КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра прикладної статистики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 124 Системний аналіз

на тему:

МЕТОДИ МАШИННОГО НАВЧАННЯ В ЗАДАЧІ СЕГМЕНТАЦІЇ КЛІЄНТІВ

Виконала студентка 4-го курсу
Кондюк Катерина Богданівна

(підпис)

Науковий керівник:
Доцент, кандидат фіз.-мат. наук
Лівінська Ганна Володимирівна

(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри прикладної статистики.
«05» червня 2023 р.,
протокол № 11
Завідувач кафедри
І. В. Розора

(підпис)

КИЇВ - 2023

ЗМІСТ

РЕФЕРАТ

Обсяг роботи 45 сторінок, 39 рисунків, 10 джерел посилань, додаток А.

АГЛОМЕРАТИВНА КЛАСТЕРИЗАЦІЯ, АНАЛІЗ ОСОБИСТОСТІ КЛІЄНТА, ВІЗУАЛІЗАЦІЯ ДАНИХ, КЛАСТЕРИЗАЦІЯ, КЛАСТЕРНИЙ АНАЛІЗ, МАШИННЕ НАВЧАННЯ, МЕТОД ГОЛОВНИХ КОМПОНЕНТ, МЕТОД ЛІКТЯ, СЕГМЕНТАЦІЯ КЛІЄНТІВ.

Об’єктом роботи є клієнти деякого бізнесу, їх особистісні дані та поведінка покупок.

Метою роботи є визначення групи клієнтів з подібними характеристиками і тенденціями за допомогою методів машинного навчання, що базуються на кластеризації.

Методи та інструменти розроблення: метод агломеративної кластеризації, метод головних компонент (PCA), метод ліктя, мова Python для реалізації алгоритмів і візуалізації результатів.

Результати роботи: було визначено 4 групи клієнтів з унікальними характеристиками і тенденціями покупок. Ці групи дозволяють бізнесу більш цілеспрямовано розробляти і просувати свої товари.

Інформація щодо впровадження: результати цієї роботи можуть бути використані в різних відділах бізнесу, включаючи маркетинг, продажі, управління відносинами з клієнтами та стратегічне планування.

Рекомендації щодо використання результатів роботи: результати цієї роботи можуть бути використані для розробки більш цільових маркетингових стратегій, покращення підходу до управління відносинами з клієнтами і для підвищення ефективності просування продуктів.

Сфера застосування: маркетинг, продажі, управління відносинами з клієнтами, стратегічне планування.

ВСТУП

Проєкт зосереджений на аналізі даних за допомогою методів машинного навчання, зокрема кластеризації даних мовою Python. Основна мета мого проєкту — вивчити набір даних «Аналіз особистості клієнта» та отримати уявлення про основні моделі та зв'язки в даних. Використовуючи кластеризацію, ми прагнемо об'єднати покупців у групи на основі їхніх характеристик і виявити будь-які тенденції, які можуть існувати в цих групах.

Машинне навчання — це галузь, яка швидко розвивається, і знайшла застосування в різних сферах, зокрема в охороні здоров'я, фінансах і роздрібній торгівлі. Це передбачає використання алгоритмів для аналізу та вивчення даних, що дозволяє комп'ютерам ідентифікувати закономірності та робити прогнози. Кластеризація є одним із найпоширеніших методів машинного навчання, який використовується для дослідницького аналізу даних. Він передбачає групування точок даних на основі їх подібності з метою виявлення закономірностей і зв'язків, які можуть бути неочевидними на перший погляд.

У нашому проєкті ми будемо використовувати набір даних «Аналіз особистості клієнта», який містить особисті дані про покупців. Використовуючи кластеризацію, ми прагнемо визначити групи клієнтів, які мають подібні характеристики, і дослідити будь-які тенденції, які можуть існувати в цих групах.

Аналіз особистості клієнта – це детальний аналіз ідеальних клієнтів компанії. Це допомагає бізнесу краще розуміти своїх клієнтів і полегшує модифікацію продуктів відповідно до конкретних потреб, поведінки та проблем різних типів клієнтів. Аналіз особистості клієнта допомагає компанії модифікувати свій продукт на основі цільової аудиторії із різних сегментів споживачів. Наприклад, замість того, щоб витрачати гроші на просування нового продукту кожному клієнту в базі даних, компанія може проаналізувати, який сегмент споживачів найімовірніше придбає продукт, а потім продавати продукт лише в цьому конкретному сегменті.

Наш проєкт має кілька ключових цілей. По-перше, ми прагнемо продемонструвати, як методи машинного навчання можуть бути застосовані до реальних наборів даних, і надати інформацію, яка допоможе прийняти рішення. По-друге, ми сподіваємося провести сегментацію клієнтів, щоб допомогти бізнесу оптимізувати та модифікувати продукти відповідно до конкретних потреб і поведінки покупців, що допоможе компанії задовольнити проблеми всіх типів клієнтів. Нарешті, ми сподіваємося розвинути наші навички аналізу даних і машинного навчання, використовуючи Python для впровадження алгоритмів кластеризації та візуалізації результатів.

Щоб досягти цих цілей, ми почнемо з вивчення набору даних і виявлення будь-яких відсутніх значень або викидів, які, можливо, потрібно буде вирішити. Потім ми попередньо обробимо дані, перетворивши категоричні змінні в числові та масштабуючи дані, щоб гарантувати, що всі функції мають однаковий діапазон. Ми будемо використовувати різні алгоритми кластеризації, такі як K-середні та ієрархічна кластеризація, щоб групувати покупців на основі їхніх характеристик. Потім ми проаналізуємо отримані кластери, щоб визначити будь-які базові закономірності та зв’язки, використовуючи методи візуалізації, такі як діаграми розсіювання та теплові карти, щоб допомогти нашому аналізу.

Загалом наш проєкт має на меті забезпечити комплексний аналіз набору даних «Аналіз особистості клієнта» за допомогою методів машинного навчання. Таким чином ми сподіваємося отримати уявлення про різні групи клієнтів конкретного бізнесу, для подальшої оптимізації продуктів під потреби цільової аудиторії.

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ МЕТОДІВ ВИКОРИСТАНИХ В РОБОТІ

1.1 Метод агломеративної ієрархічної кластеризації

Кластеризація або кластерний аналіз - це статистична процедура, яка проявляється шляхом поділу вибраних об'єктів на непересічні підмножини та застосовується в кластерах. Кожен кластер складається з подібних об'єктів, а об'єкти різних кластерів повинні істотно відрізнятися один від одного. Задачу кластеризації можна описати як задачу класифікації. Кластеризації насправді є завданням класифікації, оскільки в усіх випадках ми поділяємо об’єкти на основі їх подібності, але у випадку кластеризації немає умови, що об’єкти належать до якогось класу.

Формально задачу кластеризації можна описати таким чином:

Нехай X — деяка множина об'єктів, Y — множина номерів кластерів. Задано функцію відстані між об'єктами $\rho(x, x')$. Є кінцева вибірка об'єктів $X^m = \{x_1, \dots, x_m\} \subset X$. Потрібно розділити вибірку на непересічні підмножини (кластери), так, щоб кожен кластер складався з об'єктів, близьких по метриці ρ, а об'єкти різних кластерів суттєво відрізнялися. При цьому кожному об'єкту $x_i \in X^m$ приписується номер кластеру y_i.

Алгоритм кластеризації — це функція $a: X \to Y$, яка будь-якому об'єкту ставить у відповідність номер кластера $y \in Y$. Множина Y зазвичай невідома, і ціллю задачі є знайти оптимальне число кластерів, зважаючи на деякий критерій кластеризації, хоча в деяких випадках множина Y може бути задана заздалегідь.

Алгоритми, які використовуються для побудови кластерів, можуть сильно відрізнятися залежно від того, що вони включають у кластер і як їх можна ефективніше шукати. Кластери можуть бути сформовані на основі відстані між ними, щільності ділянок у просторі даних, їх розділення або певного статистичного розподілу. Все залежить від конкретного набору даних і мети використання результатів. Приклад популярних алгоритмів кластеризації:

- K-середніх - це алгоритм розбиття, який мінімізує внутрішньокластерну дисперсію та максимізує міжкластерну дисперсію. K-середніх намагається знайти K центроїдів (K - задана кількість кластерів), а потім призначити кожен об'єкт до кластера з найближчим центроїдом;
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - це алгоритм, який здійснює кластеризацію, базуючись на густині. DBSCAN розглядає кластери як області високої густини, розділені областями низької густини;
- Алгоритм Expectation-Maximization (EM) - це статистичний алгоритм, що використовується для знаходження параметрів статистичних моделей, коли доступна лише неповна інформація. В контексті кластеризації, EM зазвичай використовується з гауссовими моделями змішування;
- Алгоритм Spectral Clustering - цей метод використовує власні вектори графа, що представляє дані, щоб розділити об'єкти на кластери;
- Агломеративний ієрархічний алгоритм - це ієрархічна техніка, яка припускає, що кожен об'єкт в множині даних на початку є окремим кластером, а потім об'єднує кластери за основою найменшої відстані (або найбільшої схожості);
- BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) - це алгоритм, який може ефективно виконувати кластеризацію для великих наборів даних.

У своїй роботі ми використовували метод агломеративної кластеризації, оскільки він створює ієрархічну структуру кластерів, що може бути особливо корисним, коли рівень деталізації важливий. Наприклад, магазин може захотіти подивитись на великі кластери покупців для загальної стратегії маркетингу, а потім зосередитися на менших кластерах для більш тонкої, персоналізованої стратегії.

Також, на відміну від методу k-середніх, агломеративна кластеризація не вимагає випадковості при виборі початкових центрів кластерів, що робить результати детермінованими і повторюваними. Так само важливим є те, що даний метод має змогу працювати з кластерами різної форми, а не лише сферичними.

Агломеративна кластеризація — це тип ієрархічного алгоритму кластеризації. Це техніка неконтрольованого машинного навчання, яка розділяє генеральну сукупність на кілька кластерів таким чином, що точки даних в одному кластері є більш схожими, а точки даних у різних кластерах відрізняються.

Даний метод кластеризації це висхідний підхід, тобто спочатку кожна точка даних є власним кластером, а далі окремі кластери потім об'єднуються за допомогою ітеративного процесу, поки не досягнуть заданого кількості кластерів або не буде створено єдиний кластер, який включає всі об'єкти.

Етапи агломеративного кластеризування:

1. Прийняття всіх точок даних як окремих кластерів;
2. Групування двох найближчі точок (кластерів) в один;
3. Рекурсивне виконання кроку 2, доки не отримаємо потрібну кількість кластерів.

Даний метод дозволяє створити дендрограми – деревоподібні діаграми, якими можна наочно продемонструвати принцип цього алгоритму.

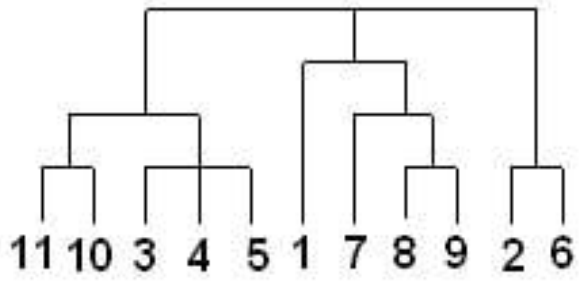

Рисунок 1.1 - приклад дендрограми

1.2 Метод головних компонент (PCA)

Існує багато методів для просторового розподілу вхідних даних, класифікації вхідних значень відповідно до конкретних критеріїв і вибору параметрів з найбільшою кількістю інформації, таких як:

- Факторний аналіз - статистичний метод, який використовується для виявлення структури в сеті кореляцій між змінними. З метою видалення перекривання, він агрегує пов'язані змінні в загальні "фактори";
- Дискримінантний аналіз - метод, який використовується для відділення об'єктів на дві або більше груп на основі множини змінних;
- Методи розпізнавання образів - галузь машинного навчання, що має на меті виявлення та визначення об'єктів та особливостей у даних;
- Метод головних компонент (PCA) - метод зниження розмірності, який перетворює оригінальні змінні на новий набір ортогональних змінних (головні компоненти), які максимізують дисперсію даних.

У даній задачі ми використали метод головних компонент, оскільки робота з таким великим набором даних без зниження розмірності значно ускладнює процес кластеризації, а саме цей метод дозволяє знизити розмірність даних, зберігаючи при цьому якомога більше варіативності інформації.

Також перевагою цього методу є те, що PCA перетворює оригінальні змінні в новий набір ортогональних (некорельованих) змінних. Це є важливою умовою, тому що метод ієрархічної агломеративної кластеризації, який було використано у цій задачі, базується на концепції відстані між об'єктами. Коли змінні корельовані, вони можуть викривлювати ці відстані, так як кореляція означає, що змінні змінюються разом. Це може призвести до того, що кластери будуть формуватися на основі цих кореляцій, а не на основі інших, можливо більш важливих, властивостей даних. Важливо зазначити, що кореляція між змінними часто означає, що існує "надлишкова" інформація в наборі даних. Це приводить до проблеми

високої розмірності, коли багато змінних фактично представляють одну і ту ж інформацію. Метод головних компонент допомагає впоратися з цим, видаляючи кореляції і зменшуючи розмірність простору даних. Крім того, коли змінні не корелюють, результати кластеризації стають легшими для інтерпретації. Кожен кластер може бути описаний набором характеристик, що відрізняються від інших кластерів, а не залежать від кореляцій між змінними.

