**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра теорії та технології програмування

Звіт

до лабораторної роботи №2 на тему

«Сегментація клієнтів»

Виконав студент 4-го курсу

Групи ТТП-42

Ходаков Максима Олеговича

**Київ – 2025**

**Анотація**

У цій лабораторній роботі було реалізовано:

* **алгоритм K-Means кластеризації** для сегментації клієнтів торгового центру на основі їхніх характеристик (вік, річний дохід та рейтинг витрат);
* **досліджувальний аналіз даних (EDA)**, включаючи описову статистику, окремі гістограми, парні діаграми та кореляційну матрицю для виявлення закономірностей та взаємозв'язків між ознаками;
* **визначення оптимальної кількості кластерів** за допомогою Методу Ліктя та Аналізу Силуетів для забезпечення якісної кластеризації;
* **візуалізацію результатів кластеризації** за допомогою методу головних компонент (PCA) та діаграм коробок (box plots) для інтерпретації характеристик кожного кластеру;
* **збереження всіх візуалізацій** у папку images для подальшого використання у звітах та презентаціях.

Для реалізації використовувалась мова програмування Python, яка має розвинену екосистему бібліотек для обробки даних, візуалізації та машинного навчання. Було використано такі бібліотеки:

* **pandas** – універсальна бібліотека для роботи з табличними даними, що надає зручні структури (DataFrame) та операції для маніпуляцій із даними;
* **numpy** – бібліотека для підтримки великих багатовимірних масивів та матриць, а також високорівневих математичних функцій;
* **matplotlib** і **seaborn** – для створення статичних, анімаційних та інтерактивних візуалізацій даних;
* **scikit-learn** – бібліотека для машинного навчання, що надає інструменти для кластеризації, оцінки якості моделей та зменшення вимірності даних (PCA).

Таким чином, у межах лабораторної роботи було показано, як здійснити сегментацію клієнтів за допомогою алгоритму K-Means, провести детальний аналіз даних для вибору релевантних ознак та визначення кількості кластерів, а також візуалізувати та інтерпретувати результати кластеризації для прийняття обґрунтованих бізнес-рішень.

**Вступ**

Сегментація клієнтів є ключовим етапом у розвитку сучасних бізнес-стратегій, оскільки дозволяє ідентифікувати та розділити споживачів на окремі групи з подібними характеристиками та поведінкою. Це сприяє більш ефективному таргетуванню маркетингових кампаній, підвищенню рівня задоволеності клієнтів та збільшенню прибутковості компанії. Метою даного проєкту є сегментація клієнтів торгового центру на окремі групи на основі їхніх спільних характеристик.

Основним концептом, використаним у цьому дослідженні, є кластеризація, зокрема алгоритм K-means кластеризації. K-means є одним із найпоширеніших методів кластеризації, який дозволяє ефективно розділити дані на задану кількість кластерів, мінімізуючи внутрішньокластерну варіативність.

Для реалізації кластеризації було використано мову програмування Python, яка має розвинену екосистему бібліотек для обробки даних, машинного навчання та візуалізації. Основні бібліотеки, що були використані у цьому проєкті:

* **pandas** – універсальна бібліотека для роботи з табличними даними, що надає зручні структури (DataFrame) та операції для маніпуляцій із даними;
* **numpy** – бібліотека для підтримки великих багатовимірних масивів та матриць, а також високорівневих математичних функцій;
* **matplotlib** та **seaborn** – для створення статичних, анімаційних та інтерактивних візуалізацій даних;
* **scikit-learn** – бібліотека для машинного навчання, що надає інструменти для кластеризації, оцінки якості моделей та зменшення вимірності даних (PCA).

Використовувані набори даних включають:

* **Mall Customer Segmentation Data** – дані про клієнтів торгового центру, що містять інформацію про вік, річний дохід та рейтинг витрат;
* **Credit Card Dataset for Clustering** – набір даних про використання кредитних карток, який може бути використаний для подальших аналізів та кластеризації.

Процес роботи був розбитий на кілька ключових завдань:

