**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет комп’ютерних наук та кібернетики

Кафедра теорії та технології програмування

**Звіт**

до лабораторної роботи № 3

Виконав студент 4-го курсу

Групи ТТП-42

Ходаков Максим Олегович

Київ – 2025

**Короткий опис області**

Даний дата сет візуалізує список книжок наявних на книжковій платформі, разом з рейтингом та кількістю рецензій. Також для кожної книжки наявні автор, мова написання, дата випуску та видавництва.

**DataSet**

Посилання: <https://www.kaggle.com/datasets/bahramjannesarr/goodreads-book-datasets-10m>

Приблизно 10 000 000 книг доступні в архівах сайту, і ці набори даних збираються з них. для запитів через API ми використовували бібліотеку python Goodreads, набори даних оновлюватимуться кожні 2 дні.

Ці набори даних дуже добре підходять для виконання двох завдань:

1 . Створення системи рекомендацій книг на основі 10 млн книг

2 . Використання колонок опису для НЛП

Репозиторій на Github

Посилання на проект на github: <https://github.com/BahramJannesar/GoodreadsBookDataset>

Зміст

Приблизно 10 000 000 книг доступні в архівах сайту, і ці набори даних збираються з них. для запитів через API ми використовували бібліотеку python Goodreads, набори даних оновлюватимуться кожні 2 дні.

Приклади даних books в JSON:

{

"Id": "5107",

"Name": "The Catcher in the Rye",

"RatingDist1": "1:133165",

"RatingDist2": "2:224884",

"RatingDist3": "3:553476",

"RatingDist4": "4:808278",

"RatingDist5": "5:891037",

"pagesNumber": 277,

"RatingDistTotal": "total:2610840",

"PublishMonth": 30,

"PublishDay": 1,

"Publisher": "Back Bay Books",

"CountsOfReview": 44046,

"PublishYear": 2001,

"Language": "eng",

"Authors": "J.D. Salinger",

"Rating": 3.8,

"ISBN": "0316769177",

"Count of text reviews": 55539,

"Description": "The hero-narrator of The Catcher in the Rye is an ancient child of sixteen, a native New Yorker named Holden Caulfield. Through

circumstances that tend to preclude adult, secondhand description, he leaves his prep school in Pennsylvania and goes underground in New York City for

three days. "

}

Кількість файлів при завантажені архіву:

A screenshot of a computer

Description automatically generated

Приклади завантажених .CSV

Books:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **RatingDist1** | **pagesNumber** | **RatingDist4** | **RatingDistTotal** | **PublishMonth** | **PublishDay** | **Publisher** | **CountsOfReview** | **PublishYear** | **Language** | **Authors** | **Rating** | **RatingDist2** | **RatingDist5** | **ISBN** | **RatingDist3** |
| Harry Potter and the Half-Blood Prince (Harry Potter, #6) | 1:9896 | 652 | 4:556485 | total:2298124 | 16 | 9 | Scholastic Inc. | 28062 | 2006 | eng | J.K. Rowling | 4.57 | 2:25317 | 5:1546466 |  | 3:159960 |

UserRating:

A screen shot of a computer screen

Description automatically generated

Основні поля для роботи з книгами:

Name – назва книги

RatingDist1 – кількість оцінок 1 для книги

RatingDist2 – кількість оцінок 2 для книги

RatingDist3 – кількість оцінок 3 для книги

RatingDist4 – кількість оцінок 4 для книги

RatingDist5 – кількість оцінок 5 для книги

PublishYear – рік випуску

PublishMonth – місяць випуску

PublishDay – день випуску

CountsOfReview – кількість рецензій

Publisher – видавець

Language – мова книги

Authors – автори

**Task 1**

Прогнозування рейтингу за допомогою лінійної регресії

### Лінійна регресія

Лінійна регресія — це метод прогнозування, який використовується для передбачення значень залежної змінної на основі незалежних змінних. Вона передбачає наявність лінійного зв'язку між незалежними змінними (в даному випадку, характеристиками книги) та залежною змінною (рейтинг).

Задача лінійної регресії полягає в тому, щоб знайти ці коефіцієнти таким чином, щоб мінімізувати помилку прогнозу, яку зазвичай вимірюють за допомогою метрики середньоквадратичної помилки (MSE).

### Метрики оцінки моделі

1. **R² (Коефіцієнт детермінації):** R² — це метрика, яка вимірює, яку частину змінності залежної змінної можна пояснити змінними незалежними змінними. Значення R² коливається від 0 до 1, де значення близьке до 1 означає, що модель добре пояснює варіативність у даних.
2. **Mean Squared Error (MSE):** Це метрика, яка вимірює середнє квадратичне відхилення між передбаченими значеннями та фактичними. Менше значення MSE свідчить про кращу модель.
3. **Mean Absolute Error (MAE):** Метріка, яка вимірює середнє абсолютне відхилення прогнозованих значень від фактичних.

## Підготовка даних

### 1. Завантаження та очищення даних

Ми завантажили набір даних за допомогою бібліотеки Pandas з CSV-файлу за допомогою функції pd.read\_csv. Використано роздільник ;, оскільки дані зберігаються у форматі CSV з таким роздільником.

### 2. Обробка пропущених значень

У процесі роботи з даними деякі значення можуть бути відсутні. У нашому випадку ми використовуємо SimpleImputer для заповнення відсутніх значень у числових колонках середнім значенням. Для категоріальних змінних ми заміняємо пропущені значення на найбільш часті.

### 3. Перетворення категоріальних змінних

Категоріальні змінні, такі як мова книги, були перетворені в числові за допомогою техніки **one-hot encoding**. Це дозволяє перетворити кожне категоріальне значення на окрему бінарну змінну.

## Алгоритм побудови моделі

1. **Розподіл даних:** Для тренування моделі ми розділили дані на тренувальну та тестову вибірки за допомогою функції train\_test\_split з параметром test\_size=0.2, що означає, що 20% даних було відкладено для тестування моделі.
2. **Побудова моделі лінійної регресії:** За допомогою класу LinearRegression з бібліотеки sklearn.linear\_model була побудована модель, яка тренувалась на тренувальних даних.
3. **Оцінка моделі:** Для оцінки моделі використовували метрики **R²** та **Mean Squared Error** (MSE).
4. **Прогнозування:** Після тренування моделі, ми використовуємо її для прогнозування рейтингу книги на основі введених користувачем даних.

**Код програми**

import pandas as *pd*import numpy as *np*import matplotlib.pyplot as *plt*from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import OneHotEncoder  
import string  
  
  
*# --- Допоміжні функції ---*def extract\_letter\_features(*title*):  
 *"""  
 Для заданої назви книги повертаємо словник із ознаками:  
 для кожної літери англійського алфавіту (a-z) 1, якщо вона зустрічається в назві, або 0, якщо ні.  
 """  
 title* = str(*title*).lower()  
 *features* = {}  
 for *letter* in string.ascii\_lowercase:  
 *features*[f'letter\_{*letter*}'] = 1 if *letter* in *title* else 0  
 return *features  
  
  
# Функція для отримання введених даних від користувача*def get\_user\_input():  
 print("Введіть дані для прогнозування рейтингу книги:")  
 *publish\_year* = int(input("Введіть рік публікації: "))  
 *language* = input("Введіть мову (наприклад, 'eng'): ")  
 *title* = input("Введіть назву книги: ")  
 *# Отримуємо ознаки з назви  
 letter\_features* = extract\_letter\_features(*title*)  
 return *publish\_year*, *language*, *letter\_features  
  
  
# --- Завантаження та підготовка даних ---  
  
# Завантаження даних з CSV; роздільник ';'  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Прибираємо зайві пробіли в назвах колонок, якщо потрібно  
df*.columns = [*col*.strip() for *col* in *df*.columns]  
  
*# Замінюємо кому на крапку в колонці 'Rating' та переводимо у тип float  
df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
  
*# Видаляємо записи з пропущеним рейтингом  
df* = *df*.dropna(subset=['Rating'])  
  
*# Використовуємо наступні ознаки для прогнозу:  
# - PublishYear (рік публікації)  
# - Language (мова)  
# - Літери, присутні в назві (Name)  
# Для цього спочатку створимо ознаки з назви книги.  
letter\_features\_df* = *df*['Name'].apply(extract\_letter\_features).apply(pd.Series)  
  
*# Об'єднуємо оригінальний DataFrame з ознаками з назви  
df* = pd.concat([*df*, *letter\_features\_df*], axis=1)  
  
*# Підготовка числових ознак: PublishYear  
features\_numeric* = ['PublishYear']  
*df*[*features\_numeric*] = *df*[*features\_numeric*].apply(pd.to\_numeric, errors='coerce')  
*imputer* = SimpleImputer(strategy='mean')  
*df*[*features\_numeric*] = *imputer*.fit\_transform(*df*[*features\_numeric*])  
  
*# One-hot кодування для колонки 'Language'  
encoder* = OneHotEncoder(sparse\_output=False)  
*language\_encoded* = *encoder*.fit\_transform(*df*[['Language']])  
*language\_encoded\_df* = pd.DataFrame(*language\_encoded*, columns=*encoder*.get\_feature\_names\_out(['Language']))  
*df* = pd.concat([*df*, *language\_encoded\_df*], axis=1)  
  
*# Остаточний набір ознак: PublishYear + закодована мова + ознаки з назви (letter\_a ... letter\_z)  
feature\_cols* = *features\_numeric* + list(*language\_encoded\_df*.columns) + [*col* for *col* in *df*.columns if  
 *col*.startswith('letter\_')]  
*X* = *df*[*feature\_cols*]  
*y* = *df*['Rating']  
  
