Learning Analytics.2017. V.4(2). P. 47-61.

10. M. Girvan, M. Newman. *Community structure in social and biological networks.* Proceedings of the National Academy of Sciences. 2002. V.99, N.12, P. 7821-7826.

11. G. Palla, I. Derenyi, I. Farkas, T. Vicsek. *Uncovering the overlapping community structure of complex networks in nature and society.* Nature. 2005. V.435, P. 814-818.

12. *National Institute on Drug Abuse*. Online ресурс. Доступно за посиланням: https://datashare.nida.nih.gov/study/nidacenicp1s1.

13. E. Donny, R. Denlinger and et al. *Randomized Trial of Reduced-Nicotine Standards for Cigarettes.* The New England Journal of Medicine. 2015. V. 373, N.14, P.1340-1349.

14. S. Fortunato. *Community detection in graphs*. Physics Reports. 2010. V.486. N.3-5. P.75-174.

15. S. Fortunato, D. Hric. *Community detection in networks: a user guide*. Physics Reports. 2016. V.659., P1-44.

16. M. Nicolau, A. Levine, G. Carlsson. *Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival.* Proceedings of the National Academy of Sciences of the United States of America. 2011.

.

Додаток 1 Програмний код для попередньої підготовки даних

```
options validvarname=v7;
%macro import(in=, out=);
	FILENAME REFFILE "/home/wasp660/sasuser.v94/&in..csv";
	PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=WORK.&out.;
		GETNAMES=YES ; GUESSINGROWS = 1000;
	RUN;

	data &out.; set &out. ;
	run;
%mend;
*** Отримання сирих даних;
%import (in=PRIMARY_BASELINE, out=PBL_raw);
%import (in=PRIMARY_POSTRAND, out=PPBL_raw);
*** Початкова фільтрація некоректних даних (CDP>1000 сигарет в день);
data pbl (keep=CENIC_SUBJECT_ID VISIT BASELINE_CPD);
	set PBL_raw ;
	VISIT=0;
	where BASELINE_CPD<1000;
run;
data ppbl;
	set ppbl_raw;
	keep CENIC_SUBJECT_ID VISIT STUDY_CPD;
	where STUDY_CPD<1000;
run;
*** Комбінування BASELINE та POS-BASELINE візитів;
data combined_cdp (rename=CENIC_SUBJECT_ID=subject);
	set pbl (rename=BASELINE_CPD=CDP) ppbl (rename=STUDY_CPD=CDP);
run;
proc sort data=combined_cdp;
	by subject visit;
run;
*** Транспонування значень CPD з списку в матрицю;
proc transpose data=combined_cdp out=comb_cdp_trans prefix=V;
	by subject; id visit;
	var cdp;
run;
*** Експорт датасету для аналізу та додаткова фільтрація - видалення
суб'єктів з пустими даними;
libname out "/home/wasp660";
data out.cdp_filtered (drop=i _name_);
	set comb_cdp_trans;
	array v {7} V:;
	do i=1 to dim (v);
		if v(i)=. then delete;
	end;
run;
```