1. **Передобробка Даних**
   * **Завантаження Даних**: Імпорт набору даних у відповідний формат для подальшої обробки.
   * **Очищення Даних**: Обробка відсутніх значень та аномалій, видалення дублікованих записів.
   * **Вибір Ознак**: Вибір релевантних ознак для кластеризації, що мають найбільший вплив на сегментацію.
   * **Нормалізація Даних**: Масштабування даних для забезпечення рівного внеску всіх ознак у розрахунки відстаней.
2. **Досліджувальний Аналіз Даних (EDA)**
   * **Аналіз Розподілу Ключових Змінних**: Огляд статистичних показників та візуалізація розподілу основних ознак.
   * **Візуалізація Взаємозв'язків Між Ознаками**: Використання діаграм розсіяння, гістограм та кореляційних матриць для виявлення закономірностей та взаємозв'язків між ознаками.
3. **Реалізація K-means Кластеризації**
   * **Вибір Кількості Кластерів (k)**: Використання Методу Ліктя та Аналізу Силуетів для визначення оптимальної кількості кластерів.
   * **Кластеризація**: Застосування алгоритму K-means для сегментації клієнтів на окремі групи.
   * **Аналіз Кластерів**: Оцінка характеристик кожного кластеру, таких як середній вік, річний дохід та рейтинг витрат клієнтів.
4. **Оцінка Якості Кластерів**
   * **Оцінка Якості Кластеризації**: Використання метрик, таких як силуетний бал, для оцінки ефективності кластеризації.
   * **Інтерпретація Кластерів**: Аналіз кластерів у контексті бізнес-проблеми для визначення їхнього практичного значення.
5. **Візуалізація**
   * **Візуалізація Кластерів за Допомогою PCA**: Зменшення вимірності даних до двох компонентів для легшої візуалізації кластерів.
   * **Створення Графічних Представлень**: Побудова діаграм коробок (box plots) та інших візуальних інструментів для ілюстрації характеристик кожного кластеру.

**Теоретична частина**

### 1. Кластеризація

**Кластеризація** є одним з основних методів у сфері аналізу даних та машинного навчання, який полягає у розподілі об'єктів на групи, або **кластерні структури**, з максимально схожими характеристиками в межах кожного кластеру, і водночас з мінімальними схожостями між різними кластерами. Це дозволяє виявляти природні структури в даних, сегментувати об'єкти або споживачів, та створювати більш точні та індивідуальні стратегії для кожної групи.

Основним критерієм кластеризації є подібність об'єктів в одному кластері та відмінність об'єктів різних кластерів. Кластеризація є одним з методів **неконтрольованого навчання** (unsupervised learning), оскільки не потребує попередньо відомих міток або цільових значень для класифікації об'єктів.

### 2. Алгоритм K-Means Кластеризації

**K-Means кластеризація** — один з найпоширеніших та найпростіших алгоритмів кластеризації, який дозволяє групувати об'єкти на основі схожості їхніх характеристик. Основна ідея алгоритму полягає в тому, щоб поділити набір об'єктів на задану кількість кластерів, де кожен об'єкт належить до того кластеру, центр якого є найближчим.

#### Принцип Роботи Алгоритму K-Means:

1. **Ініціалізація**: Вибирається кількість кластерів (K). Існує кілька способів ініціалізації, але зазвичай початкові центри кластерів вибираються випадковим чином.
2. **Призначення об'єктів**: Кожен об'єкт (дані клієнта, продукт тощо) призначається до найближчого центру кластера. Це робиться на основі визначеної метрики схожості (найчастіше використовується евклідова відстань).
3. **Оновлення центрів кластерів**: Після того як всі об'єкти розподілені по кластерах, обчислюються нові центри кластерів як середнє значення всіх точок у кожному кластері.
4. **Повторення кроків 2 і 3**: Процес повторюється, поки центри кластерів не перестануть змінюватися або поки не буде досягнуто заданої кількості ітерацій.

#### Властивості Алгоритму K-Means:

* **Швидкість**: Алгоритм є дуже ефективним для великих обсягів даних завдяки простоті обчислень.
* **Гнучкість**: K-Means можна адаптувати до різних типів даних і задач.
* **Обмеження**: Алгоритм може не спрацювати добре, якщо дані мають складну структуру або багато викидів. Крім того, результат сильно залежить від початкової ініціалізації центрів кластерів.

### 3. Метод Головних Компонент (PCA)

**Метод головних компонент (PCA)** — це техніка для зменшення вимірності даних, що дозволяє перетворити великий набір змінних на меншу кількість нових змінних, які зберігають найбільшу частину інформації (варіації). Нові змінні, звані **головними компонентами**, є лінійними комбінаціями оригінальних змінних і направлені таким чином, щоб максимально зберегти варіацію даних.

PCA особливо корисний при роботі з великими наборами даних з багатьма змінними, де може бути складно візуалізувати або аналізувати взаємозв'язки між змінними.

#### Принцип Роботи PCA:

1. **Центрування даних**: Для початку дані централізуються, тобто кожна змінна набуває середнього значення 0.
2. **Обчислення коваріаційної матриці**: Визначається, як змінні взаємозв'язуються одна з одною.
3. **Пошук головних компонент**: Визначаються нові напрямки, які максимізують дисперсію даних.
4. **Проекція даних**: Дані проєктуються на нові головні компоненти, що дозволяє зменшити кількість вимірів, зберігаючи при цьому більшість інформації.