*# Видаляємо можливі NaN (хоча зазвичай їх не має після імпутації)  
X* = *X*.dropna()  
*y* = *y*[*X*.index]  
  
*# Розподіл даних на тренувальну та тестову вибірки  
X\_train*, *X\_test*, *y\_train*, *y\_test* = train\_test\_split(*X*, *y*, test\_size=0.2, random\_state=42)  
  
*# --- Тренування моделі ---  
model* = LinearRegression()  
*model*.fit(*X\_train*, *y\_train*)  
  
*# Оцінка моделі  
y\_pred* = *model*.predict(*X\_test*)  
*r2* = r2\_score(*y\_test*, *y\_pred*)  
*mse* = mean\_squared\_error(*y\_test*, *y\_pred*)  
print(f"Linear Regression: R2 Score = {*r2*:.2f}, Mean Squared Error = {*mse*:.2f}")  
  
*# --- Business Value Explanation ---*print("\nBusiness Value:")  
print("Ця модель прогнозує рейтинг книги на основі наступних чинників:")  
print(" - Рік публікації: дозволяє врахувати тренди та зміни вподобань аудиторії з часом.")  
print(" - Мова: різні ринки можуть мати різне сприйняття книг.")  
print(" - Літери в назві: аналізує, чи впливає стиль оформлення назви (наявність певних літер) на рейтинг.")  
print("\nВидавці можуть використовувати цей аналіз для оптимізації назв книг, адаптуючи їх під цільову аудиторію,")  
print("а також для прогнозування потенційної успішності книги перед її виходом на ринок.\n")  
  
*# --- Отримання введених даних від користувача та прогноз ---  
publish\_year*, *language*, *user\_letter\_features* = get\_user\_input()  
  
*# Формуємо DataFrame для введених даних  
user\_data* = pd.DataFrame({  
 'PublishYear': [*publish\_year*]  
})  
  
*# One-hot кодування введеної мови (маємо використовувати вже навчений encoder)  
user\_language\_encoded* = *encoder*.transform([[*language*]])  
*user\_language\_df* = pd.DataFrame(*user\_language\_encoded*, columns=*encoder*.get\_feature\_names\_out(['Language']))  
  
*# Перетворення ознак з назви (вже отриманий словник) у DataFrame  
user\_letter\_df* = pd.DataFrame([*user\_letter\_features*])  
  
*# Об'єднуємо введені дані з усіма необхідними ознаками  
user\_input\_final* = pd.concat([*user\_data*, *user\_language\_df*, *user\_letter\_df*], axis=1)  
  
*# Переконуємося, що всі колонки присутні у потрібному порядку*for *col* in *X*.columns:  
 if *col* not in *user\_input\_final*.columns:  
 *user\_input\_final*[*col*] = 0  
*user\_input\_final* = *user\_input\_final*[*X*.columns]  
  
*# Прогноз  
prediction* = *model*.predict(*user\_input\_final*)  
print(f"\nПрогнозований рейтинг книги: {*prediction*[0]:.2f}")  
  
*# --- Візуалізація результатів ---*plt.figure(figsize=(8, 6))  
plt.scatter(*y\_test*, *y\_pred*, color='blue', alpha=0.6)  
plt.plot([*y*.min(), *y*.max()], [*y*.min(), *y*.max()], color='red', linestyle='--')  
plt.title('Фактичні vs Прогнозовані Рейтинги')  
plt.xlabel('Фактичні Рейтинги')  
plt.ylabel('Прогнозовані Рейтинги')  
plt.show()

### Опис програми

### Детальний опис роботи коду

1. **Імпорт бібліотек та налаштування:**
   * Імпортуються основні бібліотеки: pandas для роботи з даними, numpy для числових операцій, matplotlib для побудови графіків, sklearn для побудови моделі машинного навчання, а також string для роботи з символами.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import OneHotEncoder

import string

1. **Функція extract\_letter\_features:**
   * Ця функція приймає назву книги (типу рядок), переводить її у нижній регістр та перевіряє для кожної літери англійського алфавіту, чи присутня вона у назві.
   * Для кожної літери створюється ознака (наприклад, letter\_a, letter\_b тощо), яка набуває значення 1, якщо літера є, або 0, якщо немає.

def extract\_letter\_features(title):

title = str(title).lower()

features = {}

for letter in string.ascii\_lowercase:

features[f'letter\_{letter}'] = 1 if letter in title else 0

return features

1. **Функція get\_user\_input:**
   * Функція запитує у користувача дані для прогнозу:
     + Рік публікації (PublishYear),
     + Мову (Language),
     + Назву книги (для подальшого виділення ознак з літер).
   * Функція повертає рік, мову та словник із ознаками, отриманими за допомогою extract\_letter\_features.

def get\_user\_input():

print("Введіть дані для прогнозування рейтингу книги:")

publish\_year = int(input("Введіть рік публікації: "))

language = input("Введіть мову (наприклад, 'eng'): ")

title = input("Введіть назву книги: ")

letter\_features = extract\_letter\_features(title)

return publish\_year, language, letter\_features

1. **Завантаження та підготовка даних:**
   * Дані завантажуються з CSV-файлу, де роздільником є ;.
   * Назви стовпців обрізаються від зайвих пробілів.
   * Колонка Rating (рейтинг книги) очищається від ком (замінюється на крапку) та приводиться до типу float.
   * Записи з пропущеним рейтингом видаляються.

df = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')

df.columns = [col.strip() for col in df.columns]

df['Rating'] = df['Rating'].replace({',': '.'}, regex=True).astype(float)

df = df.dropna(subset=['Rating'])

1. **Створення ознак на основі назви книги:**
   * Для кожного запису (назви книги) викликається функція extract\_letter\_features, що створює новий DataFrame із стовпцями letter\_a ... letter\_z.
   * Цей DataFrame об'єднується з основним, щоб кожен запис містив додаткові ознаки з назви.

letter\_features\_df = df['Name'].apply(extract\_letter\_features).apply(pd.Series)

df = pd.concat([df, letter\_features\_df], axis=1)

1. **Підготовка числових ознак та імпутація:**
   * В якості числової ознаки використовується PublishYear. Переконуємося, що дані числові, і заповнюємо можливі пропуски за допомогою SimpleImputer (стратегія – середнє значення).

features\_numeric = ['PublishYear']

df[features\_numeric] = df[features\_numeric].apply(pd.to\_numeric, errors='coerce')

imputer = SimpleImputer(strategy='mean')

df[features\_numeric] = imputer.fit\_transform(df[features\_numeric])

1. **One-hot кодування мови:**
   * Колонка Language кодується за допомогою OneHotEncoder.
   * Отримані ознаки додаються до DataFrame.

encoder = OneHotEncoder(sparse\_output=False)

language\_encoded = encoder.fit\_transform(df[['Language']])

language\_encoded\_df = pd.DataFrame(language\_encoded, columns=encoder.get\_feature\_names\_out(['Language']))

df = pd.concat([df, language\_encoded\_df], axis=1)

1. **Формування фінального набору ознак для моделі:**
   * Остаточний набір ознак включає:
     + Числова ознака PublishYear.
     + Всі колонки, отримані з one-hot кодування Language (наприклад, Language\_eng тощо).
     + Усі ознаки, які починаються на letter\_ (ознаки з назви книги).
   * Цей набір записується у змінну X, а цільова змінна Rating – у y.

feature\_cols = features\_numeric + list(language\_encoded\_df.columns) + [col for col in df.columns if col.startswith('letter\_')]

X = df[feature\_cols]

y = df['Rating']

X = X.dropna()

y = y[X.index]

1. **Розподіл даних на тренувальну та тестову вибірки та тренування моделі:**
   * Дані діляться на тренувальну та тестову вибірки за допомогою train\_test\_split.
   * Тренується модель лінійної регресії (LinearRegression) на тренувальних даних.
   * Після тренування здійснюється оцінка моделі за допомогою R² та MSE (середньоквадратична помилка).

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Linear Regression: R2 Score = {r2:.2f}, Mean Squared Error = {mse:.2f}")

1. **Пояснення бізнес-цінності:**
   * В консолі виводяться пояснення, як модель може бути корисною для бізнесу:
     + Прогноз рейтингу на основі року публікації дає змогу врахувати тренди.
     + Мова книги впливає на сприйняття в різних ринках.
     + Аналіз літер у назві допомагає виявити, чи впливають стилістичні особливості на успішність книги.

print("\nBusiness Value:")

print("Ця модель прогнозує рейтинг книги на основі наступних чинників:")

print(" - Рік публікації: дозволяє врахувати тренди та зміни вподобань аудиторії з часом.")

print(" - Мова: різні ринки можуть мати різне сприйняття книг.")

print(" - Літери в назві: аналізує, чи впливає стиль оформлення назви (наявність певних літер) на рейтинг.")

print("\nВидавці можуть використовувати цей аналіз для оптимізації назв книг, адаптуючи їх під цільову аудиторію,")

print("а також для прогнозування потенційної успішності книги перед її виходом на ринок.\n")

1. **Отримання даних від користувача та прогноз:**
   * Викликається функція get\_user\_input(), яка запитує рік публікації, мову та назву книги.
   * Для введеної назви книги обчислюються ознаки (належність літер).
   * Формується DataFrame з даними користувача, проводиться one-hot кодування мови, та дані комбінуються таким чином, щоб порядок колонок відповідав набору ознак, на якому тренувалася модель.
   * Модель робить прогноз, і прогнозований рейтинг виводиться у консоль.

publish\_year, language, user\_letter\_features = get\_user\_input()

user\_data = pd.DataFrame({

'PublishYear': [publish\_year]

})

user\_language\_encoded = encoder.transform([[language]])

user\_language\_df = pd.DataFrame(user\_language\_encoded, columns=encoder.get\_feature\_names\_out(['Language']))

user\_letter\_df = pd.DataFrame([user\_letter\_features])

user\_input\_final = pd.concat([user\_data, user\_language\_df, user\_letter\_df], axis=1)

for col in X.columns:

if col not in user\_input\_final.columns:

user\_input\_final[col] = 0

user\_input\_final = user\_input\_final[X.columns]

prediction = model.predict(user\_input\_final)

print(f"\nПрогнозований рейтинг книги: {prediction[0]:.2f}")

1. **Візуалізація результатів:**
   * Створюється графік, який показує фактичні значення рейтингу з тестової вибірки та прогнозовані значення.
   * Діагональна лінія (червона, пунктирна) показує ідеальний сценарій (коли прогноз рівний фактичному значенню).