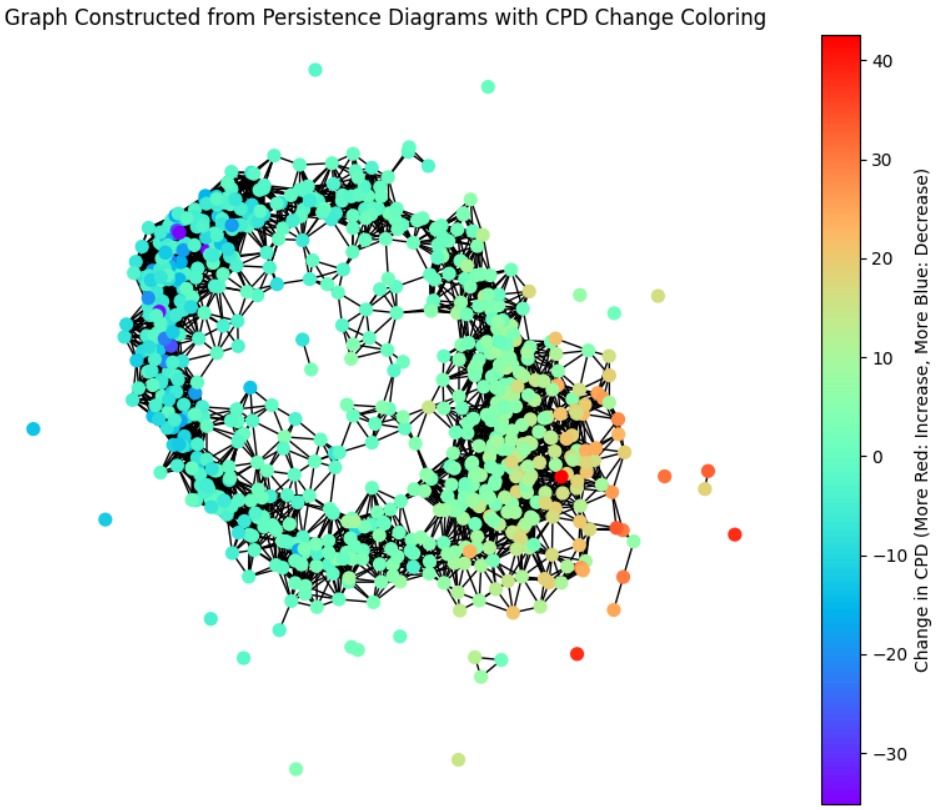
Додаток 2. Програмний код для підготовки даних для аналізу та код ТАД

```
import pandas as pd
import numpy as np
from sklearn.manifold import MDS
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
from ripser import ripser
from persim import plot_diagrams
plt.interactive(False)
import matplotlib as mpl
mpl.rcParams.update(mpl.rcParamsDefault)
import networkx as nx
### Визначення кореляційної відстані
def correlation_distance(u, v):
    mean_u = np.mean(u)
    mean_v = np.mean(v)
    u_centered = u - mean_u
    v_centered = v - mean_v
    numerator = np.dot(u_centered, v_centered)
    denominator = np.linalg.norm(u_centered) * np.linalg.norm(v_centered)
    if denominator == 0:
        return 1  # Return 1 to handle cases where denominator is zero
    correlation = numerator / denominator
    return 1 - correlation

### Визначення відстані середніх значень
def mean_difference_distance(u, v, max_value):
    mean_u = np.mean(u)
    mean_v = np.mean(v)
    mean_diff = np.abs(mean_u - mean_v)
    return 2 * (mean_diff / max_value)

### Обчислення комбінованої метрики
def combined_metric(u, v, max_value):
    dc = correlation_distance(u, v)
    dm = mean_difference_distance(u, v, max_value)
    return dc + dm

### Функція для створення матриці комбінованих метрик
def create_combined_metric_matrix(data):
    num_vectors = data.shape[0]
    max_value = np.max(data.values)  # Максимальне значення CPD
    combined_matrix = np.zeros((num_vectors, num_vectors))

    for i in range(num_vectors):
        for j in range(i + 1, num_vectors):
            combined = combined_metric(data.iloc[i], data.iloc[j], max_value)
            combined_matrix[i, j] = combined
            combined_matrix[j, i] = combined

    return combined_matrix


### Візуалізація за допомогою MDS
def plot_mds(mds_coords, subjects):
    plt.figure(figsize=(10, 8))
    plt.scatter(mds_coords[:, 0], mds_coords[:, 1], c='blue', marker='o')
    plt.title('MDS Projection of Subjects')
    plt.xlabel('MDS Dimension 1')
    plt.ylabel('MDS Dimension 2')
    plt.grid(True)
```

```
    plt.show()

def plot_mds_colored(mds_coords, subjects, cpd_data):
    cpd_change = cpd_data.iloc[:, -1] - cpd_data.iloc[:, 0]
    norm = Normalize(vmin=cpd_change.min(), vmax=cpd_change.max())
    colors = plt.cm.rainbow(norm(cpd_change))

    plt.figure(figsize=(10, 8))
    sc = plt.scatter(mds_coords[:, 0], mds_coords[:, 1], c=colors,
marker='o')
    plt.colorbar(sc, label='Change in CPD (More Red: Increase, More Blue:
Decrease)')
    plt.title('MDS Projection of Subjects')
    plt.xlabel('MDS Dimension 1')
    plt.ylabel('MDS Dimension 2')
    plt.grid(True)
    plt.show()

data  = pd.read_sas(r"C:\Users\sostapchuk\work_laptop\personal\Masters
program\Diploma\Code\CDP_FILTERED.sas7bdat")

### Підготовка даних для аналізу
subjects = data['subject']
cpd_data = data.drop('subject', axis=1)

### Розрахунок комбінованої матриці відстаней
combined_matrix = create_combined_metric_matrix(cpd_data)

### Розрахунок координат MDS
mds = MDS(n_components=2, dissimilarity='precomputed', random_state=42)
mds_coords = mds.fit_transform(combined_matrix)

### Візуалізація результатів за допомогою MDS
plot_mds(mds_coords, subjects)
plot_mds_colored(mds_coords, subjects, cpd_data)

### Розрахунок діаграм стійкості для хмари точок
diagrams = ripser(mds_coords)['dgms']

### Вивід діаграм стійкості
plot_diagrams(diagrams, plot_only=[0], show=True)
plot_diagrams(diagrams, plot_only=[1], show=True)

# Розрахунок зміни CPD
cpd_change = cpd_data.iloc[:, -1] - cpd_data.iloc[:, 0]
norm = Normalize(vmin=cpd_change.min(), vmax=cpd_change.max())
colors_map = plt.cm.rainbow(norm(cpd_change))

### Створення графу
G = nx.Graph()

### Пороги для H0 і H1
h0 = 0.05
h1 = 0.2

### Створення важливих вузлів графу
significant_nodes = [i for i, (birth, death) in enumerate(diagrams[0]) if
death - birth > h0]
for node in significant_nodes:
    G.add_node(node, pos=mds_coords[node], change=cpd_change[node])

### Додавання ребер на основі близькості у просторі MDS
for i in significant_nodes:
    for j in significant_nodes:
```