Алгоритм PCA можна описати наступними кроками:

1. **Стандартизація даних.** Перш ніж застосовувати метод головних компонент, датасет потрібно стандартизувати. Це означає, що кожна змінна (колонка даних) має мати середнє значення, що дорівнює 0, і стандартне відхилення, що дорівнює 1;
2. **Обчислення матриці коваріації.** Матриця коваріації описує ступінь кореляції між парами змінних у датасеті;
3. **Обчислення власних значень і власних векторів матриці коваріації.** Власні значення представляють дисперсію даних у відповідному напрямку, а власні вектори представляють ці напрямки у просторі. Власні вектори, що відповідають найбільшим власним значенням, стають першими головними компонентами;
4. **Формування нового датасету.** На останньому кроці кожний об'єкт (рядок даних) проектується на власні вектори, що формують новий датасет. Цей новий датасет має меншу кількість розмірностей, але втрата інформації мінімізована завдяки вибору головних компонент.

1.3 Метод ліктя

Нарешті перейдемо до одного з основних викликів задач кластеризації – це визначення оптимальної кількості кластерів.

Оптимальна кількість кластерів - це кількість груп, в які дані можна ефективно розділити таким чином, щоб максимізувати внутрішню гомогенність кластерів (тобто схожість елементів у межах одного кластеру) та мінімізувати

міжкластерну гомогенність (тобто схожість між різними кластерами).

Визначення оптимальної кількості кластерів є непростою задачею з декількох причин. Насамперед, кількість кластерів, яка видається оптимальною, може значно варіюватися в залежності від використовуваного методу кластеризації, параметрів цього методу, а також від метрики, яка використовується для оцінки якості кластеризації.

Існує багато методів для визначення оптимальної кількості груп, і кожен з них має свої переваги та недоліки. Рішення про те, який саме метод обрати, буде залежати від самої задачі, початкових даних та самого методу кластеризації, ось найпопулярніші з них:

1. **Метод силуету (Silhouette Method):** Метод силуету використовує силуетний коефіцієнт, щоб визначити ступінь близькості між кластерами. Коефіцієнт силуету варіюється від -1 до 1, де значення, близьке до 1, означає, що зразки добре згруповані, а значення, близьке до -1, означає, що зразки погано згруповані. Найвищий середній коефіцієнт силуету вказує на найкращу кількість кластерів;
2. **Метод відношення відстаней (Gap Statistic Method):** Цей метод порівнює варіацію внутрішнього кластера для різної кількості кластерів з їхніми очікуваннями під нульовою гіпотезою. Ідеальна кількість кластерів - це та, при якій відстань стає більше, ніж можна було б очікувати відповідно до нульової гіпотези;
3. **Метод внутрішнього критерію (Internal Criterion Method):** Цей метод використовує такі критерії, як критерій Калинського-Харабаса і критерій Девіса-Боулдіна. Вони використовують різні метрики (наприклад, відстань між кластерами і внутрішню дисперсію кластера), щоб визначити оптимальну кількість кластерів;
4. **Метод ліктя (Elbow method):** Це евристичний метод, що використовується в кластерному аналізі, щоб визначити оптимальну кількість кластерів для алгоритму.

Метод ліктя базується на візуалізації варіації відстані від кожної точки до її найближчого центроїда. Це зазвичай вимірюється за допомогою суми квадратів відстаней від кожної точки до центроїда її кластера, що також називають як "середній квадрат відхилення від центроїда" або "дисперсія відстані".

При застосуванні методу ліктя, ми спершу виконуємо кластерний аналіз для різної кількості кластерів (наприклад, від 1 до 10) і обчислюємо дисперсію відстані для кожної кількості кластерів. Потім ми візуалізуємо ці дані, зображуючи кількість кластерів на осі x і дисперсію відстані на осі y. На цьому графіку ми шукаємо "лікоть" — місце, де зростання дисперсії відстані раптово знижується. Це вказує на те, що додавання додаткових кластерів не приносить значного зменшення варіативності.

Для даної задачі було обрано саме метод ліктя, оскільки він простий у застосуванні і інтуїтивно зрозумілий, а також не має жодних особливих вимог до вхідних даних. Він не потребує великого обсягу обчислень і є хорошим початковим кроком для визначення кількості кластерів. Проте варто зазначити, що він може не завжди давати чіткий "лікоть" або оптимальне значення може змінюватися в залежності від даних.

РОЗДІЛ 2 ЗАДАЧА КЛАСТЕРИЗАЦІЇ КЛІЄНТІВ НА ОСНОВІ ДАНИХ ПРО ЇХ ОСОБИСТІСТЬ

2.1 Постановка задачі

У нашій задачі ми будемо працювати з набором даних "Аналіз особистості клієнта", який зберігає відомості про особистість покупців. Ця інформація буде використана для групування клієнтів за допомогою кластеризації, в результаті чого ми зможемо виявити різні категорії споживачів і вивчити патерни, які можуть бути притаманні цим групам.

Аналіз клієнтської особистості дає можливість більш глибоко розуміти цільових клієнтів бізнесу. Цей процес дозволяє компаніям адаптувати свої товари та послуги до конкретних вимог, поведінки та викликів, з якими стикаються різні сегменти клієнтів. Детальний аналіз клієнтської особистості допомагає бізнесу налаштувати свій продукт для відповідної цільової аудиторії з різних споживчих сегментів. Таким чином, замість витрати ресурсів на маркетинг нового продукту для всієї бази даних клієнтів, компанія може визначити, який споживчий сегмент найбільш схильний до покупки цього продукту, і зосередити свої зусилля на маркетингу в цьому конкретному сегменті.

Цей датасет містить файл ‘`marketing_campaign.csv`’ з особистими даними про кожного покупця, кількість грошей, що була витрачена на конкретні категорії продуктів, позитивну чи негативну реакцію на кожну з 5-ти рекламних кампаній, що були запущені бізнесом, а також місце, де були зроблені покупки (веб-сайт, каталог чи сам магазин).

2.2 Очищення даних

Першим етапом будь-якого проєкту аналізу даних полягає в дослідженні набору даних і виявленні будь-яких відсутніх значень або викидів. Для цього ми можемо використовувати різні функції та бібліотеки в Python. Даний код завантажить файл "`marketing_campaign.csv`"

```
#Loading the dataset
data =
pd.read_csv("c:/Users/46325/OneDrive/Desktop/diploma/market
ing_campaign.csv", sep="\t")
print("Кількість даних:", len(data))
data.head()

#Information on features
data.info()
```

```
Кількість даних: 2240
```

Рисунок 2.1 - Імпорт даних

Щоб отримати повне уявлення про те, які кроки потрібно зробити для очищення набору даних, давайте подивимося на інформацію в них (рисунок 2.2).

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    2240 non-null   int64
 1   Дата_народження       2240 non-null   int64
 2   Освіта                2240 non-null   object
 3   Сімейний_стан         2240 non-null   object
 4   Дохід                 2216 non-null   float64
 5   Діти                  2240 non-null   int64
 6   Підлітки              2240 non-null   int64
 7   Дата_клієнт           2240 non-null   object
 8   Нещодавність          2240 non-null   int64
 9   Вина                  2240 non-null   int64
 10  Фрукти                2240 non-null   int64
 11  М'ясо                 2240 non-null   int64
 12  Риба                  2240 non-null   int64
 13  Солодощі              2240 non-null   int64
 14  Золото                2240 non-null   int64
 15  Кільк_Покупок         2240 non-null   int64
 16  Кільк_Покупок_Сайт    2240 non-null   int64
 17  Кільк_Покупок_Каталог 2240 non-null   int64
 18  Кільк_Покупок_Магазин 2240 non-null   int64
 19  Кільк_Відв_Сайту_Міс  2240 non-null   int64
 20  Прийн_Камп3           2240 non-null   int64
 21  Прийн_Камп4           2240 non-null   int64
 22  Прийн_Камп5           2240 non-null   int64
 23  Прийн_Камп1           2240 non-null   int64
 24  Прийн_Камп2           2240 non-null   int64
 25  Скарг                 2240 non-null   int64
 26  Відповідь             2240 non-null   int64
dtypes: float64(1), int64(23), object(3)
```

Рисунок 2.1 - інформація по кожній колонці

З наведеного вище результату ми можемо зробити висновок і зауважити, що:

- У колонці «Дохід» є відсутні значення;
- Колонка «Дата_Клієнт», яка вказує дату, коли клієнт приєднався до бази даних, не аналізується як дата-час;
- У нашій структурі даних є деякі якісні характеристики, тож пізніше нам потрібно буде закодувати їх у числові форми.

Перш за все, для відсутніх значень ми просто відкинемо рядки, у яких відсутні значення доходу.

```
#To remove the NA values
data = data.dropna()
```

```
Загальна кількість даних після видалення рядків із відсутніми значеннями: 2216
```

Рисунок 2.2 - кількість даних після видалення

На наступному кроці створимо функцію з колонкою «Дата_Клієнт», яка вказуватиме кількість днів, протягом яких клієнт зареєстрований у базі даних фірми. Однак для простоти візьмемо це значення щодо останнього клієнта в записі. Таким чином, щоб отримати значення, ми повинні перевірити найновішу та найдавнішу записані дати (рисунок 2.4).

```
data["Dt_Customer"] = pd.to_datetime(data["Dt_Customer"],
dayfirst=True)
dates = []
for i in data["Dt_Customer"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("Дата реєстрації останнього клієнта в
записах:",max(dates))
print("Дата реєстрації першого клієнта в
записах:",min(dates))
```

```
Дата реєстрації останнього клієнта в записах: 2014-06-29
Дата реєстрації першого клієнта в записах: 2012-07-30
```

Рисунок 2.3 - найновіший та найстаріший записи

Для того, щоб зрозуміти скільки часу клієнт був покупцем нашого магазину, створимо функцію “Клієнт_протягом”, яка підрахує кількість днів між першою і останньою зафіксованими датами покупки в магазині.

```
#Created a feature "Customer_For"
days = []
d1 = max(dates) #taking it to be the newest customer
for i in dates:
    delta = d1 - i
    days.append(delta)
data["Customer_For"] = days
data["Customer_For"] = pd.to_numeric(data["Customer_For"],
errors="coerce")
```

Для подальшої роботи з датасетом знайдемо кількість унікальних значень номінальних змінних (рисунок 2.5).

```
print("Всього категорій у функції Сімейний стан:\n",
data["Marital_Status"].value_counts(), "\n")
print("Всього категорій у функції Освіта:\n",
data["Education"].value_counts())
```

```
Всього категорій у функції Сімейний стан:
 Сімейний_стан
Одружений      857
Разом          573
Самотній       471
Розлучений     232
Вдова           76
Один             3
Все_складно      2
YOLO             2
Name: count, dtype: int64

Всього категорій у функції Освіта:
 Освіта
Випускаються    1116
Докторант        481
Магістр          365
Бакалавр         200
Базова            54
Name: count, dtype: int64
```

Рисунок 2.4 - унікальні значення в колонках

Далі було виконано додаткові маніпуляції з базою даних, щоб з нею було зручніше працювати. В першу чергу за допомогою колонки «Рік народження» знайдемо поточний вік клієнта:

```
#Age of customer today
data["Age"] = 2023-data["Year_Birth"]
```

Окремо створимо колонку «Всього_витрат», яка вказуватиме загальну суму, яку витратив покупець у різних категоріях товарів.

```
#Total spendings on various items
data["Spent"] = data["MntWines"]+ data["MntFruits"]+
data["MntMeatProducts"]+ data["MntFishProducts"]+
data["MntSweetProducts"]+ data["MntGoldProds"]
```

Також у функції «Сімейний_стан» можемо бачити 8 різних статусів, які замінимо на 2, оскільки це ніяк не повпливає на кінцевий результат, але значно полегшить подальші маніпуляції з датасетом. Проведемо таку заміну:

- “Одружений”, “Разом” на “З партнером”
- “Самотній”, “Розлучений”, “Вдова”, “Один”, “Все складно”, “YOLO” (абревіатура «живемо один раз») на “Одні”

```
#Deriving living situation by marital status
data["Living_With"]=data["Marital_Status"].replace({"Marrie
d"   :"With_Partner", "Together":"With_Partner",
"Absurd":"Alone",   "Widow":"Alone", "YOLO":"Alone",
"Divorced":"Alone",      "Single":"Alone",})
```

Створімо функцію «Діти», де об’єднаємо кількість малечі та підлітків, щоб мати більш ясне уявлення про дітей клієнта, а також одразу визначимо розмір всієї родини у новій, нами створеній функції «Розмір родини».