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='blue', alpha=0.6)

plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--')

plt.title('Фактичні vs Прогнозовані Рейтинги')

plt.xlabel('Фактичні Рейтинги')

plt.ylabel('Прогнозовані Рейтинги')

plt.show()

### Резюме

* **Підготовка даних:** Дані завантажуються, обробляються (очищуються, проводиться імпутація, one-hot кодування), а також додаються ознаки, що відображають наявність літер у назві.
* **Тренування моделі:** На основі обраних ознак тренується модель лінійної регресії, яка прогнозує рейтинг книги.
* **Бізнес-цінність:** Прогноз дозволяє приймати обґрунтовані рішення стосовно маркетингу, називання книг та розподілу ресурсів.
* **Взаємодія з користувачем:** Користувач вводить дані, після чого модель прогнозує рейтинг, що дає змогу отримати швидку зворотний зв’язок.

**Лістинг та графіки**

**A screenshot of a computer code

Description automatically generated**

Рис.1. Лістинг програми

A graph with a blue dot

Description automatically generated with medium confidence

Рис.2. Розподіл рейтингів

**Task 2**

Класифікація рейтингу

## 1. Завантаження та попередня обробка даних

* **Завантаження даних:**  
  Програма завантажує дані з CSV-файлу CSV\_BI\_Lab1\_data\_source.csv, де використовується роздільник ;. Дані містять інформацію про книги, включно з назвою, розподілом оцінок (RatingDist1 до RatingDist5), кількістю відгуків, загальним рейтингом та іншими атрибутами.
* **Обробка колонки Rating:**  
  Для стовпця Rating виконуються наступні операції:
  + Замінюються коми (,) на крапки (.) – це необхідно, оскільки у деяких форматах числові значення можуть містити коми як десятковий роздільник.
  + Значення перетворюються до типу float.
* **Видалення записів з пропущеним рейтингом:**  
  Всі записи, де значення в стовпці Rating відсутні (NaN), видаляються із датасету. Це забезпечує коректне навчання моделей без втручання пропущених даних.

## 2. Створення цільової змінної – RatingClass

* **Цільова змінна:**  
  На основі значення рейтингу створюється новий стовпець RatingClass, який класифікує книги на дві категорії:
  + **High:** Якщо Rating ≥ 4.5.
  + **Low:** Якщо Rating < 4.5.
* **Бізнес-цінність:**  
  Такий поділ допомагає визначити, які книги можуть вважатися успішними (мають високий рейтинг), а які – потенційно менш успішними. Це важливо для видавництв та маркетологів, оскільки дозволяє спрямувати зусилля і ресурси на просування перспективних проектів.

## 3. Підготовка ознак для моделювання

* **Вибір ознак:**  
  Для побудови моделей використовуються наступні ознаки:
  + Розподіл оцінок: RatingDist1, RatingDist2, RatingDist3, RatingDist4, RatingDist5.
  + Загальна кількість відгуків: CountsOfReview.
* **Заповнення пропущених значень:**  
  За допомогою класу SimpleImputer (стратегія – середнє значення) заповнюються пропущені значення у вибраних ознаках. Це гарантує, що модель отримає повний набір даних для навчання.

## 4. Розподіл даних на тренувальну та тестову вибірки

* Дані розбиваються за допомогою функції train\_test\_split у співвідношенні 80% для тренування та 20% для тестування. Це дозволяє оцінити продуктивність моделі на невідомих даних.

## 5. Навчання моделей класифікації

Програма навчає дві моделі класифікації:

### 5.1. Logistic Regression

* **Навчання моделі:**  
  Використовується LogisticRegression (максимальна кількість ітерацій встановлена в 1000).
* **Оцінка моделі:**
  + Розраховується точність (accuracy) на тестовій вибірці.
  + Створюється матриця плутанини (confusion matrix), яка показує, скільки книг було правильно/неправильно класифіковано.
  + Генерується звіт класифікації (classification report), в якому наведені показники Precision, Recall, F1-score для кожного класу ("High" і "Low").

### 5.2. Decision Tree Classifier

* **Навчання моделі:**  
  Використовується DecisionTreeClassifier з фіксованим random\_state для відтворюваності.
* **Оцінка моделі:**
  + Так само, як і для логістичної регресії, обчислюється точність, генерується матриця плутанини та звіт класифікації.

## 6. Візуалізація результатів

Програма забезпечує декілька графічних візуалізацій для кращого розуміння результатів класифікації:

* **Матриця плутанини:**  
  За допомогою бібліотеки Seaborn створюються теплові карти (heatmap) для матриць плутанини обох моделей. Це дозволяє візуально оцінити, скільки книг класифіковано правильно або неправильно.
* **Count Plot:**  
  Гістограма (countplot) показує кількість книг у кожній категорії (RatingClass). Це допомагає побачити розподіл даних за класами High та Low.
* **Bar Plot середніх значень ознак:**  
  Створюється стовпчиковий графік, який порівнює середні значення ознак (розподіли оцінок та кількість відгуків) для книг з класами High і Low. Це дозволяє виявити, які ознаки найбільше впливають на класифікацію.

## 7. Експорт результатів у CSV

* **Збереження результатів:**  
  Програма зберігає результати класифікації (стовпці Name та RatingClass) у файл classification\_results.csv. Це дозволяє використовувати результати для подальшого аналізу або інтеграції в інші бізнес-процеси.

## 8. Вивід прикладів даних

* **Приклад даних:**  
  Програма виводить перші 10 рядків датасету з колонками, що містять:
  + Назву книги (Name),
  + Розподіли оцінок (RatingDist1–RatingDist5),
  + Кількість відгуків (CountsOfReview),
  + Загальний рейтинг (Rating),
  + Клас книги (RatingClass).

Це дає змогу швидко перевірити, як відбувається класифікація на рівні окремих записів.

# Загальний висновок

Програма дозволяє:

* **Автоматизувати процес класифікації книг** за їх рейтингом, визначаючи книги з високим потенціалом успіху (High) і менш перспективні (Low) на основі ключових характеристик.
* **Надавати бізнес-аналітику:**  
  Видавці та маркетологи можуть використовувати результати для визначення пріоритетних проектів, коригування маркетингових стратегій, а також ефективного розподілу бюджету на просування книг.
* **Отримати зручні візуалізації:**  
  Теплові карти, гістограми та графіки середніх значень ознак допомагають легко інтерпретувати результати та приймати обґрунтовані рішення.
* **Експортувати результати:**  
  Збереження результатів класифікації у CSV-файл дозволяє подальше використання даних для аналізу, звітності чи інтеграції в бізнес-процеси.

**Код програми**

import pandas as *pd*import numpy as *np*import matplotlib.pyplot as *plt*import seaborn as *sns*from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  
from sklearn.impute import SimpleImputer  
  
*# Завантаження даних  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Обробка колонки 'Rating': замінюємо коми на крапки та переводимо у тип float  
df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
  
*# Видаляємо записи з пропущеним рейтингом  
df* = *df*.dropna(subset=['Rating'])  
  
*# Створення цільової змінної: якщо Rating >= 4.5, то "High", інакше "Low"  
df*['RatingClass'] = np.where(*df*['Rating'] >= 4.5, 'High', 'Low')  
  
*# Вибір ознак для моделі: розподіл оцінок та загальна кількість відгуків  
features* = ['RatingDist1', 'RatingDist2', 'RatingDist3', 'RatingDist4', 'RatingDist5', 'CountsOfReview']  
  
*# Заповнення пропущених значень середнім значенням для ознак  
imputer* = SimpleImputer(strategy='mean')  
*df*[*features*] = *imputer*.fit\_transform(*df*[*features*])  
  
*# Формування X (ознаки) та y (цільова змінна)  
X* = *df*[*features*]  
*y* = *df*['RatingClass']  
  
*# Розподіл даних на тренувальну та тестову вибірки  
X\_train*, *X\_test*, *y\_train*, *y\_test* = train\_test\_split(*X*, *y*, test\_size=0.2, random\_state=42)  
  
*# --- Модель 1: Logistic Regression ---  
log\_reg* = LogisticRegression(max\_iter=1000)  
*log\_reg*.fit(*X\_train*, *y\_train*)  
*y\_pred\_log* = *log\_reg*.predict(*X\_test*)  
*accuracy\_log* = accuracy\_score(*y\_test*, *y\_pred\_log*)  
  