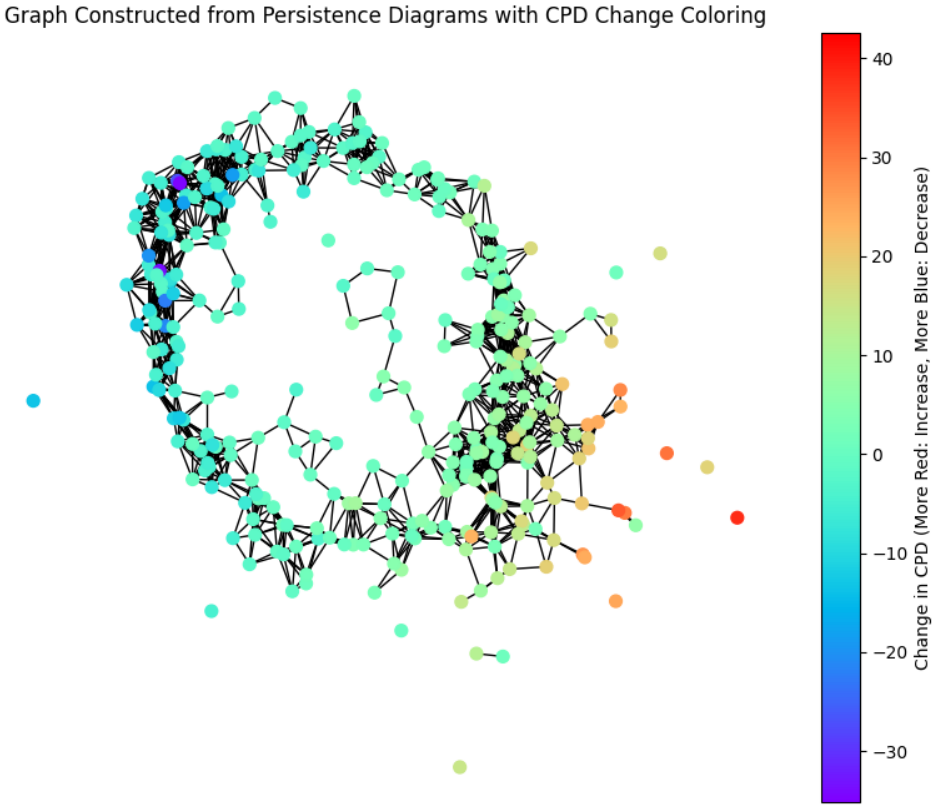
```
        if i != j and np.linalg.norm(mds_coords[i] - mds_coords[j]) < h1:
            G.add_edge(i, j)

### Отримання позицій вузлів для відображення графу
pos = nx.get_node_attributes(G, 'pos')

### Підготовка кольорів вузлів на основі змін CPD
node_colors = [plt.cm.rainbow(norm(G.nodes[node]['change'])) for node in
G.nodes()]

### Малювання графу
fig, ax = plt.subplots(figsize=(10, 8))
nx.draw(G, pos, node_color=node_colors, with_labels=False, node_size=50,
ax=ax)

sm = ScalarMappable(cmap=plt.cm.rainbow, norm=norm)
sm.set_array([])
plt.colorbar(sm, ax=ax, label='Change in CPD (More Red: Increase, More Blue:
Decrease)')

plt.title("Graph Constructed from Persistence Diagrams with CPD Change
Coloring")
plt.show()
```