```
#Feature indicating total children living in the household
data["Children"]=data["Kidhome"]+data["Teenhome"]

#Feature for total members in the householde
data["Family_Size"] = data["Living"].replace({"Alone": 1,
"With_Partner":2})+ data["Children"]

#Feature pertaining parenthood
data["Is_Parent"] = np.where(data.Children> 0, 1, 0)
```

Для колонки «Освіта» зробимо такі ж самі маніпуляції, як і з колонкою «Статус одруження», замінимо п’ять категорій на три:

- “Базова”, “Бакалавр” на “Без вищої освіти” (зазначимо, що в даному випадку «бакалавр» мається на увазі, що людина ще навчається)
- “Випускаються” на “Здобувають вищу освіту”
- “Магістр”, “Докторант” на “Мають вищу освіту”

```
#Segmenting education levels in three groups
data["Education"]=data["Education"].replace({"Basic":"Under
gr   aduate","2n Cycle":"Undergraduate",
"Graduation":"Graduate",      "Master":"Postgraduate",
"PhD":"Postgraduate"})
```

І нарешті відкинемо зайві функції:

```
#Dropping some of the redundant features
to_drop = ["Marital_Status", "Dt_Customer",
"Z_CostContact",   "Z_Revenue", "Year_Birth", "ID"]
data = data.drop(to_drop, axis=1)
```

Тепер наш датасет має такі базові характеристики (рисунок 2.6 - рисунок 2.8):

	Солодощі	Золото	Кільк_Покупок	Кільк_Покупок_Сайт	Кільк_Покупок_Каталог	Кільк_Покупок_Магазин	Кільк_Відв_Сайту_Міс
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	27.028881	43.965253	2.323556	4.085289	2.671029	5.800993	5.319043
std	41.072046	51.815414	1.923716	2.740951	2.926734	3.250785	2.425359
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	9.000000	1.000000	2.000000	0.000000	3.000000	3.000000
50%	8.000000	24.500000	2.000000	4.000000	2.000000	5.000000	6.000000
75%	33.000000	56.000000	3.000000	6.000000	4.000000	8.000000	7.000000
max	262.000000	321.000000	15.000000	27.000000	28.000000	13.000000	20.000000

Рисунок 2.6 - базові характеристики датасету (частина 1)

	Освіта	Дохід	Діти	Підлітки	Нещодавність	Вина	Фрукти	М'ясо	Риба
count	2216	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000
unique	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	Здобувають_вищу_освіту	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1116	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	52247.251354	0.441787	0.505415	49.012635	305.091606	26.356047	166.995939	37.637635
std	NaN	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224.283273	54.752082
min	NaN	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	NaN	35303.000000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000
50%	NaN	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68.000000	12.000000
75%	NaN	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232.250000	50.000000
max	NaN	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000

Рисунок 2.7 - базові характеристики датасету (частина 2)

	Прийн_Кампі3	Прийн_Кампі4	Прийн_Кампі5	Прийн_Кампі1	Прийн_Кампі2	Скарг	Відповіді	Клієнт_протягом	Вік	Всього_витрат	Живуть	Кільк_дітей	Розмір_сім'ї	Є_батьком
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2.216000e+03	2216.000000	2216.000000	2216	2216.000000	2216.000000	2216.000000
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	З партнером	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1430	NaN	NaN	NaN
mean	0.073556	0.074007	0.073105	0.064079	0.013538	0.009477	0.150271	3.054423e+16	54.179603	607.075361	NaN	0.947202	2.592509	0.714350
std	0.261106	0.261842	0.260367	0.244958	0.115588	0.096907	0.357417	1.749036e+16	11.985554	602.900476	NaN	0.749062	0.905722	0.451825
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	27.000000	5.000000	NaN	0.000000	1.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.555200e+16	46.000000	69.000000	NaN	0.000000	2.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.071520e+16	53.000000	396.500000	NaN	1.000000	3.000000	1.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.570560e+16	64.000000	1048.000000	NaN	1.000000	3.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	6.039360e+16	130.000000	2525.000000	NaN	3.000000	5.000000	1.000000

Рисунок 2.8 - базові характеристики датасету (частина 3)

Наведена вище статистика показує деякі розбіжності в середньому в доході та віці та максимальному в доході та віці. Також варто звернути увагу на те, що максимальний вік становить 130 років, оскільки розрахунок проводився на 2023 рік, а дані доволі старі. Щоб поглянути на дані ширше, змоделюємо графіки деяких функцій.

```
#Plotting following features
To_Plot = [ "Income", "Recency", "Customer_For", "Age",
"Spent", "Is_Parent"]
```

```
print("Reletive Plot Of Some Selected Features: A Data
Subset")
sns.pairplot(data[To_Plot], hue= "Is_Parent",palette=
(["#682F2F","#F3AB60"]), size=1.5, aspect=1)
plt.rcParams['figure.figsize'] = [4, 4]
#Taking hue
plt.show()
```

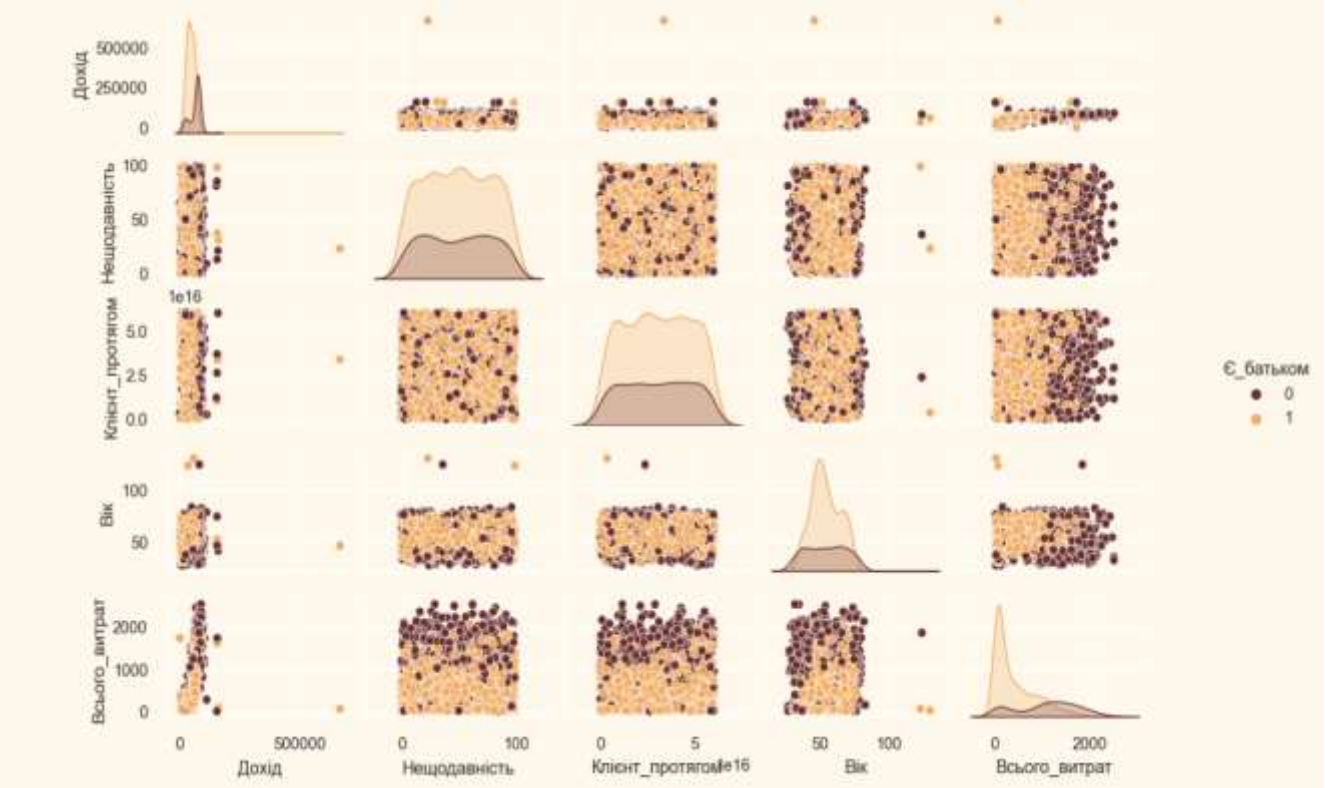

Рисунок 2.9 - парні графіки деяких функцій

З графіків на рисунку 2.9 бачимо, що очевидно є викиди у колонках «Дохід» та «Вік», тому видалимо їх.

```
#Dropping the outliers by setting a cap on Age and Income.
data = data[(data["Age"]<90)]
data = data[(data["Income"]<600000)]
print("The total number of data-points after removing the
outliers are:", len(data))
```

```
Загальна кількість даних після видалення викидів: 2212
```

Рисунок 2.10 - кількість даних після видалення викидів

Тепер спробуємо дослідити дані на наявність кореляції між змінними (не враховуючи номінальні змінні)

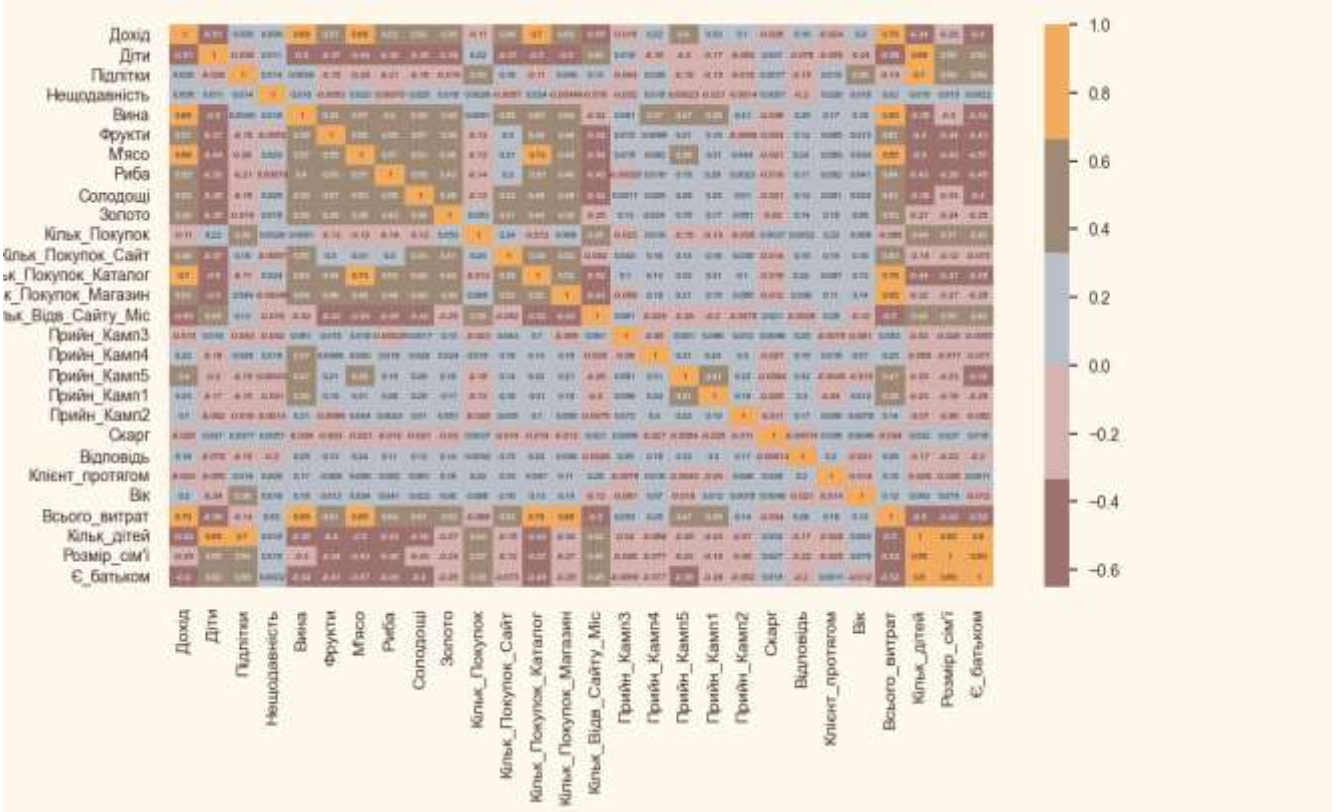

Рисунок 2.11 - матриця кореляції

Дані доволі чисті (рисунок 2.11), тому переходимо до наступного етапу – попередньої підготовки.

2.3 Попередня підготовка

Для попередньої обробки даних застосовуються наступні кроки:

1. Мітка, що кодує категоріальні ознаки

```
#Get list of categorical variables
s = (data.dtypes == 'object')
object_cols = list(s[s].index)

print("Категориальні змінні в наборі даних:", object_cols)
```

```
Категоріальні змінні в наборі даних: ['Освіта', 'Живуть']
```

Рисунок 2.10 - колонки, що містять категоріальні змінні

```
#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
     data[i]=data[[i]].apply(LE.fit_transform)
```

2. Масштабування функцій за допомогою стандартного масштабувальника та створення копії підмножини даних для зменшення розмірності

```
#Creating a copy of data
ds = data.copy()
# creating a subset of dataframe by dropping the features
on   deals accepted and promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
ds = ds.drop(cols_del, axis=1)