#### Властивості PCA:

* **Зменшення вимірності**: PCA дозволяє значно зменшити кількість змінних, що робить дані легшими для візуалізації і подальшого аналізу.
* **Збереження варіації**: Метод дозволяє зберегти більшу частину інформації про дані, навіть після зменшення вимірності.
* **Інтерпретація компонент**: Головні компоненти можуть бути інтерпретовані як комбінації оригінальних змінних, що дозволяє зрозуміти основні чинники, що впливають на дані.

### 4. Метод Ліктя (Elbow Method)

**Метод Ліктя** — це популярний метод для вибору оптимальної кількості кластерів в алгоритмі K-Means. Він ґрунтується на побудові графіку, що відображає залежність вартості функції помилки (зазвичай сума квадратів відстаней між точками і центрами кластерів) від кількості кластерів.

#### Принцип Роботи:

1. Алгоритм K-Means виконується для різних значень кількості кластерів (наприклад, від 1 до 10).
2. Для кожного значення K обчислюється сума квадратів відстаней (WCSS).
3. Будується графік WCSS від кількості кластерів.
4. Оптимальна кількість кластерів визначається як точка на графіку, де спостерігається значне зниження WCSS, після якого зміни стають незначними (це і є "ліктьова" точка).

#### Властивості:

* **Простота використання**: Легко зрозуміти і застосувати.
* **Візуальний аналіз**: Дозволяє швидко знайти оптимальне значення K за допомогою графіка.
* **Обмеження**: Іноді "ліктьова" точка може бути неявною, що ускладнює вибір K.

### 5. Аналіз Силуетів (Silhouette Analysis)

**Аналіз силуетів** є методикою для оцінки якості кластеризації, яка вимірює, наскільки добре об'єкти належать до свого кластеру, порівнюючи відстань до інших кластерів. Силуетний бал для кожного об'єкта варіюється від -1 до 1, де значення ближче до 1 вказує на добре кластеризовані об'єкти.

#### Принцип Роботи:

1. Для кожного об'єкта обчислюється середнє відстань до інших об'єктів в тому ж кластері (a(i)) та до найближчого іншого кластеру (b(i)).
2. Силуетний бал s(i)s(i)s(i) вимірюється як різниця між a(i) і b(i) в відношенні до найбільшої з цих величин.
3. Середнє значення силуетного балу для всіх об'єктів вказує на загальну якість кластеризації.

#### Властивості:

* **Оцінка якості**: Дозволяє визначити, наскільки добре були сформовані кластери.
* **Об'єктивність**: Не потребує візуальних спостережень, як у методі ліктя.
* **Універсальність**: Підходить для різних типів кластеризації та методів вимірювання відстаней.

### 6. Масштабування Даних та Виявлення Викидів

#### Масштабування Даних

Масштабування даних необхідне для того, щоб всі ознаки мали однаковий вплив на результати кластеризації. Якщо одна ознака має великі значення, це може домінувати в розрахунках відстаней. Для нормалізації використовується стандартне масштабування, яке приводить дані до середнього значення 0 та стандартного відхилення 1.

#### Виявлення Викидів

Викиди можуть сильно впливати на результати кластеризації, тому важливо виявляти та обробляти їх. Найпоширеніший метод виявлення викидів — це **метод міжквартильного розмаху (IQR)**, який дозволяє виявити значення, що виходять за межі нормального діапазону.

**Опис роботи та код програми**

### 1. **Data Preprocessing (Передобробка Даних)**

#### **a. Завантаження Даних**

На першому етапі програми ми імпортуємо набір даних, використовуючи бібліотеку **pandas**. Це дозволяє перевести дані з формату CSV у структуру **DataFrame**, що є зручним форматом для обробки та маніпуляції даними.

data = pd.read\_csv('Mall\_Customers.csv')

Після цього перевіряється коректність завантажених даних і їх структура, а також визначаються типи даних для кожної колонки.

#### **b. Очищення Даних**

На цьому етапі здійснюється перевірка на наявність **відсутніх значень** та **дублікованих рядків** в даних. Якщо виявляються дублікати, вони видаляються для уникнення спотворення результатів аналізу. Окрім того, перевіряється наявність **викидів** для кожної змінної.

1. **Виявлення відсутніх значень:**

print(data.isnull().sum())

1. **Виявлення дублікованих рядків:**

print(data.duplicated().sum())

data = data.drop\_duplicates()

1. **Обробка викидів**: Для кожної з кількісних змінних (вік, річний дохід та рейтинг витрат) обчислюються **межі для викидів** на основі **міжквартильного розмаху (IQR)**. Це дозволяє видалити значення, які значно відрізняються від основної маси даних.

Наприклад, для змінної "Age":

Q1 = data['Age'].quantile(0.25)

Q3 = data['Age'].quantile(0.75)

IQR = Q3 - Q1

age\_lower = Q1 - 1.5 \* IQR

age\_upper = Q3 + 1.5 \* IQR

data = data[(data['Age'] >= age\_lower) & (data['Age'] <= age\_upper)]

#### **c. Вибір Ознак**

Для кластеризації було вибрано три основні ознаки:

* **Вік** (Age)
* **Річний дохід** (Annual Income)
* **Рейтинг витрат** (Spending Score)

Ці ознаки мають найбільший вплив на поведінку споживачів та дозволяють здійснити точну сегментацію.