*# Детальний звіт для Logistic Regression*print("=== Logistic Regression Results ===")  
print(f"Accuracy: {*accuracy\_log*:.2f}\n")  
*cm\_log* = confusion\_matrix(*y\_test*, *y\_pred\_log*, labels=['High','Low'])  
print("Confusion Matrix (Logistic Regression):")  
print(*cm\_log*)  
print("\nClassification Report (Logistic Regression):")  
print(classification\_report(*y\_test*, *y\_pred\_log*, target\_names=['High', 'Low']))  
  
*# --- Модель 2: Decision Tree Classifier ---  
tree\_clf* = DecisionTreeClassifier(random\_state=42)  
*tree\_clf*.fit(*X\_train*, *y\_train*)  
*y\_pred\_tree* = *tree\_clf*.predict(*X\_test*)  
*accuracy\_tree* = accuracy\_score(*y\_test*, *y\_pred\_tree*)  
  
*# Детальний звіт для Decision Tree*print("\n=== Decision Tree Classifier Results ===")  
print(f"Accuracy: {*accuracy\_tree*:.2f}\n")  
*cm\_tree* = confusion\_matrix(*y\_test*, *y\_pred\_tree*, labels=['High','Low'])  
print("Confusion Matrix (Decision Tree):")  
print(*cm\_tree*)  
print("\nClassification Report (Decision Tree):")  
print(classification\_report(*y\_test*, *y\_pred\_tree*, target\_names=['High', 'Low']))  
  
*# --- Візуалізація результатів класифікації ---  
  
# Функція для побудови heatmap матриці плутанини*def plot\_confusion\_matrix(*cm*, *title*):  
 plt.figure(figsize=(6, 4))  
 sns.heatmap(*cm*, annot=True, fmt='d', cmap='Blues', xticklabels=['High','Low'], yticklabels=['High','Low'])  
 plt.title(*title*)  
 plt.xlabel("Predicted")  
 plt.ylabel("Actual")  
 plt.show()  
  
plot\_confusion\_matrix(*cm\_log*, "Confusion Matrix - Logistic Regression")  
plot\_confusion\_matrix(*cm\_tree*, "Confusion Matrix - Decision Tree")  
  
*# Візуалізація розподілу кількості книг за класами*plt.figure(figsize=(6,4))  
sns.countplot(data=*df*, x='RatingClass', palette='viridis')  
plt.title("Кількість книг за класами RatingClass")  
plt.xlabel("Rating Class")  
plt.ylabel("Кількість книг")  
plt.show()  
  
*# Візуалізація середніх значень ознак для кожного класу  
class\_summary* = *df*.groupby('RatingClass')[*features*].mean().reset\_index()  
*class\_summary* = pd.melt(*class\_summary*, id\_vars='RatingClass', var\_name='Feature', value\_name='Mean Value')  
  
plt.figure(figsize=(10,6))  
sns.barplot(data=*class\_summary*, x='Feature', y='Mean Value', hue='RatingClass', palette='viridis')  
plt.title("Середні значення ознак для класів High та Low")  
plt.xticks(rotation=45)  
plt.ylabel("Середнє значення")  
plt.xlabel("Ознака")  
plt.show()  
  
*# Виведення перших 10 рядків даних з класифікацією*print("\nПерші 10 рядків даних з колонкою 'RatingClass':")  
print(*df*[['Name', 'RatingDist1', 'RatingDist2', 'RatingDist3', 'RatingDist4', 'RatingDist5', 'CountsOfReview', 'Rating', 'RatingClass']].head(10))  
  
*# --- Збереження результатів класифікації в CSV ---  
# Зберігаємо лише стовпці 'Name' та 'RatingClass'  
classification\_results* = *df*[['Name', 'RatingClass']]  
*classification\_results*.to\_csv('classification\_results.csv', index=False, encoding='utf-8-sig')  
print("\nРезультати класифікації збережено у файл 'classification\_results.csv'")

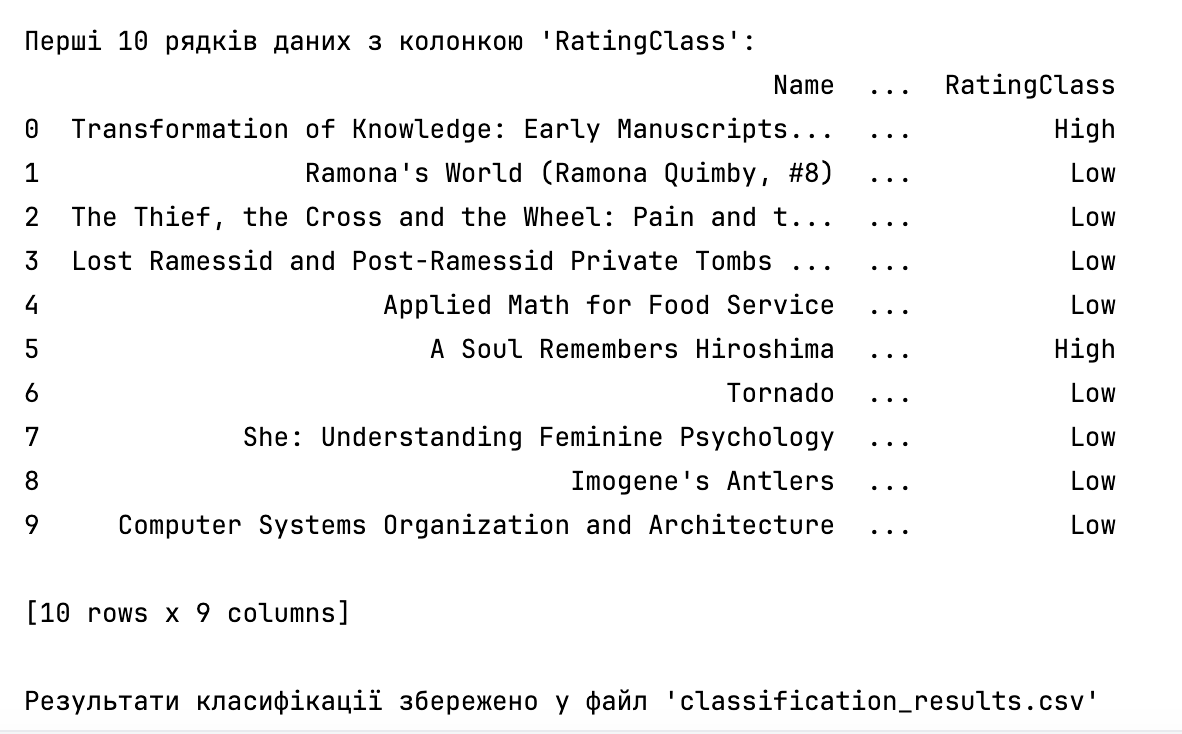
Лістинг програми

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated



A graph of confusion matrix

Description automatically generated

Рис.3. Confusion matrix

A graph with a green and blue bar

Description automatically generated

Рис.4. За розподіл за класами

A graph of different colored bars

Description automatically generated

Рис.5. Середні значення ознак по різним класам

A screenshot of a computer screen

Description automatically generated

Рис.6. Результат класифікації

**Task 3**

Кластеризація по роках за допомогою алгоритму Kmeans

**Кластеризація** — це метод навчання без учителя, який дозволяє поділити об'єкти на групи (класи, кластери), де об'єкти в одному кластері є схожими, а між кластерами — значні відмінності. У випадку з даними книг класифікація на основі подібних характеристик дозволяє отримати групи книг, які мають схожі відгуки, оцінки та інші фактори.

Алгоритм **KMeans** є одним з найбільш популярних алгоритмів кластеризації. Відмінність цього методу полягає в тому, що кількість кластерів KKK необхідно вказати заздалегідь. Алгоритм працює наступним чином:

1. Вибирає KKK початкових центрів кластерів (центроїдів).
2. Призначає кожну точку до найближчого центроїда.
3. Обчислює нові центроїди для кожного кластера.
4. Повторює кроки 2 та 3, поки центри кластерів не стабілізуються (тобто, класи не змінюються).

### 2. **Методи оцінки кластеризації**

* **Silhouette Score** — це міра, яка показує, як правильно кожна точка кластеризована. Значення варіюються від -1 до 1:
  + Значення близьке до 1 вказує на те, що точка добре класифікована в свій кластер.
  + Значення близьке до 0 вказує на те, що точка знаходиться на межі двох кластерів.
  + Значення близьке до -1 свідчить про неправильну класифікацію.
* **Davies-Bouldin Index** — це метрика, яка вимірює середню схожість між кожною парою кластерів. Чим менше значення індексу, тим краща кластеризація.
* **Inertia (WSS)** — це міра того, як сильно точки в кластері відрізняються від центроїда. Чим менше значення Inertia, тим більше згуртовані кластери.

### 3. **PCA (Principal Component Analysis)**

**PCA** — це метод зменшення розмірності, який дозволяє звести кілька взаємозалежних ознак до меншої кількості незалежних компонент. Цей метод корисний, коли дані мають велику кількість ознак і потрібно їх візуалізувати або зменшити обчислювальні витрати.

Після кластеризації ми застосовуємо PCA для проекції даних на дві основні компоненти (2D), що дозволяє зручніше візуалізувати результати кластеризації.

## Опис програми

### 1. Завантаження та обробка даних

df = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')

df = df.dropna(subset=['Language', 'CountsOfReview']) # Видалення рядків з пропущеними значеннями

Ми завантажуємо набір даних за допомогою бібліотеки **Pandas** і використовуємо роздільник ;, оскільки це формат CSV-файлу. Далі видаляємо рядки, де відсутні значення в колонках Language та CountsOfReview, оскільки вони критично важливі для кластеризації.