#Scaling
scaler = StandardScaler()
scaler.fit(ds)
scaled_ds = pd.DataFrame(scaler.transform(ds),columns=
ds.columns )
print("All features are now scaled")
```

Дата фрейм для подальшого моделювання:

	Освіта	Дохід	Діти	Підлітки	Нещодавність	Вина	Фрукти
0	-0.411675	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041
1	-0.411675	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461
2	-0.411675	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540
3	-0.411675	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961
4	1.123949	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540

	М'ясо	Риба	Солодощі	Золото	Кільк_Покупок	Кільк_Покупок_Сайт
0	1.690293	2.453472	1.483713	0.852576	0.351030	1.426865
1	-0.718230	-0.651004	-0.634019	-0.733642	-0.168701	-1.126420
2	-0.178542	1.339513	-0.147184	-0.037254	-0.688432	1.426865
3	-0.655787	-0.504911	-0.585335	-0.752987	-0.168701	-0.761665
4	-0.218684	0.152508	-0.001133	-0.559545	1.390492	0.332600

	Кільк_Покупок_Каталог	Кільк_Покупок_Магазин	Кільк_Відв_Сайту_Міс
0	2.503607	-0.555814	0.692181 \
1	-0.571340	-1.171160	-0.132545
2	-0.229679	1.290224	-0.544908
3	-0.913000	-0.555814	0.279818
4	0.111982	0.059532	-0.132545

	Клієнт_протягом	Вік	Всього_витрат	Живуть	Кільк_дітей
0	1.527721	1.018352	1.676245	1.349603	-1.264598 \
1	-1.189011	1.274785	-0.963297	1.349603	1.404572
2	-0.206048	0.334530	0.280110	-0.740959	-1.264598
3	-1.060584	-1.289547	-0.920135	-0.740959	0.069987
4	-0.951915	-1.033114	-0.307562	-0.740959	0.069987

	Розмір_сім'ї	Є_батьком
0	-1.758359	-1.581139
1	0.449070	0.632456
2	-0.654644	-1.581139
3	0.449070	0.632456
4	0.449070	0.632456

Рисунок 2.13 - датафрейм, який будемо застосовувати надалі

2.4 Зменшення розмірності

У цій задачі є багато факторів, на основі яких буде зроблена остаточна класифікація. Ці фактори в основному є атрибутами або особливостями. Чим більше функцій, тим важче з нею працювати. Багато з цих функцій корельовані, а отже, зайві. Ось чому ми виконаємо зменшення розмірності для вибраних об’єктів перед тим, як пропустити їх через класифікатор.

Зменшення розмірності - це процес зменшення кількості випадкових величин, що розглядаються, шляхом отримання набору головних змінних.

Аналіз головних компонентів (PCA) — це техніка для зменшення розмірності таких наборів даних, підвищення інтерпретації, але водночас мінімізує втрату інформації.

Для того, щоб зрозуміти на скільки варто зменшувати розмірність, поглянемо на графік долі дисперсії головних компонент.

```
PC_values = np.arange(pca.n_components_) + 1
plt.bar(PC_values, pca.explained_variance_ratio_, color
="#682F2F")
plt.title('Графік головних компонент')
plt.xlabel('Головні компоненти')
plt.ylabel('Доля дисперсії')
plt.show()
print(pca.explained_variance_ratio_)
```

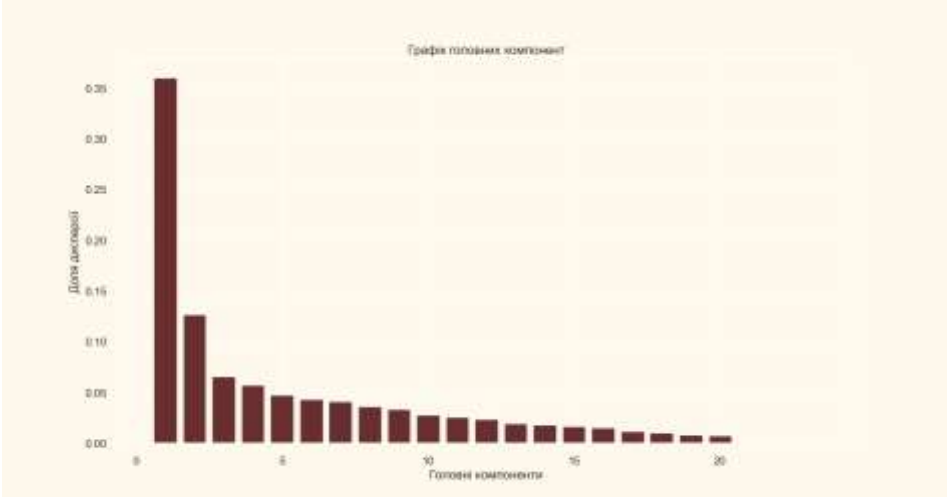

Рисунок 2.14 - Графік головних компонент (PCA)

З рисунків 2.14 та 2.15, що наведені вище, бачимо, що перші 3 компоненти беруть на себе більший відсоток дисперсії, тому зменшимо розмірність до 3.

```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(scaled_ds)
PCA_ds = pd.DataFrame(pca.transform(scaled_ds),
columns=(["col1","col2", "col3"]))
print(PCA_ds.describe().T)

#A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]

#To plot
```

```
[3.59919233e-01 1.28198452e-01 6.85562940e-02 6.07879404e-02
 4.75197959e-02 4.37575318e-02 4.30068752e-02 3.38071201e-02
 2.86016521e-02 2.79510592e-02 2.62200328e-02 2.30178712e-02
 1.94201019e-02 1.84364912e-02 1.70137538e-02 1.50784438e-02
 1.20597552e-02 1.03970951e-02 8.69385381e-03 7.55664721e-03
 2.70424387e-32 2.63031768e-32 4.13826387e-33]
```

Рисунок 2.15 - відсоткове співвідношення дисперсії кожного компонента

```
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("A 3D Projection Of Data In The Reduced
Dimension")
plt.show()
```

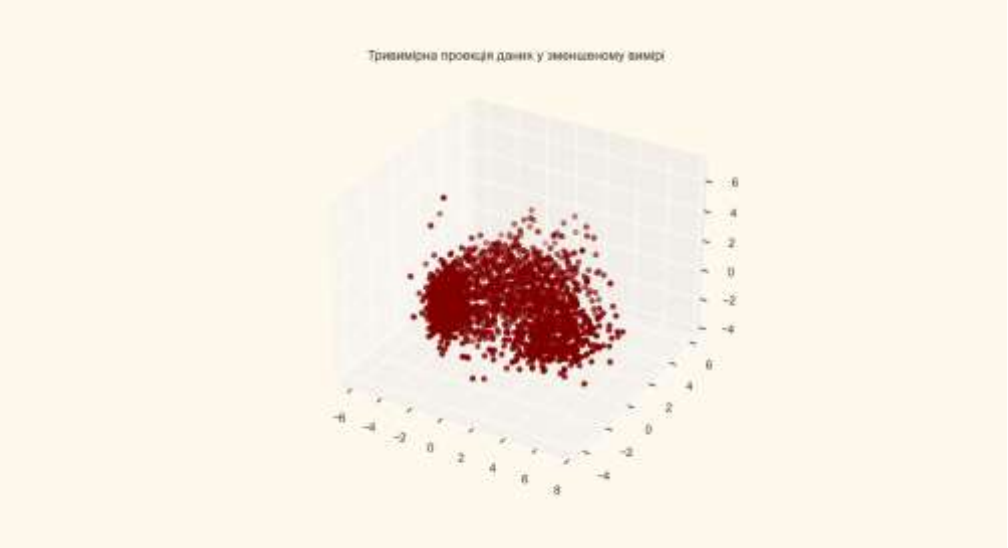

Рисунок 2.16 - 3D проєкція зменшених даних

2.5 Кластеризація

Тепер, коли було зменшено розмірність до трьох вимірів, ми будемо виконувати кластеризацію за допомогою методу агломеративної кластеризації. Метод агломеративної кластеризація — це ієрархічний метод кластеризації, який передбачає злиття прикладів, поки не буде досягнуто бажаної кількості кластерів.

Почнемо з методу ліктя для визначення оптимальної кількості кластерів, які мають бути сформовані:

```
# Quick examination of elbow method to find numbers of
clusters to make.
print('Elbow Method to determine the number of clusters to
be   formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()
```

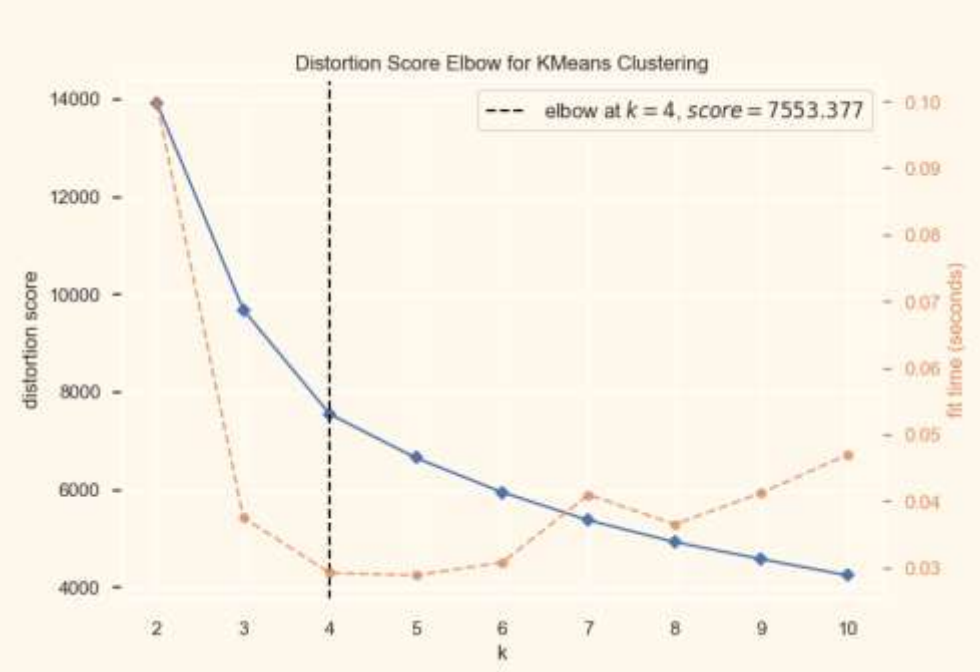

Рисунок 2.17 - метод ліктя

Наведений вище графік на рисунку 2.17 вказує, що оптимальною кількістю кластерів для цих даних буде чотири. Далі ми будемо адаптувати модель агломеративної кластеризації, щоб отримати остаточні кластери.

```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
```

```
PCA_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["Clusters"]= yhat_AC
```

Щоб дослідити сформовані кластери, давайте подивимося на їх тривимірний розподіл (рисунок 2.18).

```
#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o',
cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
```

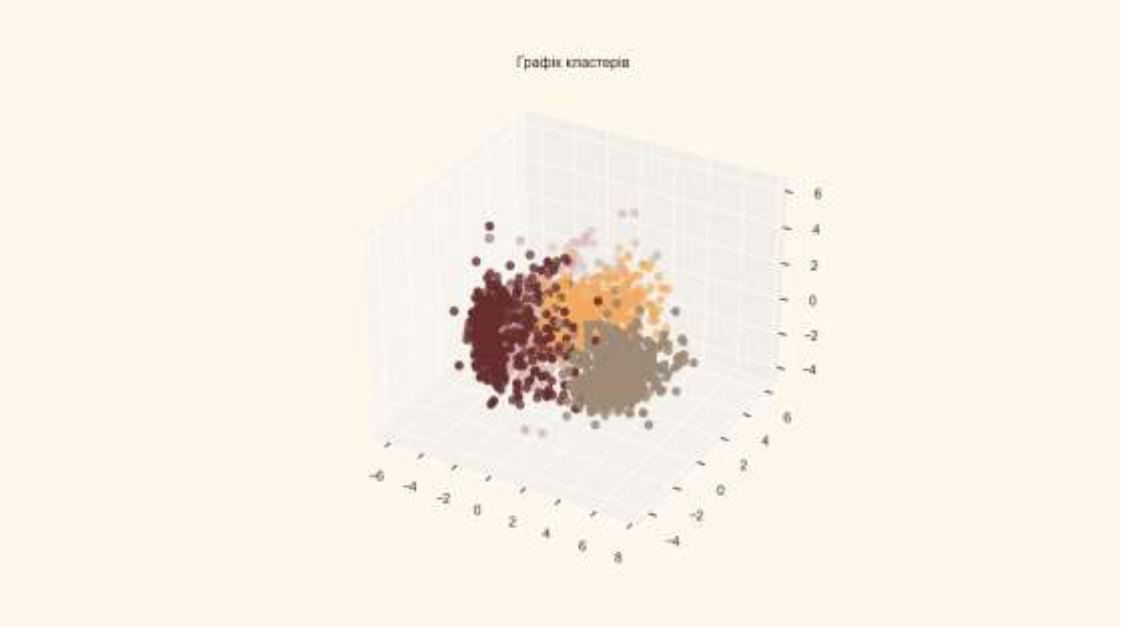

Рисунок 2.18 - розподіл утворених кластерів у тривимірному просторі

2.6 Оцінка моделей

Оскільки це неконтрольоване кластеризування, у нас немає функції з тегами, щоб дати оцінку або визначити нашу модель. Метою цього розділу є вивчення закономірностей у сформованих кластерах та вивчення характеру структур кластерів. Для цього ми будемо дивитися на дані в світлі кластерів за допомогою дослідницького аналізу даних і робити висновки.

По-перше, давайте подивимося на груповий розподіл кластеризації (рисунок 2.19 – рисунок 2.20).