X = data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]

#### **d. Масштабування Даних**

Для коректної роботи алгоритму K-means і уникнення домінування ознак з більшими масштабами, всі дані були **нормалізовані** за допомогою **StandardScaler**.

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

Це дозволяє привести всі ознаки до однакового масштабу з середнім значенням 0 і стандартним відхиленням 1.

### 2. **Exploratory Data Analysis (EDA) (Досліджувальний Аналіз Даних)**

#### **Аналіз Розподілу Ключових Змінних**

Перш за все, здійснюється описова статистика для трьох основних змінних:

* Вік
* Річний дохід
* Рейтинг витрат

Це дозволяє отримати загальне уявлення про дані та визначити, чи є які-небудь аномалії або нерівномірні розподіли.

print(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].describe())

#### **Візуалізація Взаємозв'язків між Ознаками**

Для візуалізації розподілу кожної ознаки та взаємозв'язків між ними використовуються різні типи діаграм:

1. **Гістограми** для кожної ознаки:

sns.histplot(data['Age'], bins=10, kde=True, color='blue')

1. **Парні діаграми** (pairplot) для вивчення взаємозв'язків між кількома змінними:

sns.pairplot(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']], diag\_kind='kde')

1. **Кореляційна матриця** для аналізу зв'язків між змінними:

sns.heatmap(data[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].corr(), annot=True, cmap='coolwarm')

### 3. **Implementing K-means Clustering (Реалізація K-means Кластеризації)**

#### **Вибір Кількості Кластерів (k)**

Для вибору оптимальної кількості кластерів були використані два методи:

1. **Метод Ліктя** (Elbow Method): Графічно визначено, при якому значенні K спостерігається значне зменшення WCSS (Within-Cluster Sum of Squares).

wcss = []

K\_range = range(1, 11)

for k in K\_range:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(X\_scaled)

wcss.append(kmeans.inertia\_)

1. **Аналіз Силуетів** (Silhouette Analysis): Для кожного значення K обчислюється силуетний бал, що показує, наскільки добре об'єкти належать до своїх кластерів. Чим вищий цей бал, тим якісніше кластеризація.

silhouette\_scores = []

K\_silhouette = range(2, 11)

for k in K\_silhouette:

kmeans = KMeans(n\_clusters=k, random\_state=42)

cluster\_labels = kmeans.fit\_predict(X\_scaled)

score = silhouette\_score(X\_scaled, cluster\_labels)

silhouette\_scores.append(score)

#### **Кластеризація**

Після визначення оптимальної кількості кластерів (K=5) застосовуємо алгоритм K-means для сегментації клієнтів:

kmeans = KMeans(n\_clusters=5, random\_state=42)

clusters = kmeans.fit\_predict(X\_scaled)

data['Cluster'] = clusters

#### **Аналіз Кластерів**

Після кластеризації аналізуються середні значення кожної ознаки для кожного кластеру. Це дозволяє виявити основні характеристики кожної групи клієнтів (наприклад, середній вік, річний дохід та рейтинг витрат).

cluster\_summary = data.groupby('Cluster').agg({

'Age': 'mean',

'Annual Income (k$)': 'mean',

'Spending Score (1-100)': 'mean',

'CustomerID': 'count'

}).rename(columns={'CustomerID': 'Count'}).reset\_index()

### 4. **Evaluation of Clusters (Оцінка Якості Кластерів)**

#### **Оцінка Якості Кластерів**

Для оцінки якості кластеризації використовується **силуетний бал**. Цей бал вимірює, наскільки добре об'єкти належать до своїх кластерів.

score = silhouette\_score(X\_scaled, data['Cluster'])

Якщо середній силуетний бал високий, це вказує на хорошу кластеризацію. Чим ближчий цей бал до 1, тим краща якість кластеризації.

#### **Інтерпретація Кластерів**

Аналізуємо, що кожен кластер представляє, з точки зору бізнес-проблеми:

* Наприклад, один кластер може містити клієнтів з високим доходом і низьким рейтингом витрат, що вказує на потенціал для підвищення витрат через маркетингові кампанії.