### 2. Перетворення категоріальних змінних

encoder = OneHotEncoder(sparse\_output=False)

language\_encoded = encoder.fit\_transform(df[['Language']])

language\_encoded\_df = pd.DataFrame(language\_encoded, columns=encoder.get\_feature\_names\_out(['Language']))

Для перетворення категоріальної змінної **Language** в числову форму використовується **OneHotEncoder**, що дозволяє створити нові бінарні змінні для кожної унікальної мови. Це необхідно, оскільки **KMeans** працює тільки з числовими ознаками.

### 3. Заповнення пропущених значень

imputer = SimpleImputer(strategy='mean')

df[['CountsOfReview'] + language\_encoded\_df.columns.tolist()] = imputer.fit\_transform(df[['CountsOfReview'] + language\_encoded\_df.columns.tolist()])

Пропущені значення для числових колонок (у тому числі **CountsOfReview**) заповнюються середнім значенням кожної колонки, що дозволяє зберегти цілісність даних для кластеризації.

### 4. Нормалізація даних

scaler = StandardScaler()

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

Для коректної кластеризації необхідно нормалізувати дані. Це робиться через **StandardScaler**, який змінює середнє значення кожної ознаки до 0, а стандартне відхилення — до 1. Це необхідно, щоб жодна з ознак не мала значно більшого масштабу, ніж інші.

### 5. Кластеризація за допомогою KMeans

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(X\_scaled)

df['KMeans\_Cluster'] = kmeans.labels\_

Алгоритм **KMeans** виконує кластеризацію на трьох кластерах. Після виконання кластеризації кожен елемент отримує мітку кластера в колонці KMeans\_Cluster.

### 6. Оцінка якості кластеризації

#### Silhouette Score

silhouette = silhouette\_score(X\_sampled, kmeans.labels\_[sample\_indices])

Для оцінки якості кластеризації використовуємо **Silhouette Score**, що показує, наскільки добре розділені кластери.

#### Davies-Bouldin Index

davies\_bouldin = davies\_bouldin\_score(X\_sampled, kmeans.labels\_[sample\_indices])

**Davies-Bouldin Index** оцінює схожість між кластерами. Чим менше значення, тим краще.

#### Inertia (WSS)

inertia = kmeans.inertia\_

**Inertia** вимірює згуртованість кластерів — чим менше значення Inertia, тим кращі кластери.

### 7. Візуалізація результатів кластеризації

#### Візуалізація PCA

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

Для зручності візуалізації ми використовуємо **PCA** для зменшення вимірності даних до 2D. Це дозволяє нам побудувати графік, на якому кожен кластер видно в двовимірному просторі.

#### Графіки результатів кластеризації

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=df['KMeans\_Cluster'], cmap='viridis', alpha=0.6)

Ми будуємо графік, на якому кожна точка представлена двома компонентами після **PCA**, а колір точки вказує на приналежність до конкретного кластера.

**Код програми**

import pandas as *pd*import matplotlib.pyplot as *plt*import seaborn as *sns*from sklearn.cluster import KMeans  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.impute import SimpleImputer  
from sklearn.metrics import silhouette\_score, davies\_bouldin\_score  
import numpy as *np*from sklearn.decomposition import PCA  
  
*# Завантаження даних  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Видаляємо записи з пропущеними значеннями для важливих колонок  
df* = *df*.dropna(subset=['Language', 'CountsOfReview', 'PublishYear'])  
  
*# Переконуємося, що PublishYear має числовий тип  
df*['PublishYear'] = *df*['PublishYear'].astype(int)  
  
*# Перетворення категоріальної змінної 'Language' у числову за допомогою OneHotEncoder  
encoder* = OneHotEncoder(sparse\_output=False)  
*language\_encoded* = *encoder*.fit\_transform(*df*[['Language']])  
*language\_encoded\_df* = pd.DataFrame(*language\_encoded*, columns=*encoder*.get\_feature\_names\_out(['Language']))  
  
*# Додавання закодованих колонок до основного DataFrame  
df* = pd.concat([*df*, *language\_encoded\_df*], axis=1)  
  
*# Заповнення пропущених значень для числових колонок (CountsOfReview та закодовані колонки)  
imputer* = SimpleImputer(strategy='mean')  
*cols\_to\_impute* = ['CountsOfReview'] + *language\_encoded\_df*.columns.tolist()  
*df*[*cols\_to\_impute*] = *imputer*.fit\_transform(*df*[*cols\_to\_impute*])  
  
*# Вибір ознак для кластеризації: закодована мова + кількість відгуків  
X* = *df*[*language\_encoded\_df*.columns.tolist() + ['CountsOfReview']]  
  
*# Нормалізація ознак  
scaler* = StandardScaler()  
*X\_scaled* = *scaler*.fit\_transform(*X*)  
  
*# Кластеризація за допомогою KMeans (3 кластери)  
kmeans* = KMeans(n\_clusters=3, random\_state=42)  
*kmeans*.fit(*X\_scaled*)  
*df*['KMeans\_Cluster'] = *kmeans*.labels\_  
  
*# Виведення інформації про кількість елементів в кожному кластері*print("Кількість елементів в кожному кластері:")  
print(*df*['KMeans\_Cluster'].value\_counts())  
  
*# Виведення координат центрів кластерів*print("\nЦентри кластерів (KMeans):")  
print(*kmeans*.cluster\_centers\_)  
  
*# Статистика по кожному кластеру для кожної ознаки*print("\nСтатистика по кожному кластеру:")  
for *cluster* in range(3):  
 *cluster\_data* = *df*[*df*['KMeans\_Cluster'] == *cluster*]  
 print(f"\nКластер {*cluster*}:")  
 print(*cluster\_data*[*language\_encoded\_df*.columns.tolist() + ['CountsOfReview']].describe())  
  
*# Розподіл по кожному кластеру для ознак Language та CountsOfReview*print("\nРозподіл по кожному кластеру для ознак Language та CountsOfReview:")  
print(*df*.groupby('KMeans\_Cluster')[*language\_encoded\_df*.columns.tolist() + ['CountsOfReview']].mean())  
  
*# Додаткові метрики:  
sample\_size* = int(0.1 \* *X\_scaled*.shape[0]) *# 10% від усіх даних  
sample\_indices* = np.random.choice(*X\_scaled*.shape[0], *sample\_size*, replace=False)  
*X\_sampled* = *X\_scaled*[*sample\_indices*]  
*silhouette* = silhouette\_score(*X\_sampled*, *kmeans*.labels\_[*sample\_indices*])  
*davies\_bouldin* = davies\_bouldin\_score(*X\_sampled*, *kmeans*.labels\_[*sample\_indices*])  
*inertia* = *kmeans*.inertia\_  
print(f"\nSilhouette Score: {*silhouette*:.4f}")  
print(f"\nDavies-Bouldin Index: {*davies\_bouldin*:.4f}")  
print(f"\nInertia (WSS): {*inertia*:.4f}")  
  
*# Візуалізація результатів кластеризації за допомогою PCA (зменшення вимірності до 2D)  
pca* = PCA(n\_components=2)  
*X\_pca* = *pca*.fit\_transform(*X\_scaled*)  
plt.figure(figsize=(10, 6))  
plt.scatter(*X\_pca*[:, 0], *X\_pca*[:, 1], c=*df*['KMeans\_Cluster'], cmap='viridis', alpha=0.6)  
*centers* = *pca*.transform(*kmeans*.cluster\_centers\_)  
plt.scatter(*centers*[:, 0], *centers*[:, 1], c='red', marker='X', s=200, label='Cluster Centers')  
plt.title('KMeans Clustering with Centers (PCA Projection)', fontsize=16)  
plt.xlabel('PCA Component 1', fontsize=12)  
plt.ylabel('PCA Component 2', fontsize=12)  
plt.colorbar(label='Cluster')  
plt.legend()  
plt.grid(True)  
plt.show()  
  
*# Візуалізація розподілу мов по кластерах*plt.figure(figsize=(10, 6))  
sns.countplot(data=*df*, x='Language', hue='KMeans\_Cluster', palette='Set2')  
plt.title('Розподіл по мовах в кожному кластері')  
plt.ylabel('Кількість книг')  
plt.xlabel('Мова')  
plt.xticks(rotation=45)  
plt.show()  
  
*# Візуалізація розподілу "CountsOfReview" по кластерах*plt.figure(figsize=(10, 6))  
sns.boxplot(x='KMeans\_Cluster', y='CountsOfReview', data=*df*, palette='Set3')  
plt.title('Розподіл кількості оглядів по кластерах')  
plt.xlabel('Кластер')  
plt.ylabel('Кількість оглядів')  
plt.show()  
  
*# Додаткова сегментація: Групування даних за роками публікації  
year\_cluster* = *df*.groupby(['PublishYear', 'KMeans\_Cluster']).size().reset\_index(name='Count')  
print("\nРозподіл кластерів по роках публікації:")  
print(*year\_cluster*)  
  
*# Візуалізація розподілу кластерів за роками (barplot)*plt.figure(figsize=(12, 6))  
*year\_pivot* = *year\_cluster*.pivot(index='PublishYear', columns='KMeans\_Cluster', values='Count').fillna(0)  
*year\_pivot*.plot(kind='bar', stacked=True, colormap='viridis', figsize=(12, 6))  
plt.title('Розподіл кластерів за роками публікації')  
plt.xlabel('Рік публікації')  
plt.ylabel('Кількість книг')  
plt.legend(title='Кластер')  
plt.grid(axis='y')  
plt.show()  
  