Додаток 3. Приклади фільтрації для різних порогів стійкості гомології



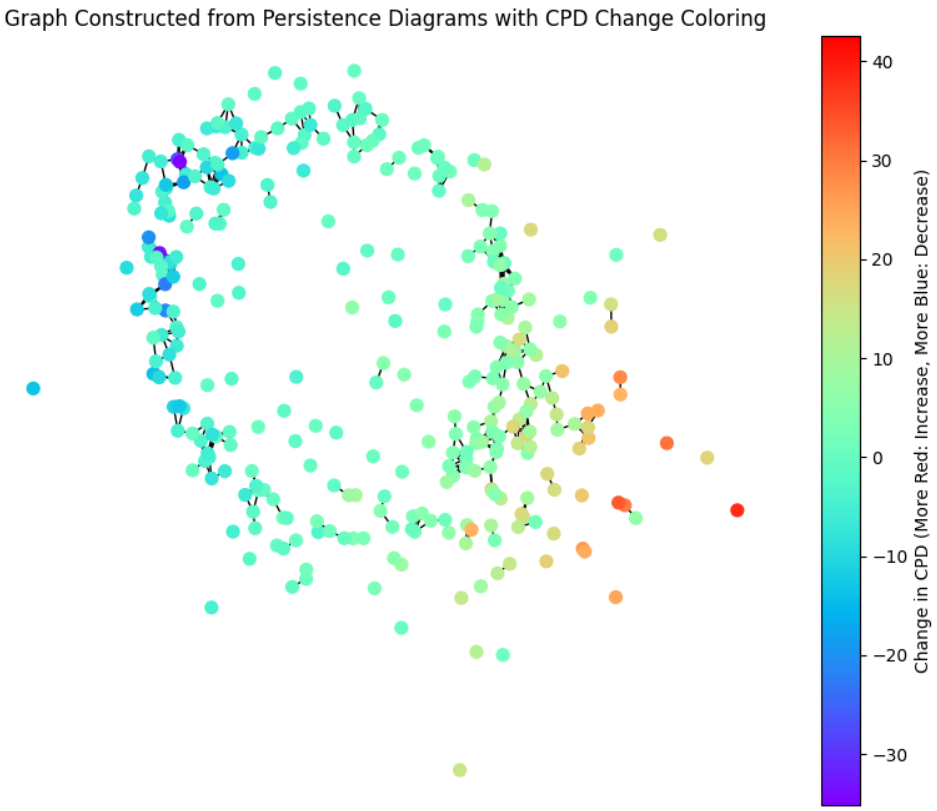

```
persistence_threshold = 0.01  # Threshold for significance in persistence
threshold_distance = 0.2  # Threshold for MDS space distance
```