```
#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
```

```
pl = sns.countplot(x=data["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()

pl = sns.scatterplot(data = data,x=data["Income"],
y=data["Spent"],hue=data["Clusters"], palette= pal)
pl.set_title("Cluster's Profile Based On Income And
Spending")
plt.legend()
plt.show()
```

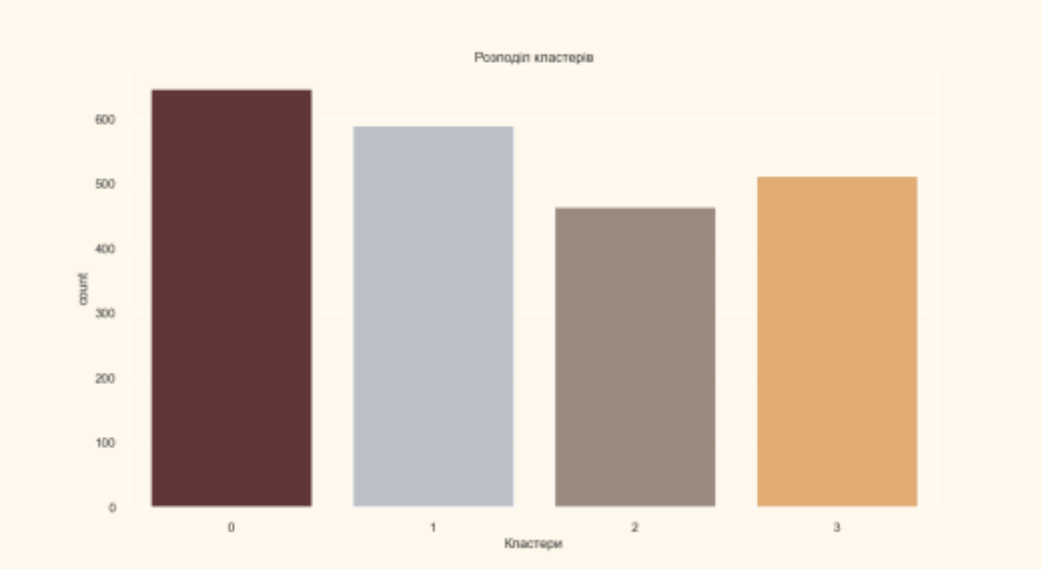

Рисунок 2.19 - графік розподілу кластерів

Рисунок 2.5 - графік розподілу кластерів за доходом та витратами

На графіку доходів і витрат показано модель кластерів, з якої ми можемо зробити такі висновки:

- група 0: низькі витрати та низький дохід;
- група 1: низькі витрати та середній дохід;
- група 2: великі витрати та високий дохід;
- група 3: середні витрати та середній дохід.

Далі розглянемо детальний розподіл кластерів за різними категоріями продуктів, що були надані в цьому датасеті, а саме: вина, фрукти, м'ясо, риба, солодощі та золото.

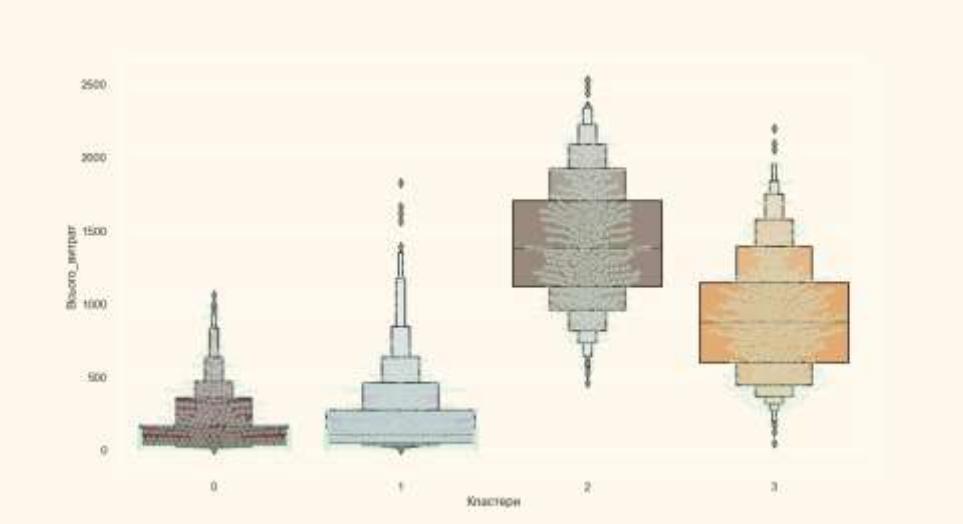

Рисунок 2.21 - графік розподілу клієнтів за кластерами

```
plt.figure()
pl=sns.swarmplot(x=data["Clusters"], y=data["Spent"],
color=    "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=data["Clusters"], y=data["Spent"],
palette=pal)
plt.show()
```

З наведеного вище графіка на рисуунку 2.21 чітко видно, що кластер 2 – це наша найбільша група клієнтів, за якою слідує кластер 3. Ми можемо дослідити, на що витрачає гроші кожен кластер для цільових маркетингових стратегій, тому давайте розглянемо результати рекламних кампаній магазину у минулому.

```
     #Creating a feature to get a sum of accepted
promotions
     data["Total_Promos"] = data["AcceptedCmp1"]+
     data["AcceptedCmp2"]+ data["AcceptedCmp3"]+
     data["AcceptedCmp4"]+ data["AcceptedCmp5"]
     #Plotting count of total campaign accepted.
     plt.figure()
```

```
    pl =
    sns.countplot(x=data["Total_Promos"],hue=data["Cluster
s"],     palette= pal)
    pl.set_title("Count Of Promotion Accepted")
    pl.set_xlabel("Number Of Total Accepted Promotions")
```

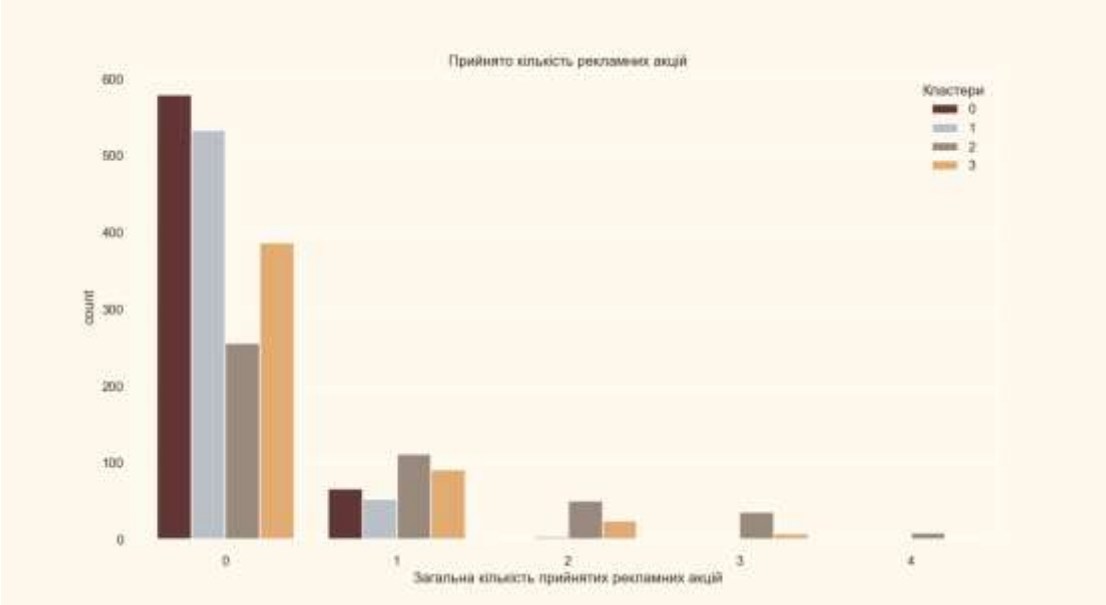

Рисунок 2.22 - графік рекламних кампаній

Наразі не було надзвичайного відгуку на кампанії. Загалом учасників дуже мало, крім того, ніхто не бере участь у всіх 5 з них. Можливо, для збільшення продажів потрібні більш цільові та краще сплановані кампанії.

```
    plt.figure()
    pl=sns.boxenplot(y=data["NumDealsPurchases"],x=data["C
lusters   "], palette= pal)
```

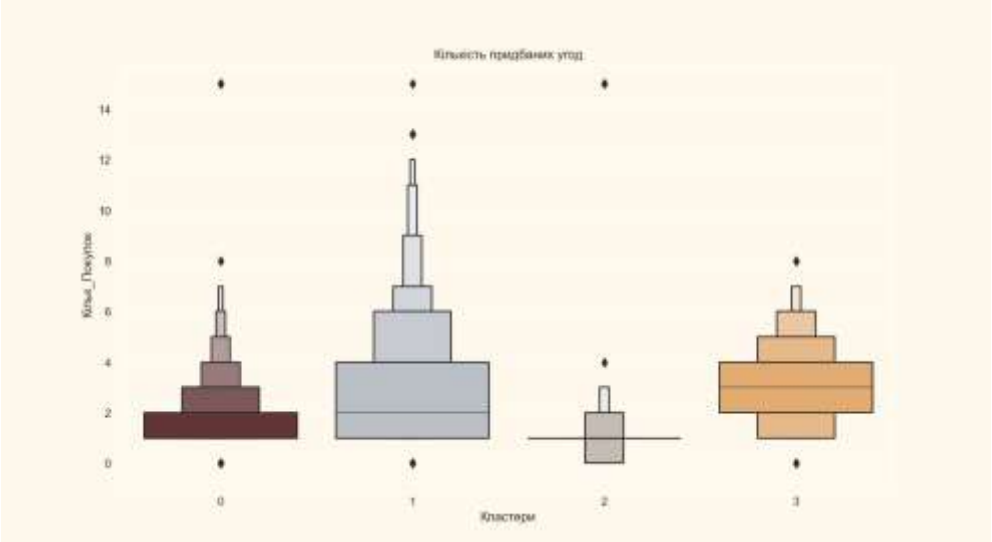

Рисунок 2.23 - кількість придбаних угод

```
        pl.set_title("Number of Deals Purchased")
        plt.show()
```

На відміну від кампаній, запропоновані бізнесом спеціальні пропозиції вдалися. Вони мають найкращі результати з кластером 1 і кластером 3. Однак наші найактивніші клієнти кластера 2 не надто люблять угоди.

2.7 Профайлинг

Тепер, коли ми сформували кластери та подивилися на їхні купівельні звички, давайте подивимося, які саме люди складають виявлені кластери і виявимо спільні риси. Для цього ми профілюємо сформовані кластери та дійдемо висновку про те, хто є нашим головним клієнтом, а хто потребує більше уваги з боку маркетингової команди магазину.

Визначимо деякі характеристики, які вказують на особисті риси клієнта, у світлі кластера, у якому вони перебувають. На основі результатів зможемо дійти певних висновків.

```
Personal = [ "Kidhome","Teenhome","Customer_For", "Age",
     "Children", "Family_Size", "Is_Parent",
     "Education","Living_With"]