*# Візуалізація середніх значень CountsOfReview за роками для кожного кластера*plt.figure(figsize=(12, 6))  
sns.lineplot(data=*df*, x='PublishYear', y='CountsOfReview', hue='KMeans\_Cluster', marker='o', palette='viridis')  
plt.title('Середня кількість відгуків по кластерах за роками публікації')  
plt.xlabel('Рік публікації')  
plt.ylabel('CountsOfReview')  
plt.grid(True)  
plt.show()  
  
*# Додатково: відображення перших кількох рядків даних з кластеризацією*print("\nПерші кілька рядків з класифікацією:")  
print(*df*[['Name', 'PublishYear', 'Language', 'CountsOfReview', 'KMeans\_Cluster']].head())  
  
*# --- Business Value ---*print("\n=== Business Value ===")  
print("1. Сегментація ринку за мовою та кількістю відгуків допомагає видавцям:")  
print(" - Розуміти, які мовні сегменти приносять більше відгуків та, можливо, є більш прибутковими.")  
print(" - Оптимізувати маркетингові стратегії, спрямовуючи зусилля на сегменти з високою активністю.")  
print("2. Групування за роками публікації дозволяє відслідковувати динаміку змін:")  
print(" - Аналізувати, як змінюється популярність мов та відгуки з часом.")  
print(" - Визначати тренди: чи зростає кількість книг у певних кластерах, що може вказувати на зростання попиту.")  
print("3. Загалом, цей аналіз допомагає приймати стратегічні рішення щодо:")  
print(" - Планування майбутніх випусків.")  
print(" - Розподілу рекламного бюджету та ресурсів.")  
print(" - Оптимізації продуктового портфеля відповідно до ринкових трендів.\n")

Лістинг та результати роботи

Кількість елементів в кожному кластері:

KMeans\_Cluster

0 117951

2 1857

1 1

Name: count, dtype: int64

Центри кластерів (KMeans):

[[ 1.59334125e-04 7.96653163e-05 2.75990931e-04 1.37986392e-04

7.96653163e-05 7.96653163e-05 3.65125695e-04 -1.24729557e-01

7.48973401e-03 3.89781096e-02 2.39009918e-04 7.96653163e-05

1.12664594e-04 3.62988331e-03 7.96653163e-05 2.25339065e-04

2.54297431e-03 7.96653163e-05 1.78142269e-04 1.37986392e-04

5.22560778e-04 1.59334125e-04 7.96653163e-05 7.96653163e-05

1.12664594e-04 -2.85491680e-03 1.12664594e-04 1.12664594e-04

9.05671360e-04 8.84319626e-04 2.98108633e-04 7.96653163e-05

4.50757179e-04 7.96653163e-05 1.12664594e-04 8.58743483e-04

7.96653163e-05 1.12664594e-04 3.73720795e-04 1.59334125e-04

4.14031777e-04 1.59334125e-04 6.66864097e-04 1.59334125e-04

4.71423478e-04 1.78142269e-04 7.96653163e-05 3.29196157e-03

7.96653163e-05 3.73720795e-04 7.96653163e-05 7.96653163e-05

7.96653163e-05 7.96653163e-05 2.75990931e-04 1.78142269e-04

7.96653163e-05 1.37986392e-04 7.96653163e-05 6.94886971e-04

-7.35049312e-04]

[-1.01149727e-02 -5.05737552e-03 -1.75206708e-02 -8.75975935e-03

-5.05737552e-03 -5.05737552e-03 -2.31791931e-02 -2.20953622e-01

-4.75469115e-01 -2.47443865e+00 -1.51730134e-02 -5.05737552e-03

-7.15226131e-03 -2.30435073e-01 -5.05737552e-03 -1.43051497e-02

-1.61435072e-01 -5.05737552e-03 -1.13089659e-02 -8.75975935e-03

-3.31736094e-02 -1.01149727e-02 -5.05737552e-03 -5.05737552e-03

-7.15226131e-03 3.46131838e+02 -7.15226131e-03 -7.15226131e-03

-5.74945331e-02 -5.61390658e-02 -1.89247639e-02 -5.05737552e-03

-2.86153175e-02 -5.05737552e-03 -7.15226131e-03 -5.45154212e-02

-5.05737552e-03 -7.15226131e-03 -2.37248339e-02 -1.01149727e-02

-2.62838870e-02 -1.01149727e-02 -4.23343849e-02 -1.01149727e-02

-2.99272716e-02 -1.13089659e-02 -5.05737552e-03 -2.08982863e-01

-5.05737552e-03 -2.37248339e-02 -5.05737552e-03 -5.05737552e-03

-5.05737552e-03 -5.05737552e-03 -1.75206708e-02 -1.13089659e-02

-5.05737552e-03 -8.75975935e-03 -5.05737552e-03 -4.41133548e-02

-6.31422940e-17]

[-1.01149727e-02 -5.05737552e-03 -1.75206708e-02 -8.75975935e-03

-5.05737552e-03 -5.05737552e-03 -2.31791931e-02 7.92256162e+00

-4.75469115e-01 -2.47443865e+00 -1.51730134e-02 -5.05737552e-03

-7.15226131e-03 -2.30435073e-01 -5.05737552e-03 -1.43051497e-02

-1.61435072e-01 -5.05737552e-03 -1.13089659e-02 -8.75975935e-03

-3.31736094e-02 -1.01149727e-02 -5.05737552e-03 -5.05737552e-03

-7.15226131e-03 -5.05737552e-03 -7.15226131e-03 -7.15226131e-03

-5.74945331e-02 -5.61390658e-02 -1.89247639e-02 -5.05737552e-03

-2.86153175e-02 -5.05737552e-03 -7.15226131e-03 -5.45154212e-02

-5.05737552e-03 -7.15226131e-03 -2.37248339e-02 -1.01149727e-02

-2.62838870e-02 -1.01149727e-02 -4.23343849e-02 -1.01149727e-02

-2.99272716e-02 -1.13089659e-02 -5.05737552e-03 -2.08982863e-01

-5.05737552e-03 -2.37248339e-02 -5.05737552e-03 -5.05737552e-03

-5.05737552e-03 -5.05737552e-03 -1.75206708e-02 -1.13089659e-02

-5.05737552e-03 -8.75975935e-03 -5.05737552e-03 -4.41133548e-02

4.66880998e-02]]

Статистика по кожному кластеру:

Кластер 0:

Language\_afr Language\_ang ... Language\_zho CountsOfReview

count 117951.000000 117951.000000 ... 117951.000000 117951.000000

mean 0.000059 0.000015 ... 0.001128 156.972420

std 0.005823 0.002912 ... 0.025369 890.095830

min 0.000000 0.000000 ... 0.000000 0.000000

25% 0.000000 0.000000 ... 0.000000 8.000000

50% 0.000000 0.000000 ... 0.000000 152.000000

75% 0.000058 0.000015 ... 0.001110 157.634143

max 1.000000 1.000000 ... 1.000000 94850.000000

[8 rows x 61 columns]

Кластер 1:

Language\_afr Language\_ang ... Language\_zho CountsOfReview

count 1.0 1.0 ... 1.0 1.000000

mean 0.0 0.0 ... 0.0 157.634143

std NaN NaN ... NaN NaN

min 0.0 0.0 ... 0.0 157.634143

25% 0.0 0.0 ... 0.0 157.634143

50% 0.0 0.0 ... 0.0 157.634143

75% 0.0 0.0 ... 0.0 157.634143

max 0.0 0.0 ... 0.0 157.634143

[8 rows x 61 columns]

Кластер 2:

Language\_afr Language\_ang ... Language\_zho CountsOfReview

count 1857.0 1857.0 ... 1857.0 1857.000000

mean 0.0 0.0 ... 0.0 199.664786

std 0.0 0.0 ... 0.0 1401.602033

min 0.0 0.0 ... 0.0 0.000000

25% 0.0 0.0 ... 0.0 157.634143

50% 0.0 0.0 ... 0.0 157.634143

75% 0.0 0.0 ... 0.0 157.634143

max 0.0 0.0 ... 0.0 57034.000000

[8 rows x 61 columns]

Розподіл по кожному кластеру для ознак Language та CountsOfReview:

Language\_afr Language\_ang ... Language\_zho CountsOfReview

KMeans\_Cluster ...

0 0.000059 0.000015 ... 0.001128 156.972420

1 0.000000 0.000000 ... 0.000000 157.634143

2 0.000000 0.000000 ... 0.000000 199.664786

[3 rows x 61 columns]

Silhouette Score: 0.7509

Davies-Bouldin Index: 0.1383

Inertia (WSS): 7057888.4644

Розподіл кластерів по роках публікації:

PublishYear KMeans\_Cluster Count

0 1852.0 0 1

1 1866.0 0 1

2 1899.0 0 1

3 1900.0 0 5

4 1913.0 0 3

.. ... ... ...

119 2017.0 0 31

120 2018.0 0 33

121 2018.0 2 1

122 2019.0 0 27

123 2020.0 0 7

[124 rows x 3 columns]

Перші кілька рядків з класифікацією:

Name ... KMeans\_Cluster

1 Ramona's World (Ramona Quimby, #8) ... 0

6 Tornado ... 0

8 Imogene's Antlers ... 0

13 Morning Comes Softly ... 0

17 Van Gogh ... 0

[5 rows x 5 columns]

=== Business Value ===

1. Сегментація ринку за мовою та кількістю відгуків допомагає видавцям:

- Розуміти, які мовні сегменти приносять більше відгуків та, можливо, є більш прибутковими.

- Оптимізувати маркетингові стратегії, спрямовуючи зусилля на сегменти з високою активністю.

2. Групування за роками публікації дозволяє відслідковувати динаміку змін:

- Аналізувати, як змінюється популярність мов та відгуки з часом.