Рисунок Д3.1



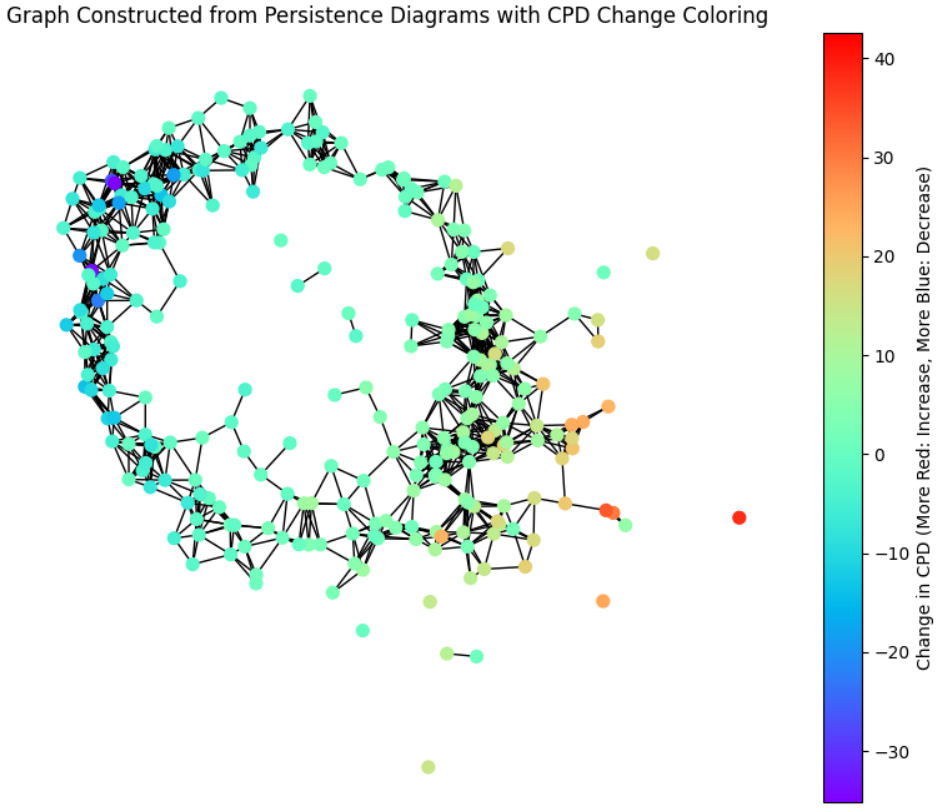

```
persistence_threshold = 0.05  # Threshold for significance in persistence
threshold_distance = 0.2  # Threshold for MDS space distance
```