for i in Personal:
     plt.figure()
     sns.jointplot(x=data[i], y=data["Spent"], hue
               =data["Clusters"], kind="kde", palette=pal)
plt.show()
```

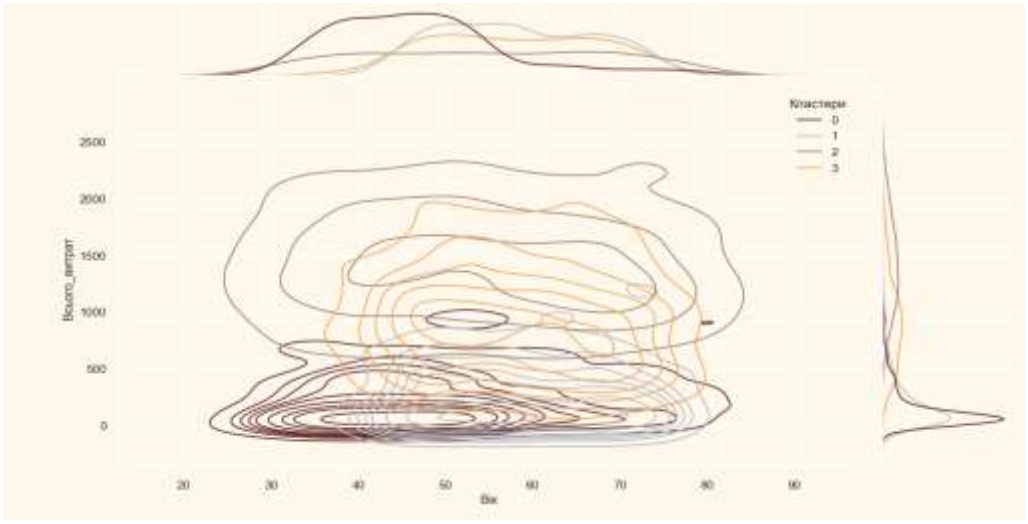

Рисунок 0.24 - розподіл кластерів за віком

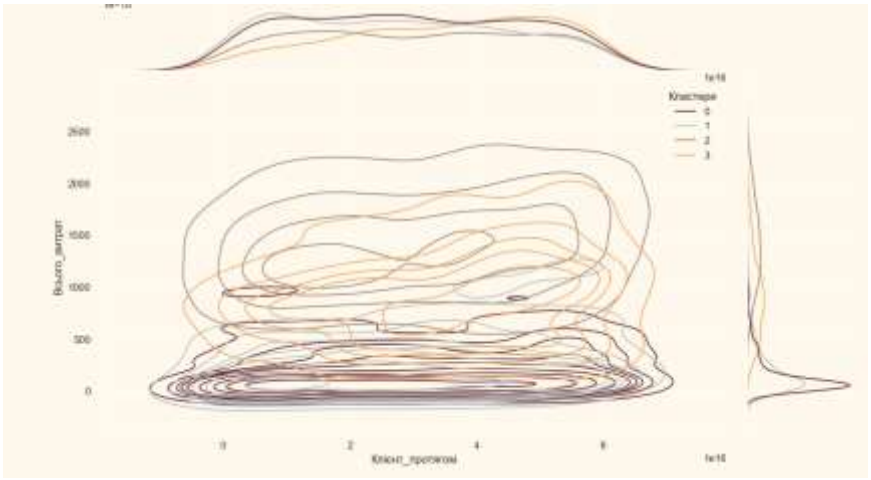

Рисунок 2. 6 - розподіл кластерів за тим, скільки покупець є клієнтом даного магазину

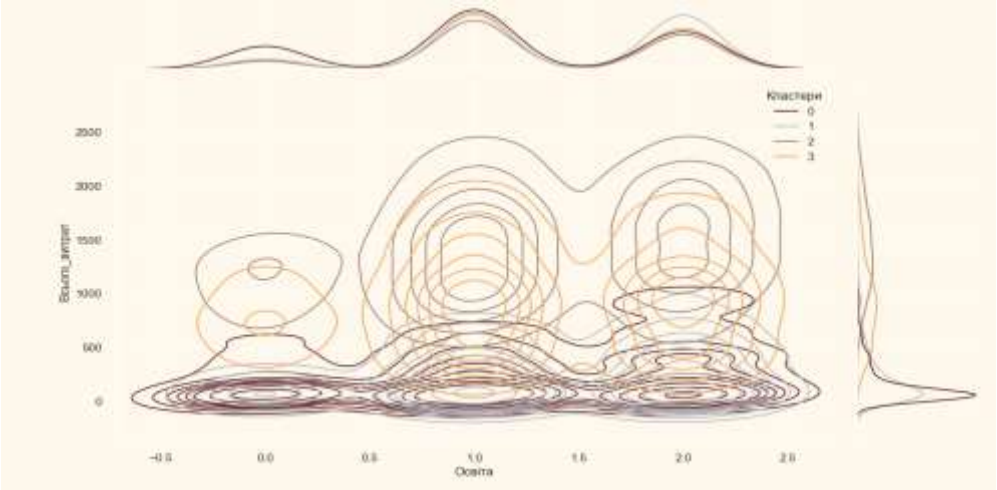

Рисунок 0.26 - розподіл кластерів за рівнем освіти

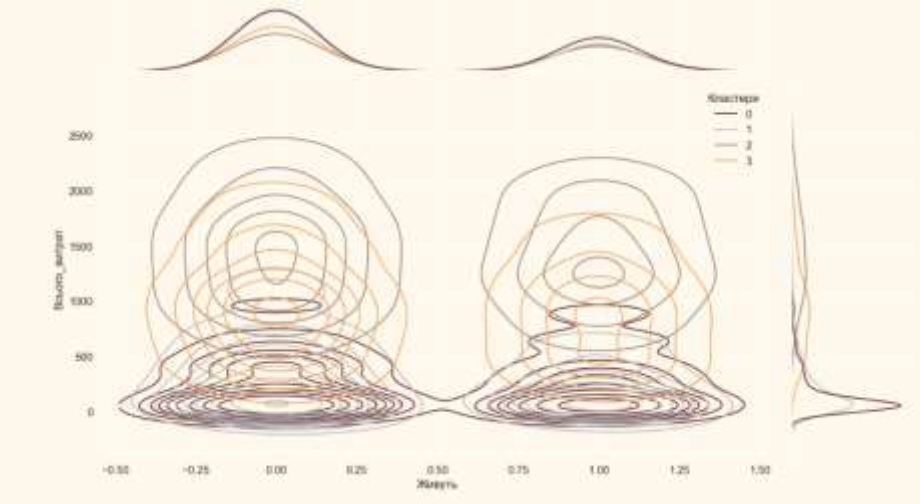

Рисунок 0.27 - розподіл кластерів за наявністю партнера

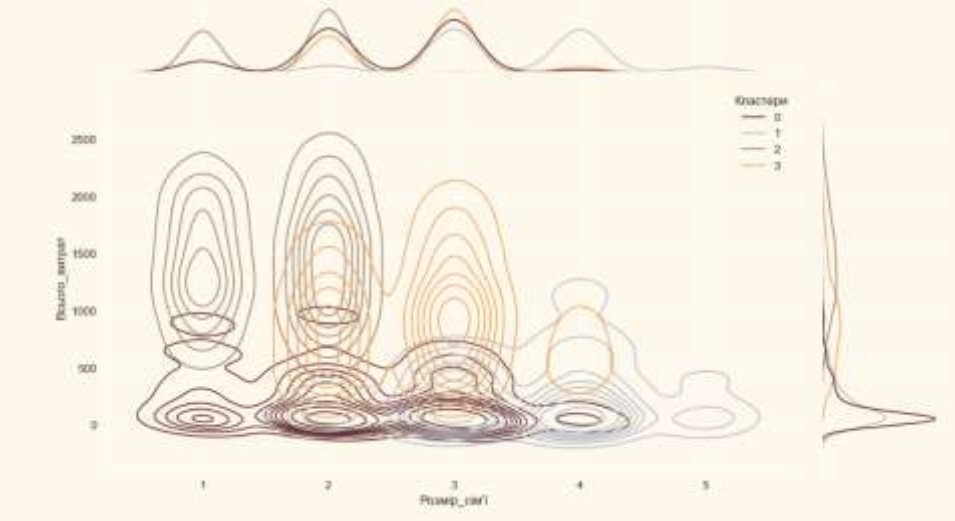

Рисунок 0.29 - розподіл кластерів за кількістю людей в сім’ї

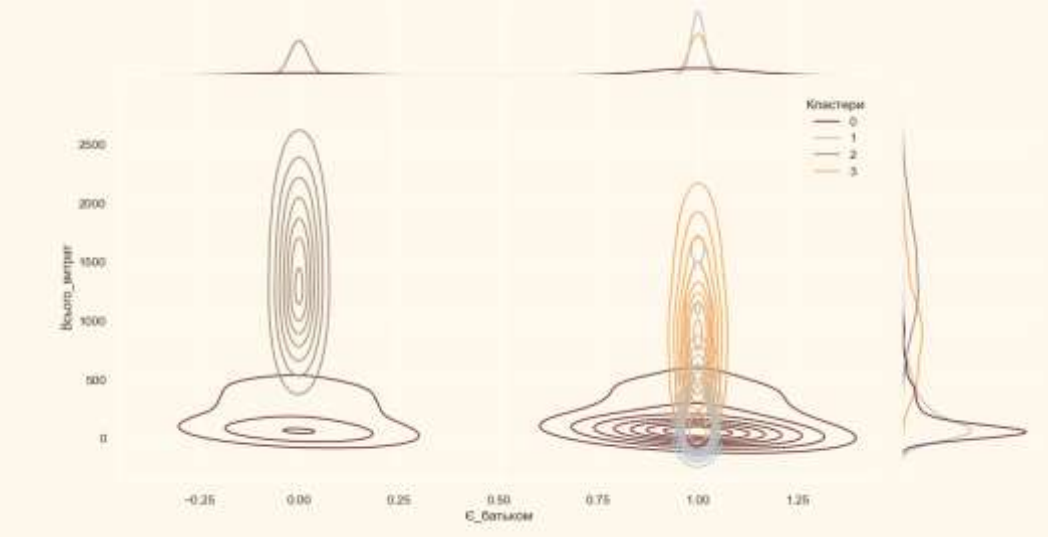

Рисунок 0.7 - розподіл кластерів за тим, чи є клієнти батьками чи ні

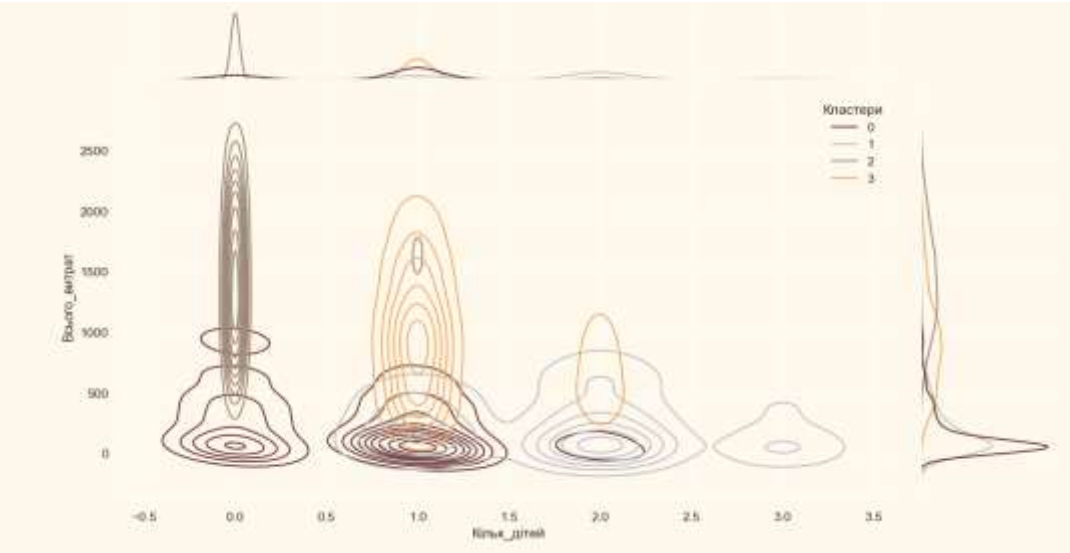

Рисунок 0.30 - розподіл кластерів за кількістю дітей загалом

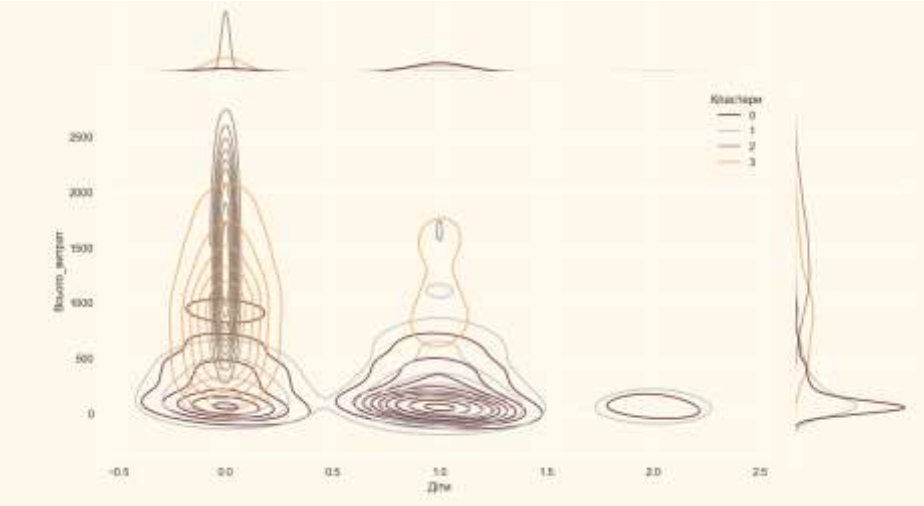

Рисунок 0.31 - розподіл кластерів за кількістю дітей не підлітків в сім’ї

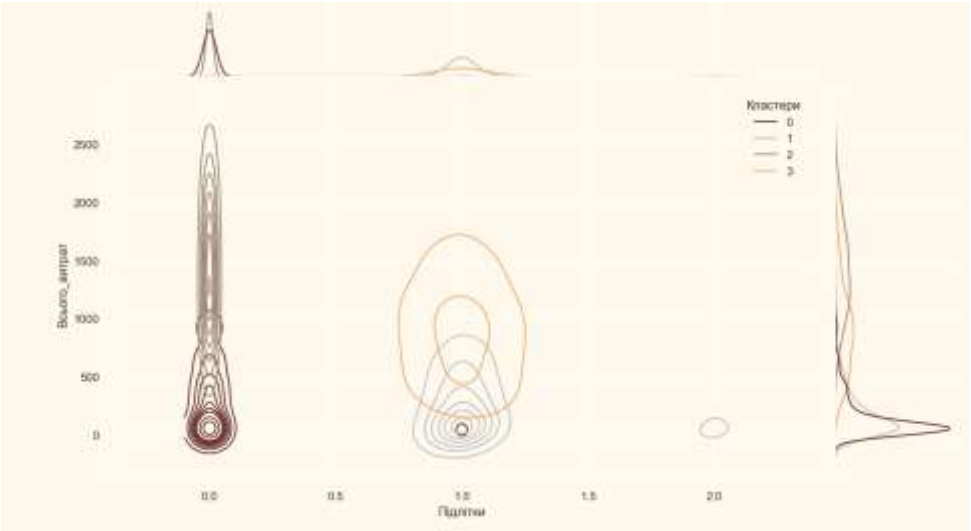

Рисунок 0.32 - розподіл кластерів за кількістю підлітків в сім’ї

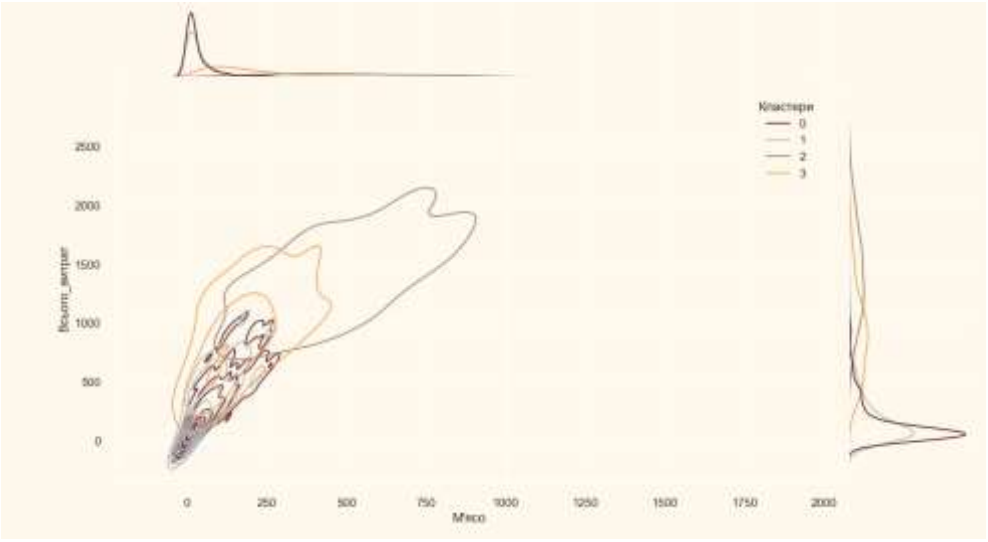

Рисунок 2.33 - розподіл кластерів за купівлею м’яса

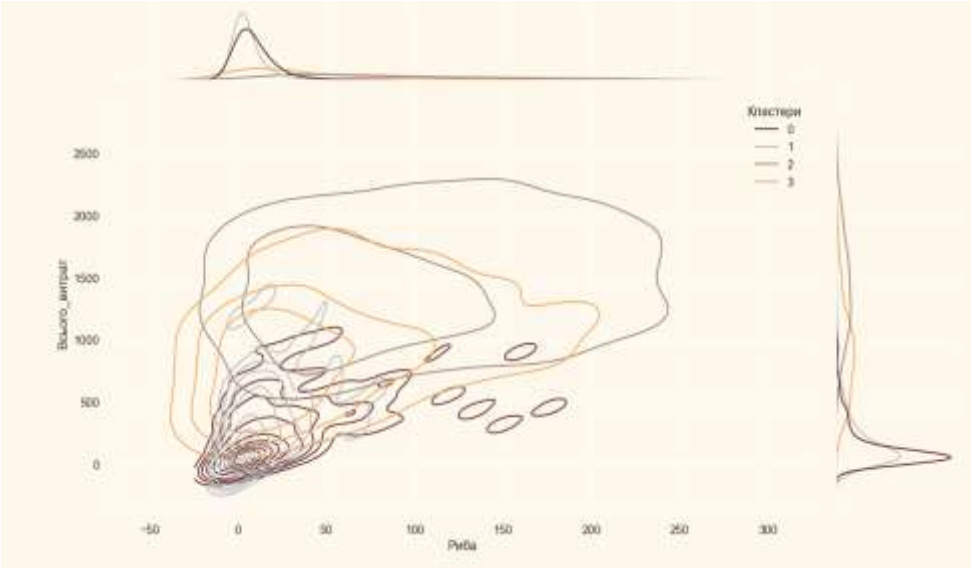

Рисунок 2.34 - розподіл кластерів за купівлею риби

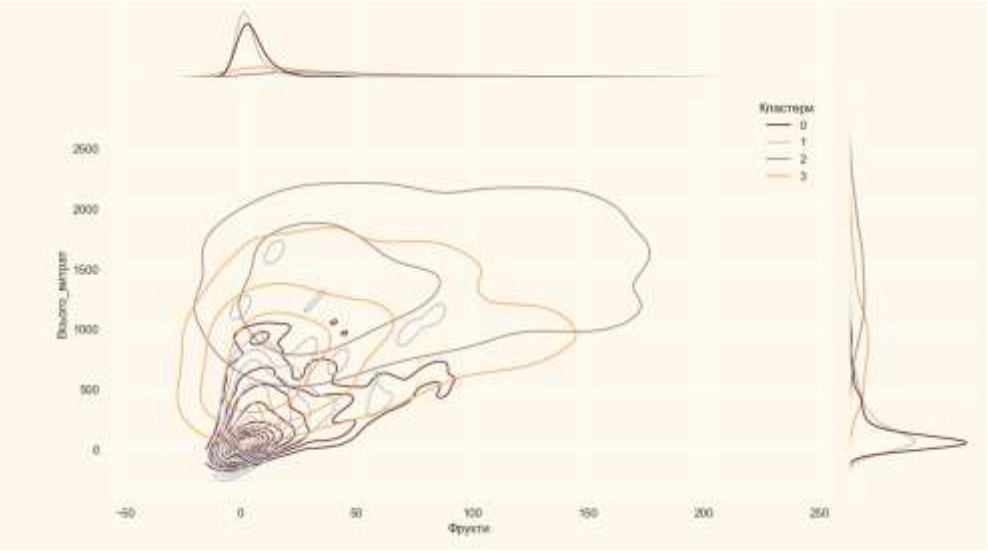

Рисунок 2.35 - розподіл кластерів за купівлею фруктів

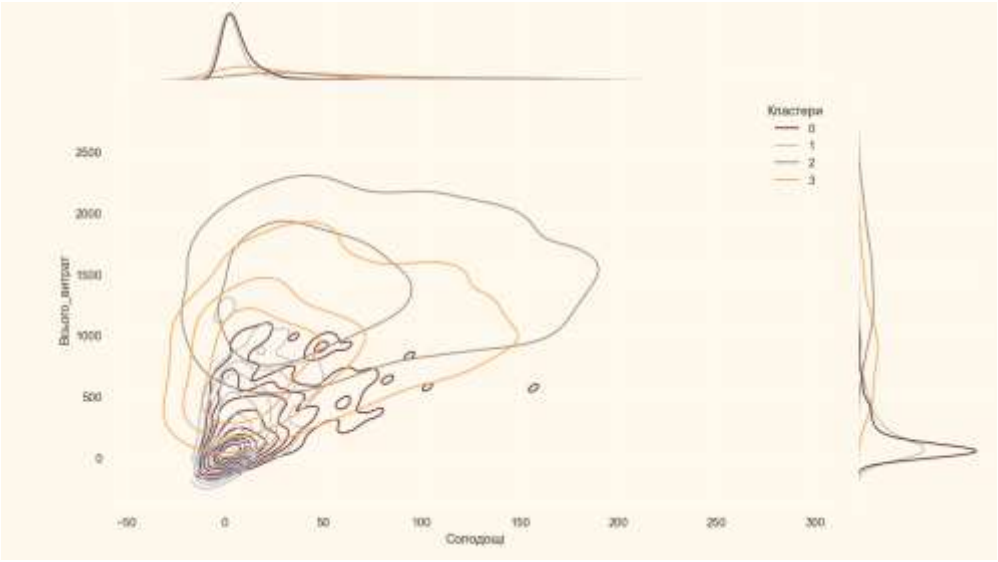

Рисунок 2.36 - розподіл кластерів за купівлею солодощів

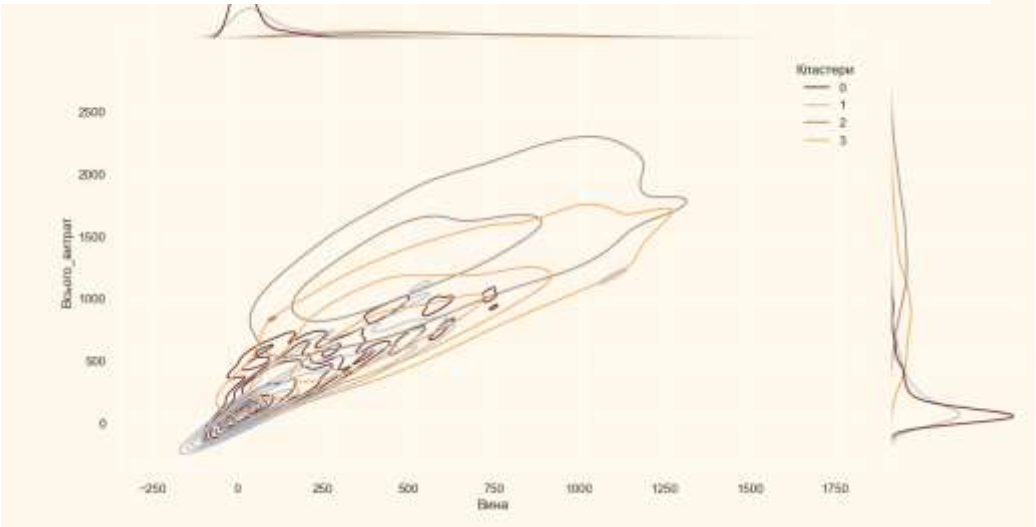

Рисунок 0.38 - розподіл кластерів за купівлею вин

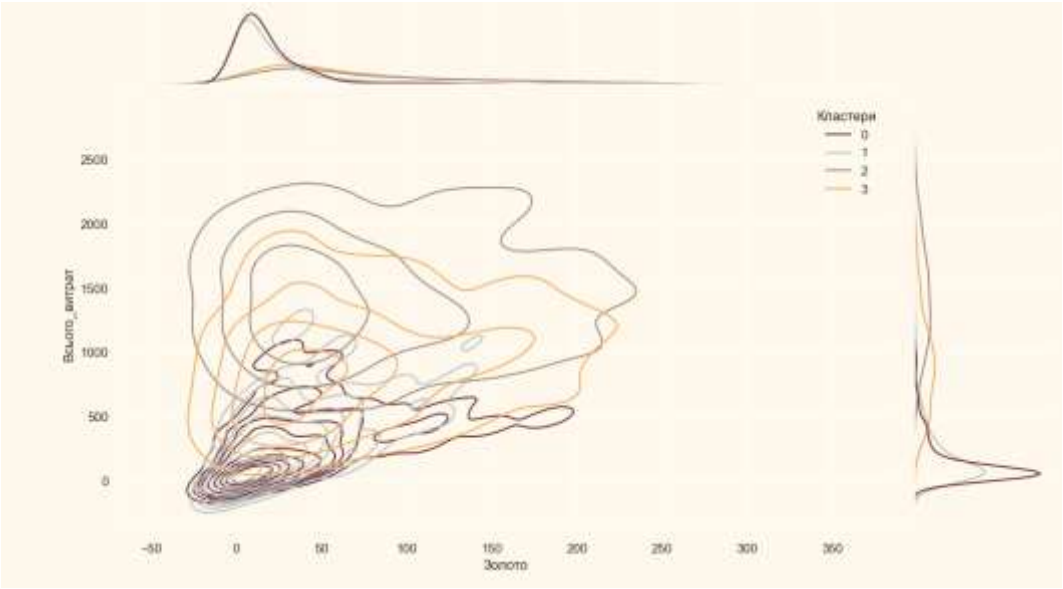

Рисунок 2.37 - розподіл кластерів за купівлею золота

Аналізуючи дані графіки, можемо зробити висновок по кожному з кластерів:

- Група 0 – 50% групи є батьками переважно одного (рідше двох) малих дітей до підліткового віку, 50% не мають дітей; здебільшого не мають вищу освіту, або в процесі її отримання, близько четвертини групи мають вищу освіту; живуть в сім’ї 1-4 людей, але переважно 2-3; знаходяться у віці 20-80 років (врахуємо те, що дані доволі старі);майже не купують вин та м’яса, виконують посередні (рідше високі) витрати на такі товари як фрукти, солодощі, риба і золото;
- Група 1 – точно є батьками (багатодітними); близько 70% не має вищої освіти або в процесі її отримання; переважно мають велику сім’ю з 2-5 людей; зазвичай мають не більше 1 дитини; знаходяться у віці 40-70 років; за роки відвідування магазину витрачають там невеликі суми на вина, м’ясо та солодощі, мають середні витрати на рибу, фрукти та золото;
- Група 2 – точно не мають дітей; переважно не мають вищої освіти, або в процесі її здобування; в рівній кількості живуть на самоті, або з партнером; знаходять у віці 25-80 років; у магазині витрачають багато грошей на всі групи товарів;
- Група 3 – точно є батьками одного-двох дітей (зазвичай підлітків); переважно не мають вищої освіти, або в процесі її здобуття; мають сім’ї з 2-3 (рідше чотирьох) людей; знаходяться у віці 35-80 років; за роки відвідування магазину витрачають значні суми на всі групи товарів, окрім м’яса.

ВИСНОВКИ