### 5. **Visualization (Візуалізація)**

#### **Візуалізація Кластерів за Допомогою PCA**

Застосовуємо метод головних компонент (PCA) для зменшення вимірності даних і візуалізації кластерів у двовимірному просторі. Це дозволяє зрозуміти, як кластеризовані об'єкти виглядають у зменшеному просторі.

pca = PCA(n\_components=2)

principal\_components = pca.fit\_transform(X\_scaled)

pca\_df = pd.DataFrame(data=principal\_components, columns=['PC1', 'PC2'])

pca\_df['Cluster'] = data['Cluster']

sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=pca\_df, palette='Set1', alpha=0.6)

#### **Візуалізація Розподілу Ознак по Кластерам**

Для подальшого аналізу кожної ознаки по кластерам використовуються **діаграми коробок (box plots)**:

sns.boxplot(x='Cluster', y='Age', hue='Cluster', data=data, palette='Set2')

Це дає змогу наочно оцінити, як розподіляються ключові змінні між різними групами клієнтів.

**Висновок**

У рамках цієї роботи була реалізована програма для сегментації клієнтів торгового центру за допомогою алгоритму **K-means кластеризації**. Описаний процес включав кілька етапів, починаючи від завантаження та попередньої обробки даних і закінчуючи візуалізацією та аналізом отриманих кластерів. Розглянемо основні результати та висновки з роботи.

### 1. **Передобробка Даних**

На початковому етапі було здійснено **завантаження даних** з CSV файлу, а також проведена перевірка на наявність відсутніх значень та дублікованих рядків. Програма успішно справилася з видаленням дублікованих даних, що дозволило уникнути спотворення результатів кластеризації.

Після цього була виконана **обробка викидів** за допомогою методу міжквартильного розмаху (IQR). Це дозволило визначити та видалити значення, які відрізнялися від основної маси даних і могли б негативно вплинути на результат аналізу. Після очистки даних було вибрано три основні ознаки: вік, річний дохід і рейтинг витрат клієнтів, що були найбільш релевантними для задачі сегментації.

**Масштабування даних** за допомогою **StandardScaler** забезпечило рівномірний вплив усіх ознак на результат кластеризації, оскільки алгоритм K-means чутливий до масштабів змінних. Це було важливим кроком для коректної роботи алгоритму, оскільки різні діапазони значень ознак можуть домінувати у розрахунках відстаней.

### 2. **Досліджувальний Аналіз Даних (EDA)**

На етапі досліджувального аналізу даних була проведена ретельна перевірка **розподілу ключових змінних**. Були побудовані гістограми для кожної ознаки, що дозволило виявити їхні характеристики та розподіл. Зокрема, це допомогло оцінити, наскільки рівномірно розподілені дані, і визначити, чи потребують вони додаткової обробки чи трансформації.

Також було проведено **аналіз взаємозв'язків між змінними** за допомогою кореляційної матриці та парних діаграм. Це дозволило побачити, які змінні мають сильний зв'язок і які можуть бути важливими для кластеризації, а також визначити можливі мультиколінеарності.

### 3. **Реалізація K-means Кластеризації**

Одним із важливих етапів роботи була реалізація **K-means кластеризації** для сегментації клієнтів на основі їхніх характеристик. Для вибору оптимальної кількості кластерів було використано два методи:

* **Метод Ліктя** (Elbow Method): за допомогою цього методу було визначено, що оптимальна кількість кластерів для цього набору даних становить 5.
* **Аналіз Силуетів**: був використаний для додаткової оцінки якості кластеризації. Силуетний бал також підтримав вибір 5 кластерів, що забезпечує хорошу сегментацію з високою якістю групування.

Результати кластеризації показали, що існує значна різниця між різними сегментами клієнтів, що дозволяє точно націлювати маркетингові кампанії та персоналізувати пропозиції для кожної групи.

### 4. **Оцінка Якості Кластерів**

Якість кластеризації була оцінена за допомогою **силуетного балу**, який підтвердив правильність вибору кількості кластерів. Середній силуетний бал був високим, що свідчить про хорошу роздільність кластерів і правильне призначення об'єктів до відповідних груп.

### 5. **Візуалізація Результатів**

Для візуалізації результатів кластеризації була використана **PCA (метод головних компонент)** для зменшення вимірності даних до двох компонентів. Це дозволило наочно зобразити кластеризацію в двовимірному просторі та оцінити, як кожен кластер виглядає відносно інших.

Також для кожного з кластерів були побудовані **діаграми коробок** (box plots), що дозволило наочно оцінити розподіл ключових характеристик (вік, річний дохід та рейтинг витрат) у межах кожного кластера.

### 6. **Інтерпретація Кластерів**

Після кластеризації було здійснено детальний **аналіз характеристик кожного кластера**. Це дозволило виділити основні групи клієнтів за їхніми характеристиками:

* Кластери з високим рівнем доходу та високим рейтингом витрат, які можуть бути цікавими для програм лояльності та маркетингових акцій.
* Кластери з низьким рівнем доходу та середнім рейтингом витрат, що можуть потребувати спеціальних пропозицій для залучення.