Рисунок Д3.2



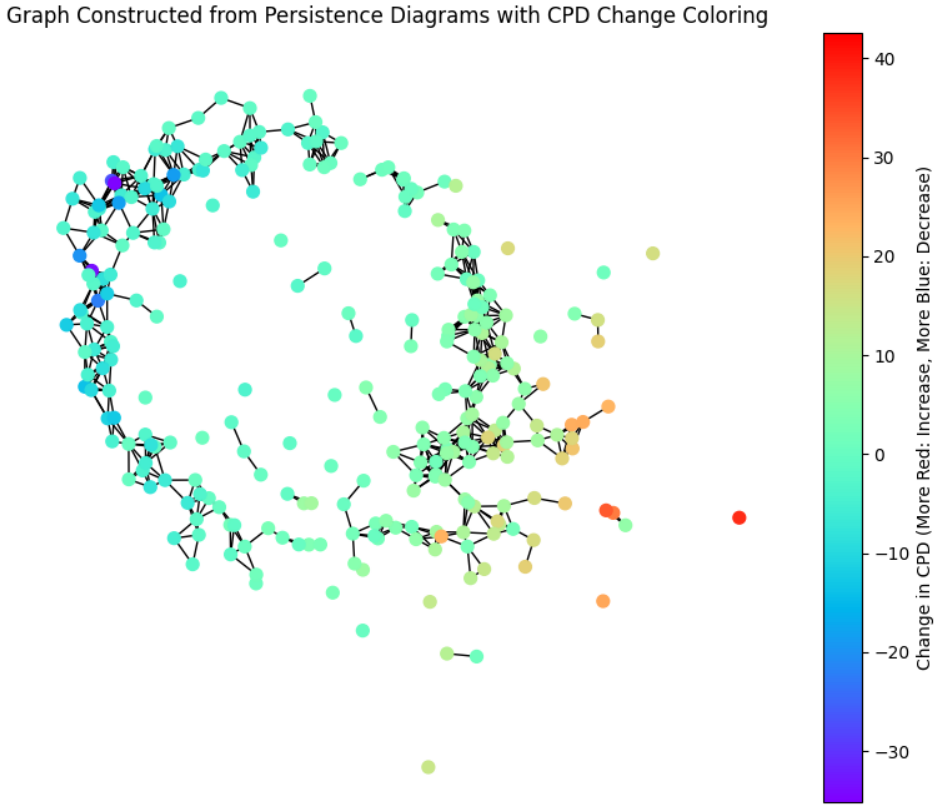

```
persistence_threshold = 0.05  # Threshold for significance in persistence
threshold_distance = 0.1  # Threshold for MDS space distance
```

Рисунок Д3.3



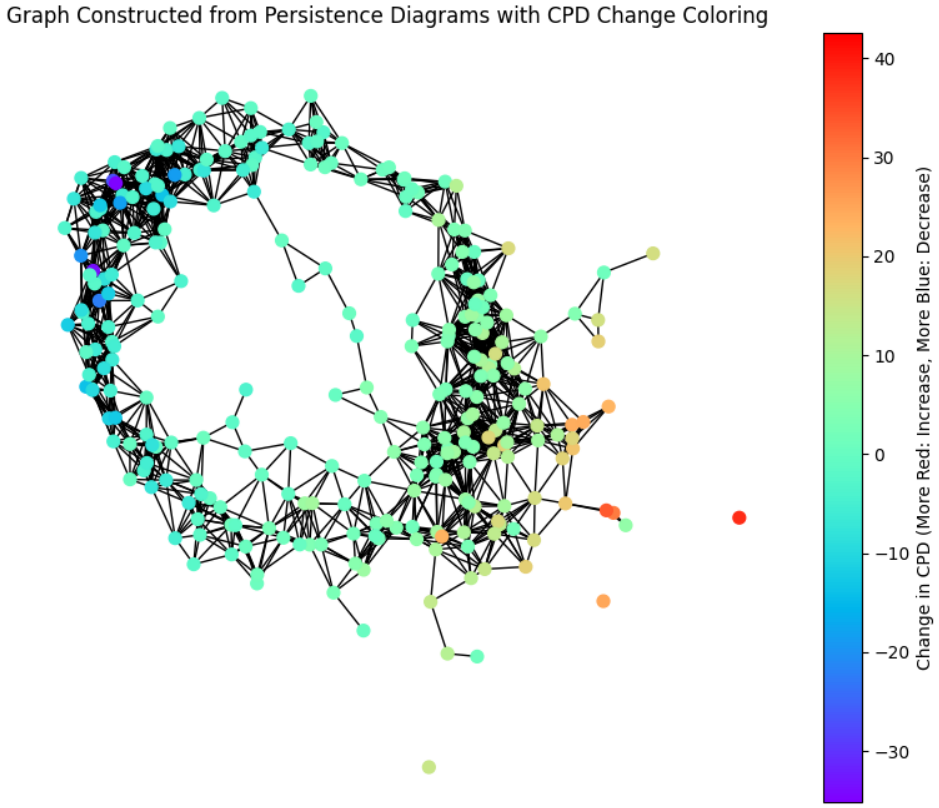

```
persistence_threshold = 0.055  # Threshold for significance in persistence
threshold_distance = 0.2  # Threshold for MDS space distance
```

Рисунок Д3.4




```
persistence_threshold = 0.055  # Threshold for significance in persistence
threshold_distance = 0.15  # Threshold for MDS space distance
```

Рисунок Д3.5




```
persistence_threshold = 0.055  # Threshold for significance in persistence
threshold_distance = 0.25  # Threshold for MDS space distance
```

Рисунок Д3.6

Додаток 4. Програмний код прикладів.