В ході цього дослідження, ми успішно застосували кілька технік машинного навчання для розуміння і кластеризації нашої клієнтської бази. Починаючи з обробки, ми приділяли особливу увагу підготовці наших даних для аналізу.

Перед тим, як зробити будь-які значущі висновки, ми здійснили важливий процес зменшення розмірності даних. Для цього ми використали метод головних компонент (PCA). Цей метод дозволив нам знизити багатовимірні дані до меншого числа вимірів без значної втрати інформації. Зменшення розмірності відіграє важливу роль в обробці даних, оскільки воно дозволяє видалити шум і полегшує візуалізацію даних, а також спрощує виконання обчислень. В результаті PCA, ми отримали новий набір ознак, які можна було легко використовувати для кластеризації.

Після того, як було зменшено розмірність, ми зосередилися на визначенні оптимальної кількості кластерів для нашого набору даних. Для цього ми використали метод ліктя. Цей метод базується на побудові графіка варіації відстані в залежності від кількості кластерів, де "лікоть" в графіку вказує на оптимальну кількість кластерів. Ми виявили, що чотири - це оптимальна кількість кластерів для нашого набору даних.

Нарешті, ми використали метод агломеративної кластеризації для групування наших клієнтів. Цей ієрархічний метод використовує принцип пошуку найменших відстаней для об'єднання об'єктів в кластери, поступово будуючи більші й більші групи. Цей метод створив ієрархічну структуру кластерів, яка показала чітку відмінність між різними групами клієнтів.

На основі аналізу ми змогли ідентифікувати чотири основні групи покупців. Ці групи включали різні типи клієнтів, кожна з яких мала свої унікальні характеристики і поведінку. Ця інформація тепер може бути використана для створення більш цільових та ефективних стратегій маркетингу та продажу, що відповідають специфічним потребам кожної групи.

В цілому, ця робота показала, що кластеризація є могутнім інструментом для розуміння і сегментації клієнтської бази. Використовуючи комбінацію PCA, методу ліктя та агломеративної кластеризації, ми змогли отримати глибоке розуміння нашої клієнтської бази, що дозволило нам краще відповідати на потреби наших клієнтів. Методи, використані в цій роботі, можуть бути застосовані до широкого спектру інших доменів і сфер, де розуміння і категоризація великих груп людей є важливою.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. ACM Computing Surveys (CSUR)
2. Tan, P. N., Steinbach, M., & Kumar, V. (2005). Introduction to Data Mining. Addison-Wesley Longman Publishing Co., Inc.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics.
4. Murtagh, F., & Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement Ward's criterion?. Journal of classification
5. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.
6. Lloyd, S. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory
7. Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science
8. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences
9. Ketchen, D. J., & Shook, C. L. (1996). The application of cluster analysis in strategic management research: an analysis and critique. Strategic management journal
10. Thorndike, R. L. (1953). Who belongs in the family?. Psychometrika

ДОДАТОК А