- Визначати тренди: чи зростає кількість книг у певних кластерах, що може вказувати на зростання попиту.

3. Загалом, цей аналіз допомагає приймати стратегічні рішення щодо:

- Планування майбутніх випусків.

- Розподілу рекламного бюджету та ресурсів.

- Оптимізації продуктового портфеля відповідно до ринкових трендів.

A graph with red and blue dots

Description automatically generated

Рис.7. Кластеризація візуалізація PCA

A white rectangular object with numbers

Description automatically generated

Рис.8. Розподіл по мовам по кластерах

A graph with numbers and lines

Description automatically generated

Рис.9. Розподіл оглядів за кластерами

A graph with purple and black lines

Description automatically generated

Рис.10. Розподіл кількості книг в кластерів за роками публікацій

A graph with a line graph

Description automatically generated

Рис.10. Розподіл відгуків по кластерам за роками публікацій

**Task 4**

Кореляції між мовою та кількості оглядів видавцем і кількості оглядів, автором та кількістю оглядів

**Кореляція та статистичні методи для аналізу взаємозв'язків**

Кореляція — це міра того, як дві змінні взаємопов'язані. Якщо між двома змінними існує сильна кореляція, це означає, що зміна однієї змінної призводить до зміни іншої. Важливі види кореляції включають:

* **Пірсонівська кореляція** — вимірює лінійну залежність між двома змінними. Вона використовується для числових змінних і варіюється від -1 до +1:
  + +1 означає ідеальну позитивну лінійну залежність.
  + -1 означає ідеальну негативну лінійну залежність.
  + 0 означає відсутність лінійної залежності.
* **Спірменівська кореляція** — вимірює монотонну залежність, що є більш гнучким методом для виявлення залежностей між змінними. Вона використовується для обчислення кореляцій між порядковими змінними або для не лінійних, але монотонних зв'язків.
* **Взаємна інформація (Mutual Information)** — міра залежності між двома змінними, яка вимірює спільну інформацію, що знаходиться в цих змінних. Це важливо для виявлення нелінійних залежностей.

**2. Хі-квадрат тест для категоріальних змінних**

Хі-квадрат тест (χ²) використовується для оцінки статистичної залежності між двома категоріальними змінними. Якщо p-значення тесту менше 0.05, це свідчить про значну залежність між змінними.

**3. Графічне представлення даних**

* **Гістограми** — використовуються для візуалізації розподілу числових даних.
* **Countplot** — використовується для візуалізації розподілу категоріальних змінних.
* **Розсіяні графіки** — використовуються для аналізу залежностей між двома змінними.

**Опис програми**

**1. Завантаження та попередня обробка даних**

df = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')

df['Language'] = df['Language'].fillna(df['Language'].mode()[0])

df['Authors'] = df['Authors'].fillna(df['Authors'].mode()[0])

df['PublisherNaming'] = df['PublisherNaming'].fillna(df['PublisherNaming'].mode()[0])

Ми завантажуємо дані з CSV-файлу, використовуючи роздільник ;. Потім обробляємо пропущені значення в колонках категоріальних змінних **Language**, **Authors**, і **PublisherNaming**, заповнюючи їх найбільш поширеним значенням за допомогою методу **fillna()**.

**2. Обробка числових даних**

df['Rating'] = df['Rating'].replace({',': '.'}, regex=True).astype(float)

df['CountsOfReviewCategory'] = pd.cut(df['CountsOfReview'], bins=[0, 100, 500, 1000, 5000, 10000, 20000],

labels=["0-100", "101-500", "501-1000", "1001-5000", "5001-10000", "10001-20000"])

У колонці **Rating** ми заміняємо коми на крапки для коректного перетворення значень на тип **float**. Потім створюємо категоріальну змінну **CountsOfReviewCategory**, яка класифікує кількість оглядів за категоріями, щоб полегшити подальший аналіз залежностей.

**3. Заповнення пропущених значень**

imputer = SimpleImputer(strategy='mean')

df[['CountsOfReview', 'Rating']] = imputer.fit\_transform(df[['CountsOfReview', 'Rating']])

З використанням **SimpleImputer** ми заповнюємо пропущені значення в числових колонках **CountsOfReview** та **Rating** середнім значенням.

**4. Хі-квадрат тест**

def chi\_square\_test(column1, column2):

crosstab = pd.crosstab(column1, column2)

chi2, p\_val, \_, \_ = chi2\_contingency(crosstab)

print(f"\nChi-squared test між {column1.name} та {column2.name}:")

print(f"Chi-squared: {chi2}")

print(f"P-value: {p\_val}")

if p\_val < 0.05:

print("Є статистична залежність між цими змінними (р < 0.05).")

else:

print("Немає статистичної залежності між цими змінними (р >= 0.05).")

Функція **chi\_square\_test** здійснює перевірку залежності між двома категоріальними змінними за допомогою **хі-квадрат тесту**. Виводиться значення статистики хі-квадрат і p-значення. Якщо p-значення менше 0.05, ми стверджуємо, що між змінними є статистична залежність.

**5. Оцінка кореляцій між числовими змінними**

correlation\_matrix = df[['CountsOfReview', 'Rating']].corr(method='pearson')

Ми обчислюємо **Пірсонівську кореляцію** між змінними **CountsOfReview** і **Rating** для оцінки лінійної залежності.

spearman\_corr, \_ = spearmanr(df['CountsOfReview'], df['Rating'])

Також обчислюємо **Спірменівську кореляцію**, яка дає оцінку монотонної залежності між двома змінними.

mi = mutual\_info\_score(df['CountsOfReviewCategory'], df['Rating'])

Для оцінки **нелінійних залежностей** між категоріальною змінною **CountsOfReviewCategory** та числовою змінною **Rating** використовується **взаємна інформація**.

**6. Візуалізація результатів**

**Графіки для категоріальних змінних**

sns.countplot(data=df, x='Language', palette='Set2')

Ми будуємо **countplot** для категоріальної змінної **Language**, щоб побачити розподіл книг за мовами.

sns.countplot(data=df, x='Authors', palette='Set2', order=df['Authors'].value\_counts().index[:10])

Побудова графіка для **Authors** дозволяє візуалізувати топ 10 авторів, що зустрічаються в даних.

**Графік для категоріальної змінної CountsOfReviewCategory**

sns.countplot(data=df, x='CountsOfReviewCategory', palette='Set3')

Цей графік показує, як розподіляються книги за категоріями кількості оглядів.

**Розподіл кількості оглядів**

sns.histplot(df['CountsOfReview'], kde=True, color='blue', bins=30)

**Гістограма** для змінної **CountsOfReview** дозволяє побачити, як розподіляються відгуки по різних діапазонах.

**Залежність між кількістю оглядів та рейтингом**

sns.scatterplot(data=df, x='CountsOfReview', y='Rating', color='green')

**Розсіяний графік** показує залежність між кількістю оглядів і рейтингом. Це допомагає зрозуміти, чи є лінійна чи інша залежність між цими двома змінними.

**Код програми**

import matplotlib.pyplot as *plt*import pandas as *pd*import seaborn as *sns*from scipy.stats import chi2\_contingency, spearmanr  
from sklearn.impute import SimpleImputer  
from sklearn.metrics import mutual\_info\_score  
  
*# Завантаження даних  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Заповнення пропусків для категоріальних змінних 'Language', 'Authors', 'PublisherNaming'  
df*['Language'] = *df*['Language'].fillna(*df*['Language'].mode()[0])  
*df*['Authors'] = *df*['Authors'].fillna(*df*['Authors'].mode()[0])  
*df*['PublisherNaming'] = *df*['PublisherNaming'].fillna(*df*['PublisherNaming'].mode()[0])  
  
*# Перетворення значень у колонці 'Rating': заміна ком на крапки та перетворення на float  
df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
  
*# Створення категоріальної змінної для 'CountsOfReview'  
df*['CountsOfReviewCategory'] = pd.cut(*df*['CountsOfReview'], bins=[0, 100, 500, 1000, 5000, 10000, 20000],  
 labels=["0-100", "101-500", "501-1000", "1001-5000", "5001-10000", "10001-20000"])  
  
*# Заповнення пропущених значень для числових колонок, якщо є  
imputer* = SimpleImputer(strategy='mean')  
*df*[['CountsOfReview', 'Rating']] = *imputer*.fit\_transform(*df*[['CountsOfReview', 'Rating']])  
  
  
*# --- Аналіз залежностей за допомогою статистичних тестів ---*def chi\_square\_test(*column1*, *column2*):  
 *crosstab* = pd.crosstab(*column1*, *column2*)  
 *chi2*, *p\_val*, *\_*, *\_* = chi2\_contingency(*crosstab*)  
 print(f"\nChi-squared test між {*column1*.name} та {*column2*.name}:")  
 print(f"Chi-squared: {*chi2*:.2f}")  
 print(f"P-value: {*p\_val*:.4f}")  
 if *p\_val* < 0.05:  
 print("Є статистична залежність між цими змінними (р < 0.05).")  
 else:  
 print("Немає статистичної залежності між цими змінними (р >= 0.05).")  
  