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from matplotlib import cm
import tadasets
import ripser
import persim
from ripser import Rips
plt.interactive(False)
import matplotlib as mpl
mpl.rcParams.update(mpl.rcParamsDefault)
import matplotlib.patches as mpatches
from matplotlib.collections import PatchCollection
np.random.seed(565656)

data_clean = tadasets.dsphere(d=1, n=100, noise=0.0)
data_noisy = tadasets.dsphere(d=1, n=200, noise=0.10)

# data_clean = tadasets.infty_sign(n=100, noise=0.0)
# data_noisy = tadasets.infty_sign(n=100, noise=0.15)

plt.rcParams["figure.figsize"] = (6, 6)
plt.scatter(data_clean[:,0], data_clean[:,1], label="clean data", s=8)
plt.scatter(data_noisy[:,0], data_noisy[:,1], label="noisy data", s=16,
alpha=0.6)
plt.axis('equal')
plt.legend()
plt.show()
def diagram_sizes(dgms):
    return ", ".join([f"|$H_{i}$|={len(d)}" for i, d in enumerate(dgms)])
dgm_clean = ripser.ripser(data_clean)['dgms']
dgm_noisy = ripser.ripser(data_noisy)['dgms']
persim.plot_diagrams(
    dgm_clean,
    show=True,
    title=f"Clean\n{diagram_sizes(dgm_clean)}"
)
persim.plot_diagrams(
    dgm_noisy,
    show=True,
    title=f"Noisy\n{diagram_sizes(dgm_noisy)}"
)

### Data reducing
def plot_rips_complex(data, R, label="data", col=1, maxdim=2):
    tab10 = cm.get_cmap('tab10')

    fig, ax = plt.subplots(figsize=(6, 6))
    ax.set_title(label)
    ax.scatter(
        data[:, 0], data[:, 1], label=label,
        s=8, alpha=0.9, c=np.array(tab10([col] * len(data)))
    )

    for xy in data:
        ax.add_patch(mpatches.Circle(xy, radius=R, fc='none', ec=tab10(col),
alpha=0.2))

    for i, xy in enumerate(data):
        if maxdim >=1:
            for j in range(i + 1, len(data)):
                pq = data[j]
```

```
                if (xy != pq).all() and (np.linalg.norm(xy - pq) <= R):
                    pts = np.array([xy, pq])
                    ax.plot(pts[:, 0], pts[:, 1], color=tab10(col),
alpha=0.6, linewidth=1)
                if maxdim == 2:
                    for k in range(j + 1, len(data)):
                        ab = data[k]
                        if ((ab != pq).all()
                                and (np.linalg.norm(xy - pq) <= R)
                                and (np.linalg.norm(xy - ab) <= R)
                                and (np.linalg.norm(pq - ab) <= R)
                        ):
                            pts = np.array([xy, pq, ab])
                            ax.fill(pts[:, 0], pts[:, 1],
facecolor=tab10(col), alpha=0.1)
                        pass

    plt.axis('equal')
    plt.tight_layout()
    plt.show()
    pass

plot_rips_complex(data_noisy, R=0.35, label="100% data", maxdim=1)

indices = np.random.choice(data_noisy.shape[0], size=data_noisy.shape[0] //
2, replace=False)

# Filter the dataset to remove the selected observations
data_reduced50 = np.delete(data_noisy, indices, axis=0)
plot_rips_complex(data_reduced50, R=0.35, label="50% data", maxdim=1)
```