```
# -*- coding: cp1251 -*-

#Importing the Libraries
import numpy as np
import pandas as pd
import datetime
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import AgglomerativeClustering
from matplotlib.colors import ListedColormap
from sklearn import metrics
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)

#Loading the dataset
data =
pd.read_csv("c:/Users/46325/OneDrive/Desktop/diploma/market
ing_campaign.csv", sep="\t")
print("Кількість даних:", len(data))
data.head()

#Information on features
# data.info()

data=data.rename(columns={"Year_Birth":
"Дата_народження","Education":"Освіта","Marital_Status":"Ci
мейний_стан","Income":"Дохід","Kidhome":"Діти","Teenhome":"
Підлітки","Dt_Customer":"Дата_клієнт","Recency":"Нещодавніс
ть", "MntWines":"Вина","MntFruits":"Фрукти",
"MntMeatProducts":"М'ясо", "MntFishProducts":"Риба",
"MntSweetProducts":"Солодощі",
"MntGoldProds":"Золото","NumDealsPurchases":"Кільк_Покупок"
, "NumWebPurchases":"Кільк_Покупок_Сайт",
```

```
"NumCatalogPurchases":"Кільк_Покупок_Каталог",
"NumStorePurchases":"Кільк_Покупок_Магазин",
"NumWebVisitsMonth":"Кільк_Відв_Сайту_Міс",
"AcceptedCmp1":"Прийн_Камп1", "AcceptedCmp2":"Прийн_Камп2",
"AcceptedCmp3":"Прийн_Камп3", "AcceptedCmp4":"Прийн_Камп4",
"AcceptedCmp5":"Прийн_Камп5", "Complain":"Скарг",
"Response":"Відповідь"})
data["Сімейний_стан"]=data["Сімейний_стан"].replace({"Marri
ed":"Одружений", "Together":"Разом",
"Absurd":"Все_складно", "Widow":"Вдова",
"Divorced":"Розлучений",
"Single":"Самотній","Alone":"Один"})
data["Освіта"]=data["Освіта"].replace({"Basic":"Базова","2n
Cycle":"Бакалавр", "Graduation":"Випускаються",
"Master":"Магістр", "PhD":"Докторант"})

for_drop = ["Z_CostContact", "Z_Revenue"]
data = data.drop(for_drop, axis=1)

data.info()

#To remove the NA values
data = data.dropna()
print("Загальна кількість даних після видалення рядків із
відсутніми значеннями:", len(data))

data["Дата_клієнт"] = pd.to_datetime(data["Дата_клієнт"],
dayfirst=True)
dates = []
for i in data["Дата_клієнт"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("Дата реєстрації останнього клієнта в
записах:",max(dates))
print("Дата реєстрації першого клієнта в
записах:",min(dates))

#Created a feature "Клієнт_протягом"
days = []
d1 = max(dates) #taking it to be the newest customer
for i in dates:
    delta = d1 - i
    days.append(delta)
data["Клієнт_протягом"] = days
data["Клієнт_протягом"] =
pd.to_numeric(data["Клієнт_протягом"], errors="coerce")
print("Всього категорій у функції Сімейний стан:\n",
data["Сімейний_стан"].value_counts(), "\n")
print("Всього категорій у функції Освіта:\n",
data["Освіта"].value_counts())
```

```
#Age of customer today
data["Вік"] = 2023-data["Дата_народження"]

#Total spendings on various items
data["Всього_витрат"] = data["Вина"]+ data["Фрукти"]+
data["М'ясо"]+ data["Риба"]+ data["Солодощі"]+
data["Золото"]

#Deriving living situation by marital status
data["Живуть"]=data["Сімейний_стан"].replace({"Одружений":"
З_партнером", "Разом":"З_партнером", "Все_складно":"Один",
"Вдова":"Один", "YOLO":"Один", "Розлучений":"Один",
"Самотній":"Один"})

#Feature indicating total children living in the household
data["Кільк_дітей"]= data["Діти"] + data["Підлітки"]

#Feature for total members in the householde
data["Розмір_сім'ї"] = data["Кільк_дітей"] +
data["Живуть"].replace({"З_партнером":2, "Один":1})

#Feature pertaining parenthood
data["Є_батьком"] = np.where(data.Кільк_дітей> 0, 1, 0)

#Segmenting education levels in three groups
data["Освіта"]=data["Освіта"].replace({"Базова":"Без_вищої_
освіти","Бакалавр":"Без_вищої_освіти",
"Випускаються":"Здобувають_вищу_освіту",
"Магістр":"Мають_вищу_освіту",
"Докторант":"Мають_вищу_освіту"})

#Dropping some of the redundant features
to_drop = ["Сімейний_стан", "Дата_клієнт",
"Дата_народження", "ID"]
data = data.drop(to_drop, axis=1)

pd.set_option('display.max_columns', None)
print(data.describe(include='all'))

#To plot some selected features
#Setting up colors prefrences
sns.set(rc={"axes.facecolor":"#FFF9ED","figure.facecolor":"
#FFF9ED"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9",
"#9F8A78", "#F3AB60"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F",
"#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"])
#Plotting following features
To_Plot = [ "Дохід", "Нещодавність", "Клієнт_протягом",
"Вік", "Всього_витрат", "Є_батьком"]
```

```
sns.pairplot(data[To_Plot], hue= "Є_батьком",palette=
(["#682F2F","#F3AB60"]), size=1.5, aspect=1)
plt.rcParams['figure.figsize'] = [4, 4]
# Taking hue
plt.show()

#Dropping the outliers by setting a cap on Age and income.
data = data[(data["Вік"]<90)]
data = data[(data["Дохід"]<600000)]
print("Загальна кількість даних після видалення викидів:",
len(data))

#correlation matrix
temp= data.drop(columns=['Освіта', 'Живуть'])
corrmat= temp.corr()
plt.rcParams.update({'font.size': 6})
plt.figure(figsize=(20,15))
sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
plt.show()

#Get list of categorical variables
s = (data.dtypes == 'object')
object_cols = list(s[s].index)

print("Категоріальні змінні в наборі даних:", object_cols)

#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
    data[i]=data[[i]].apply(LE.fit_transform)

print("Усі функції тепер числові")

#Creating a copy of data
ds = data.copy()
# creating a subset of dataframe by dropping the features
on deals accepted and promotions
cols_del = ['Прийн_Камп3', 'Прийн_Камп4', 'Прийн_Камп5',
'Прийн_Камп1','Прийн_Камп2', 'Скарг', 'Відповідь']
ds = ds.drop(cols_del, axis=1)

#Scaling
scaler = StandardScaler()
scaler.fit(ds)
scaled_ds = pd.DataFrame(scaler.transform(ds),columns=
ds.columns )
print("Усі функції тепер масштабовано")

#Scaled data to be used for reducing the dimensionality
print("Дата фрейм для подальшого моделювання:")
print(scaled_ds.head())
```

```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=23)
pca.fit(scaled_ds)
PCA_ds = pd.DataFrame(pca.transform(scaled_ds),
columns=(["col1","col2", "col3", "col4", "col5", "col6",
"col7", "col8", "col9", "col10", "col11", "col12", "col13",
"col14", "col15", "col16", "col17", "col18", "col19",
"col20", "col21", "col22", "col23"]))
print(PCA_ds.describe().T)

PC_values = np.arange(pca.n_components_) + 1
plt.bar(PC_values, pca.explained_variance_ratio_, color
="#682F2F")
plt.title('Графік головних компонент')
plt.xlabel('Головні компоненти')
plt.ylabel('Доля дисперсії')
plt.show()
print(pca.explained_variance_ratio_)

#A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]

#To plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("Тривимірна проєкція даних у зменшеному
вимірі")
plt.show()

# Quick examination of elbow method to find numbers of
clusters to make.
print('Метод ліктя для визначення кількості кластерів, які
мають бути сформовані:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()

#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["Кластери"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["Кластери"]= yhat_AC

#Plotting the clusters
fig = plt.figure(figsize=(10,8))
```

```
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Кластери"], marker='o',
cmap = cmap )
ax.set_title("Графік кластерів")
plt.show()

# #Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=data["Кластери"], palette= pal)
pl.set_title("Розподіл кластерів")
plt.show()

pl = sns.scatterplot(data = data,x=data["Дохід"],
y=data["Всього_витрат"],hue=data["Кластери"], palette= pal)
pl.set_title("Кластерний розподіл на основі доходів і
витрат")
plt.legend()
plt.show()

plt.figure()
pl=sns.swarmplot(x=data["Кластери"],
y=data["Всього_витрат"], color= "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=data["Кластери"],
y=data["Всього_витрат"], palette=pal)
plt.show()

#Creating a feature to get a sum of accepted promotions
data["Всього_камп"] = data["Прийн_Камп1"]+
data["Прийн_Камп2"]+ data["Прийн_Камп3"]+
data["Прийн_Камп4"]+ data["Прийн_Камп5"]
#Plotting count of total campaign accepted.
plt.figure()
pl =
sns.countplot(x=data["Всього_камп"],hue=data["Кластери"],
palette= pal)
pl.set_title("Прийнято кількість рекламних акцій")
pl.set_xlabel("Загальна кількість прийнятих рекламних
акцій")
plt.show()

#Plotting the number of deals purchased
plt.figure()
pl=sns.boxenplot(y=data["Кільк_Покупок"],x=data["Кластери"]
, palette= pal)
pl.set_title("Кількість придбаних угод")
plt.show()

Personal = [ "Діти","Підлітки","Клієнт_протягом", "Вік",
"Кільк_дітей", "Розмір_сім'ї", "Є_батьком",
"Освіта","Живуть", "Вина", "Фрукти", "М'ясо", "Риба",
"Солодощі", "Золото"]
```

```
for i in Personal:
    plt.figure()
    sns.jointplot(x=data[i], y=data["Всього_витрат"], hue
=data["Кластери"], kind="kde", palette=pal)
    plt.show()
```