  
*# Тести залежностей*chi\_square\_test(*df*['Language'], *df*['CountsOfReviewCategory'])  
chi\_square\_test(*df*['Authors'], *df*['CountsOfReviewCategory'])  
chi\_square\_test(*df*['PublisherNaming'], *df*['CountsOfReviewCategory'])  
  
*# Pearson кореляція (для числових змінних)  
pearson\_corr* = *df*[['CountsOfReview', 'Rating']].corr(method='pearson')  
print("\nPearson Correlation Matrix:")  
print(*pearson\_corr*)  
  
*# Spearman кореляція (для монотонних залежностей)  
spearman\_corr*, *\_* = spearmanr(*df*['CountsOfReview'], *df*['Rating'])  
print(f"\nSpearman Correlation between 'CountsOfReview' and 'Rating': {*spearman\_corr*:.4f}")  
  
*# Mutual Information (для нелінійних залежностей)  
df*['CountsOfReviewCategory'] = *df*['CountsOfReviewCategory'].fillna(*df*['CountsOfReviewCategory'].mode()[0])  
*mi* = mutual\_info\_score(*df*['CountsOfReviewCategory'], *df*['Rating'])  
print(f"\nMutual Information between 'CountsOfReviewCategory' and 'Rating': {*mi*:.4f}")  
  
*# --- Розширена візуалізація ---  
  
# 1. Візуалізація розподілу категоріальних змінних  
  
# Розподіл по мовах*plt.figure(figsize=(10, 6))  
sns.countplot(data=*df*, x='Language', palette='Set2', order=*df*['Language'].value\_counts().index)  
plt.title('Розподіл книг по мовах')  
plt.xlabel('Мова')  
plt.ylabel('Кількість книг')  
plt.xticks(rotation=45)  
plt.tight\_layout()  
plt.show()  
  
*# Топ 10 авторів*plt.figure(figsize=(10, 6))  
sns.countplot(data=*df*, x='Authors', palette='Set2', order=*df*['Authors'].value\_counts().index[:10])  
plt.title('Топ 10 авторів')  
plt.xlabel('Автор')  
plt.ylabel('Кількість книг')  
plt.xticks(rotation=90)  
plt.tight\_layout()  
plt.show()  
  
*# Розподіл по категоріям кількості оглядів*plt.figure(figsize=(10, 6))  
sns.countplot(data=*df*, x='CountsOfReviewCategory', palette='Set3',  
 order=*df*['CountsOfReviewCategory'].value\_counts().index)  
plt.title('Розподіл книг по категоріях кількості оглядів')  
plt.xlabel('Категорія кількості оглядів')  
plt.ylabel('Кількість книг')  
plt.tight\_layout()  
plt.show()  
  
*# 2. Візуалізація числових змінних  
  
# Гістограма кількості оглядів з KDE*plt.figure(figsize=(10, 6))  
sns.histplot(*df*['CountsOfReview'], kde=True, color='blue', bins=30)  
plt.title('Розподіл кількості оглядів')  
plt.xlabel('Кількість оглядів')  
plt.ylabel('Частота')  
plt.tight\_layout()  
plt.show()  
  
*# Scatter plot: залежність між кількістю оглядів та рейтингом*plt.figure(figsize=(10, 6))  
sns.scatterplot(data=*df*, x='CountsOfReview', y='Rating', color='green', alpha=0.7)  
plt.title('Залежність між кількістю оглядів та рейтингом')  
plt.xlabel('Кількість оглядів')  
plt.ylabel('Рейтинг')  
plt.tight\_layout()  
plt.show()  
  
*# Boxplot: розподіл рейтингу по категоріях кількості оглядів*plt.figure(figsize=(10, 6))  
sns.boxplot(data=*df*, x='CountsOfReviewCategory', y='Rating', palette='Set1')  
plt.title('Розподіл рейтингу по категоріях кількості оглядів')  
plt.xlabel('Категорія кількості оглядів')  
plt.ylabel('Рейтинг')  
plt.tight\_layout()  
plt.show()  
  
*# 3. Додаткова візуалізація залежностей через теплову карту кореляцій для числових змінних*plt.figure(figsize=(8, 6))  
*corr\_matrix* = *df*[['CountsOfReview', 'Rating']].corr()  
sns.heatmap(*corr\_matrix*, annot=True, cmap='coolwarm')  
plt.title('Кореляційна матриця для CountsOfReview та Rating')  
plt.tight\_layout()  
plt.show()  
  
*# --- Business Value та детальна інтерпретація ---*print("\n=== Business Value ===")  
print("1. Аналіз категоріальних змінних (Language, Authors, PublisherNaming, CountsOfReviewCategory):")  
print("- Дозволяє видавцям визначити, які сегменти ринку (мовні, авторські або за видавництвом) є найбільш активними.")  
print(" - Ця інформація може бути використана для таргетованої реклами та оптимізації маркетингових кампаній.")  
print("2. Залежність між кількістю оглядів та рейтингом:")  
print("- Кореляційний аналіз та візуалізації допомагають зрозуміти, як кількість відгуків впливає на загальний "  
 "рейтинг книги.")  
print(" - Це може вплинути на стратегію підвищення якості продукту та роботу з клієнтами.")  
print("3. Візуалізація розподілу даних:")  
print("- Графіки (histogram, scatter plot, boxplot, heatmap) забезпечують наочність даних і дозволяють виявити "  
 "закономірності,")  
print(" які важливі для прийняття бізнес-рішень, таких як планування випусків та розподіл рекламного бюджету.")  
print("4. Використання статистичних тестів (Chi-squared, Spearman, Mutual Information):")  
print("- Ці показники допомагають об'єктивно оцінити залежності між змінними та визначити, чи існують статистично "  
 "значущі зв'язки.")  
print(" - Результати тестів можна використовувати для подальшої сегментації ринку та оптимізації бізнес-процесів.")  
print("5. Загалом, цей аналіз дозволяє:")  
print(" - Зрозуміти ринкові тренди та змінність вподобань клієнтів.")  
print(" - Оптимізувати продуктовий портфель та спрямувати зусилля на найбільш перспективні сегменти.")  
print(" - Приймати обґрунтовані рішення щодо майбутніх інвестицій та маркетингових стратегій.\n")

Лістинг та результати роботи

Chi-squared test між Language та CountsOfReviewCategory:

Chi-squared: 1580.78

P-value: 0.0000

Є статистична залежність між цими змінними (р < 0.05).

Chi-squared test між Authors та CountsOfReviewCategory:

Chi-squared: 229012.88

P-value: 1.0000

Немає статистичної залежності між цими змінними (р >= 0.05).

Chi-squared test між PublisherNaming та CountsOfReviewCategory:

Chi-squared: 85099.15

P-value: 0.0000

Є статистична залежність між цими змінними (р < 0.05).

Pearson Correlation Matrix:

CountsOfReview Rating

CountsOfReview 1.000000 0.029469

Rating 0.029469 1.000000

Spearman Correlation between 'CountsOfReview' and 'Rating': 0.2171

Mutual Information between 'CountsOfReviewCategory' and 'Rating': 0.0229

=== Business Value ===

1. Аналіз категоріальних змінних (Language, Authors, PublisherNaming, CountsOfReviewCategory):

- Дозволяє видавцям визначити, які сегменти ринку (мовні, авторські або за видавництвом) є найбільш активними.

- Ця інформація може бути використана для таргетованої реклами та оптимізації маркетингових кампаній.

2. Залежність між кількістю оглядів та рейтингом:

- Кореляційний аналіз та візуалізації допомагають зрозуміти, як кількість відгуків впливає на загальний рейтинг книги.

- Це може вплинути на стратегію підвищення якості продукту та роботу з клієнтами.

3. Візуалізація розподілу даних:

- Графіки (histogram, scatter plot, boxplot, heatmap) забезпечують наочність даних і дозволяють виявити закономірності,

які важливі для прийняття бізнес-рішень, таких як планування випусків та розподіл рекламного бюджету.

4. Використання статистичних тестів (Chi-squared, Spearman, Mutual Information):

- Ці показники допомагають об'єктивно оцінити залежності між змінними та визначити, чи існують статистично значущі зв'язки.

- Результати тестів можна використовувати для подальшої сегментації ринку та оптимізації бізнес-процесів.

5. Загалом, цей аналіз дозволяє:

- Зрозуміти ринкові тренди та змінність вподобань клієнтів.

- Оптимізувати продуктовий портфель та спрямувати зусилля на найбільш перспективні сегменти.

- Приймати обґрунтовані рішення щодо майбутніх інвестицій та маркетингових стратегій.

A white rectangular object with black lines

Description automatically generated

Рис.11. Графік значної переваги англомовних книг

A graph of different colored bars

Description automatically generated

Рис.12. Рейтинг авторів за кількістю книг

A graph with numbers and a bar

Description automatically generated with medium confidence

Рис.13. Розподіл по кількості оглядів до книг

A graph with numbers and lines

Description automatically generated

Рис.14. Розподіл по кількості оглядів за частотою

A green and black dotted graph

Description automatically generated

Рис.15. Розподіл по кількості оглядів за до рейтингу

A chart with different colored rectangles

Description automatically generated

Рис.16. Розподіл рейтингу по категоріях кількості оглядів

A graph with red and blue dots

Description automatically generated

Рис.17. Залежність між оглядами та рейтингом

A screenshot of a computer screen

Description automatically generated

Рис.18. Кореляційна матриця залежності

**Висновок**

В рамках виконання лабораторної роботи було створено чотири моделі, кожна з яких реалізує різні підходи до аналізу даних та вирішення задач на основі наданих даних

**Джерела**

1. **"Hands-On Regression Analysis with R: Build Effective Models to Solve Real-World Problems"** — Абдулл Ахмад, 2020.
2. **"Pattern Recognition and Machine Learning"** — Christopher M. Bishop, 2006.
3. **"Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking"** — Foster Provost, Tom Fawcett, 2013.
4. **"Applied Predictive Modeling"** — Max Kuhn, Kjell Johnson, 2013.