### Загальні Висновки

1. **Сегментація клієнтів** на основі даних про вік, річний дохід та рейтинг витрат дозволяє точно виділити різні групи, що значно покращує стратегії взаємодії з клієнтами.
2. Програма успішно реалізувала етапи передобробки даних, включаючи очищення, нормалізацію та вибір ознак для кластеризації.
3. Результати кластеризації підтвердили гіпотезу про існування кількох значних груп клієнтів, що дозволяє підприємству здійснювати більш таргетовані маркетингові кампанії.
4. Візуалізація результатів за допомогою PCA та діаграм коробок дала змогу зрозуміти структуру кластерів і їх відмінності по ключових ознаках.

**ДОДАТОК**

*# Import Libraries*import os *# Added for directory operations*import pandas as *pd*import numpy as *np*import matplotlib.pyplot as *plt*import seaborn as *sns*from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
from sklearn.decomposition import PCA  
  
*# Ensure the 'images' directory exists*if not os.path.exists('images'):  
 os.makedirs('images')  
  
*# 1. Data Preprocessing  
  
# a. Loading Data  
# Read the dataset from a CSV file into a pandas DataFrame  
data* = pd.read\_csv('Mall\_Customers.csv')  
  
*# b. Data Cleaning  
  
# Display the number of missing values per column to identify any incomplete data*print("Missing values per column:")  
print(*data*.isnull().sum())  
  
*# Display the number of duplicate rows to ensure data uniqueness*print("\nNumber of duplicate rows:", *data*.duplicated().sum())  
  
*# Remove duplicate rows to prevent bias in analysis  
data* = *data*.drop\_duplicates()  
  
*# Outlier detection and removal for 'Age'  
# Calculate the first (Q1) and third (Q3) quartiles  
Q1* = *data*['Age'].quantile(0.25)  
*Q3* = *data*['Age'].quantile(0.75)  
*IQR* = *Q3* - *Q1 # Interquartile Range  
  
# Define the lower and upper bounds for outliers  
age\_lower* = *Q1* - 1.5 \* *IQR  
age\_upper* = *Q3* + 1.5 \* *IQR  
  
# Filter the data to remove outliers in the 'Age' column  
data* = *data*[(*data*['Age'] >= *age\_lower*) & (*data*['Age'] <= *age\_upper*)]  
  
*# Similarly, outlier detection and removal for 'Annual Income (k$)'  
Q1* = *data*['Annual Income (k$)'].quantile(0.25)  
*Q3* = *data*['Annual Income (k$)'].quantile(0.75)  
*IQR* = *Q3* - *Q1  
income\_lower* = *Q1* - 1.5 \* *IQR  
income\_upper* = *Q3* + 1.5 \* *IQR  
data* = *data*[(*data*['Annual Income (k$)'] >= *income\_lower*) & (*data*['Annual Income (k$)'] <= *income\_upper*)]  
  
*# Similarly, outlier detection and removal for 'Spending Score (1-100)'  
Q1* = *data*['Spending Score (1-100)'].quantile(0.25)  
*Q3* = *data*['Spending Score (1-100)'].quantile(0.75)  
*IQR* = *Q3* - *Q1  
spending\_lower* = *Q1* - 1.5 \* *IQR  
spending\_upper* = *Q3* + 1.5 \* *IQR  
data* = *data*[(*data*['Spending Score (1-100)'] >= *spending\_lower*) & (*data*['Spending Score (1-100)'] <= *spending\_upper*)]  
  
*# c. Feature Selection  
# Select relevant features for clustering  
X* = *data*[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]  
  
*# d. Data Normalization  
# Scale the features to have zero mean and unit variance  
scaler* = StandardScaler()  
*X\_scaled* = *scaler*.fit\_transform(*X*)  
  
*# 2. Exploratory Data Analysis (EDA)  
  
# Descriptive Statistics  
# Provide summary statistics for the selected features*print("\nDescriptive Statistics:")  
print(*data*[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].describe())  
  
*# Separate Histograms  
  
# Histogram for Age*plt.figure(figsize=(8,6))  
sns.histplot(*data*['Age'], bins=10, kde=True, color='blue')  
plt.title('Гістограма Віку') *# Title in Ukrainian: Age Histogram*plt.xlabel('Вік') *# X-axis label: Age*plt.ylabel('Частота') *# Y-axis label: Frequency*plt.savefig('images/hist\_age.png') *# Save the Age histogram*plt.show() *# Display the plot  
  
# Histogram for Annual Income*plt.figure(figsize=(8,6))  
sns.histplot(*data*['Annual Income (k$)'], bins=10, kde=True, color='green')  
plt.title('Гістограма Річного Доходу') *# Annual Income Histogram*plt.xlabel('Річний Дохід (k$)') *# Annual Income (k$)*plt.ylabel('Частота') *# Frequency*plt.savefig('images/hist\_income.png') *# Save the Annual Income histogram*plt.show()  
  
*# Histogram for Spending Score*plt.figure(figsize=(8,6))  
sns.histplot(*data*['Spending Score (1-100)'], bins=10, kde=True, color='red')  
plt.title('Гістограма Рейтингу Витрат') *# Spending Score Histogram*plt.xlabel('Рейтинг Витрат (1-100)') *# Spending Score (1-100)*plt.ylabel('Частота') *# Frequency*plt.savefig('images/hist\_spending.png') *# Save the Spending Score histogram*plt.show()  
  
*# Pairplot  
# Visualize pairwise relationships between features  
pairplot\_fig* = sns.pairplot(*data*[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']], diag\_kind='kde')  
*pairplot\_fig*.savefig('images/pairplot.png') *# Save the pairplot*plt.show()  
  
*# Correlation Matrix  
# Compute and visualize the correlation matrix to understand relationships between features*plt.figure(figsize=(8,6))  
*corr* = *data*[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']].corr()  
sns.heatmap(*corr*, annot=True, cmap='coolwarm')  
plt.title('Correlation Matrix') *# Title for the correlation matrix*plt.savefig('images/correlation\_matrix.png') *# Save the correlation matrix plot*plt.show()  
  
*# 3. Implementing K-Means Clustering  
  
# a. Choosing the Number of Clusters (k) - Elbow Method  
# The Elbow Method helps determine the optimal number of clusters by plotting WCSS against different k values  
wcss* = [] *# List to store Within-Cluster Sum of Squares  
K\_range* = range(1, 11) *# Testing k from 1 to 10*for *k\_val* in *K\_range*:  
 *kmeans* = KMeans(n\_clusters=*k\_val*, random\_state=42) *# Initialize KMeans with current k  
 kmeans*.fit(*X\_scaled*) *# Fit KMeans to the scaled data  
 wcss*.append(*kmeans*.inertia\_) *# Append the WCSS to the list  
  
# Plot the Elbow Method graph*plt.figure(figsize=(8,5))  
plt.plot(*K\_range*, *wcss*, 'bo-') *# Plot k vs WCSS*plt.xlabel('Number of Clusters (k)') *# X-axis label*plt.ylabel('WCSS') *# Y-axis label*plt.title('Elbow Method for Optimal k') *# Title of the plot*plt.savefig('images/elbow\_method.png') *# Save the Elbow Method plot*plt.show() *# Display the plot  
  
# a. Choosing the Number of Clusters (k) - Silhouette Analysis  
# Silhouette Analysis measures how similar an object is to its own cluster compared to other clusters  
silhouette\_scores* = [] *# List to store silhouette scores  
K\_silhouette* = range(2, 11) *# Silhouette score is not defined for k=1*for *k\_val* in *K\_silhouette*:  
 *kmeans* = KMeans(n\_clusters=*k\_val*, random\_state=42) *# Initialize KMeans with current k  
 cluster\_labels* = *kmeans*.fit\_predict(*X\_scaled*) *# Fit KMeans and predict cluster labels  
 score* = silhouette\_score(*X\_scaled*, *cluster\_labels*) *# Calculate silhouette score  
 silhouette\_scores*.append(*score*) *# Append the score to the list  
  
# Plot the Silhouette Analysis graph*plt.figure(figsize=(8,5))  
plt.plot(*K\_silhouette*, *silhouette\_scores*, 'bo-') *# Plot k vs Silhouette Score*plt.xlabel('Number of Clusters (k)') *# X-axis label*plt.ylabel('Silhouette Score') *# Y-axis label*plt.title('Silhouette Analysis for Optimal k') *# Title of the plot*plt.savefig('images/silhouette\_analysis.png') *# Save the Silhouette Analysis plot*plt.show() *# Display the plot  
  
# Based on Elbow and Silhouette, choose k=5  
k* = 5 *# Optimal number of clusters determined from the plots  
kmeans* = KMeans(n\_clusters=*k*, random\_state=42) *# Initialize KMeans with k=5  
clusters* = *kmeans*.fit\_predict(*X\_scaled*) *# Fit KMeans and predict cluster labels  
data*['Cluster'] = *clusters # Assign cluster labels to the original data  
  
# b. Cluster Analysis  
# Summarize the characteristics of each cluster  
cluster\_summary* = *data*.groupby('Cluster').agg({  
 'Age': 'mean', *# Average age in each cluster* 'Annual Income (k$)': 'mean', *# Average annual income in each cluster* 'Spending Score (1-100)': 'mean', *# Average spending score in each cluster* 'CustomerID': 'count' *# Number of customers in each cluster*}).rename(columns={'CustomerID': 'Count'}).reset\_index()  
  
*# Display the cluster summary*print("\nCluster Summary:")  
print(*cluster\_summary*)  
  
*# 4. Evaluation of Clusters  
# Calculate the overall silhouette score to evaluate clustering quality  
score* = silhouette\_score(*X\_scaled*, *data*['Cluster'])  
print(f'\nSilhouette Score for k={*k*}: {*score*:.4f}') *# Display the silhouette score  
  
# 5. Visualization  
  
# a. PCA for Dimensionality Reduction  
# Reduce the data to 2 dimensions for visualization purposes  
pca* = PCA(n\_components=2) *# Initialize PCA to reduce to 2 principal components  
principal\_components* = *pca*.fit\_transform(*X\_scaled*) *# Fit PCA and transform the data  
pca\_df* = pd.DataFrame(data=*principal\_components*, columns=['PC1', 'PC2']) *# Create a DataFrame for PCA results  
pca\_df*['Cluster'] = *data*['Cluster'] *# Add cluster labels to the PCA DataFrame  
  
# b. Scatter Plot of Clusters  
# Visualize the clusters in the PCA-reduced feature space*plt.figure(figsize=(8,6))  
*scatter\_fig* = sns.scatterplot(  
 x='PC1',  
 y='PC2',  
 hue='Cluster',  
 data=*pca\_df*,  
 palette='Set1',  
 alpha=0.6  
) *# Create a scatter plot with clusters colored differently*plt.title('Customer Segments (PCA Reduced)') *# Title of the plot*plt.legend(title='Cluster') *# Add legend with title*plt.savefig('images/pca\_clusters.png') *# Save the PCA scatter plot*plt.show() *# Display the plot  
  
# c. Cluster Centroids in Original Scale  
# Transform the cluster centers back to the original scale for interpretation  
centroids* = *scaler*.inverse\_transform(*kmeans*.cluster\_centers\_) *# Inverse transform to original feature scale  
centroid\_df* = pd.DataFrame(*centroids*, columns=['Age', 'Annual Income (k$)', 'Spending Score (1-100)']) *# Create DataFrame for centroids*print("\nCluster Centroids (Original Scale):")  
print(*centroid\_df*) *# Display the centroids  
  
# d. Additional Visualizations  
  
# Boxplot for Age by Cluster*plt.figure(figsize=(8,6))  
sns.boxplot(  
 x='Cluster',  
 y='Age',  
 hue='Cluster',  
 data=*data*,  
 palette='Set2'  
) *# Create a boxplot for Age distribution across clusters*plt.title('Age Distribution by Cluster') *# Title of the plot*plt.legend([], [], frameon=False) *# Disable the legend as hue is redundant with x-axis*plt.savefig('images/boxplot\_age.png') *# Save the boxplot*plt.show() *# Display the plot  
  
# Boxplot for Annual Income by Cluster*plt.figure(figsize=(8,6))  
sns.boxplot(  
 x='Cluster',  
 y='Annual Income (k$)',  
 hue='Cluster',  
 data=*data*,  
 palette='Set2'  
) *# Create a boxplot for Annual Income distribution across clusters*plt.title('Annual Income Distribution by Cluster') *# Title of the plot*plt.legend([], [], frameon=False) *# Disable the legend as hue is redundant with x-axis*plt.savefig('images/boxplot\_income.png') *# Save the boxplot*plt.show() *# Display the plot  
  
# Boxplot for Spending Score by Cluster*plt.figure(figsize=(8,6))  
sns.boxplot(  
 x='Cluster',  
 y='Spending Score (1-100)',  
 hue='Cluster',  
 data=*data*,  
 palette='Set2'  
) *# Create a boxplot for Spending Score distribution across clusters*plt.title('Spending Score Distribution by Cluster') *# Title of the plot*plt.legend([], [], frameon=False) *# Disable the legend as hue is redundant with x-axis*plt.savefig('images/boxplot\_spending.png') *# Save the boxplot*plt.show() *# Display the plot*

**Джерела**

1. **Hastie, T., Tibshirani, R., & Friedman, J. (2009).** The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
2. **James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013).** An Introduction to Statistical Learning: with Applications in R. Springer.
3. **Kaufman, L., & Rousseeuw, P. J. (2009).** Finding Groups in Data: An Introduction to Cluster Analysis. Wiley-Interscience.
4. **Shlens, J. (2014).** A Tutorial on Principal Component Analysis. arXiv.
5. **Lloyd, S. (1982).** Least Squares Quantization in PCM. IEEE Transactions on Information Theory, 28(2), 129-137.
6. **Jolliffe, I. T. (2002).** Principal Component Analysis. Springer Series in Statistics.
7. **Steinhaus, H. (1956).** Sur la division des corps matériels en parties. Bulletin of the Polish Academy of Sciences. Mathematics, 4(12), 804-812.
8. **Scikit-learn Documentation.** (n.d.). Clustering: K-means. Retrieved from https://scikit-learn.org/stable/modules/clustering.html#k-means
9. **Python for Data Analysis** by Wes McKinney (2017). O'Reilly Media
10. **Matplotlib Documentation.** (n.d.). Visualization with Matplotlib. Retrieved from https://matplotlib.org/stable/contents.html