**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*def get\_user\_input():  
 *"""  
 Отримує введення користувача для прогнозування рейтингу книги.  
 Повертає: рік публікації, мову, назву книги та ознаки з назви.  
 """* print("\nВведіть 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*, *title*, *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  
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'])  
  
*# Створення ознак з назви книги (Name)  
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 (якщо вони є)  
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")  
  
*# --- Перший прогноз ---*print("=== Перший прогноз ===")  
*publish\_year*, *language*, *title*, *user\_letter\_features* = get\_user\_input()  
  
*# Формуємо DataFrame для введених даних  
user\_data* = pd.DataFrame({  
 'PublishYear': [*publish\_year*]  
})  
  
*# One-hot кодування введеної мови (використовуємо 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 для першої книги  
predicted\_rating* = *model*.predict(*user\_input\_final*)[0]  
print(f"\nПрогнозований рейтинг книги: {*predicted\_rating*:.2f}")  
  
*# Перевірка, чи існує книга в дата-сеті (порівнюємо назву, ігноруючи регістр)  
mask* = *df*['Name'].str.lower() == *title*.lower()  
if *mask*.any():  
 *real\_rating* = *df*.loc[*mask*, 'Rating'].iloc[0]  
 print(f"Реальний рейтинг книги: {*real\_rating*:.2f}")  
else:  
 print("Книга не знайдена в дата-сеті, порівняння неможливе.")  
  
*# --- Додаткові прогнози ---*while True:  
 *choice* = input("\nБажаєте зробити ще один прогноз? (y/n): ").strip().lower()  
 if *choice* != 'y':  
 print("Завершення програми.")  
 break  
  
 *publish\_year*, *language*, *title*, *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]  
  
 *predicted\_rating* = *model*.predict(*user\_input\_final*)[0]  
 print(f"\nПрогнозований рейтинг книги: {*predicted\_rating*:.2f}")  
  
 *mask* = *df*['Name'].str.lower() == *title*.lower()  
 if *mask*.any():  
 *real\_rating* = *df*.loc[*mask*, 'Rating'].iloc[0]  
 print(f"Реальний рейтинг книги: {*real\_rating*:.2f}")  
 else:  
 print("Книга не знайдена в дата-сеті, порівняння неможливе.")  
  
*# --- Візуалізація результатів (опціонально) ---*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')

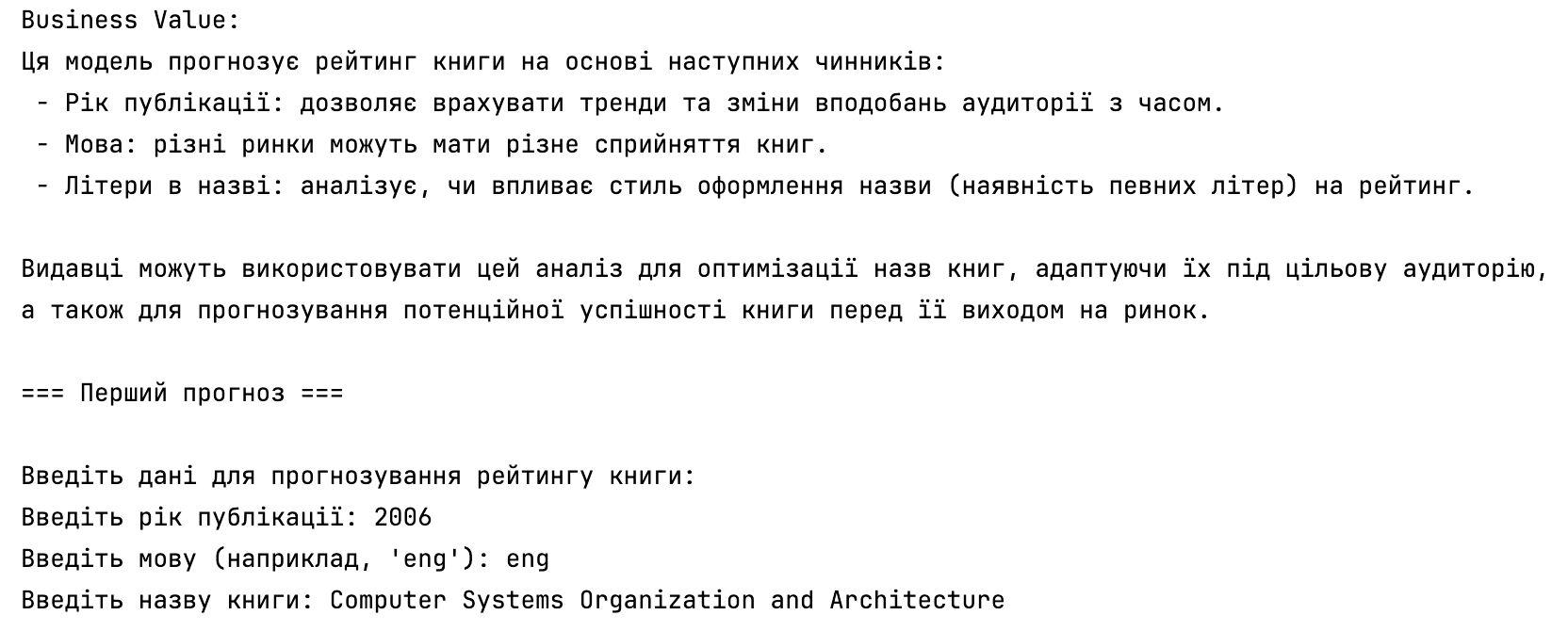
plt.ylabel('Прогнозовані Рейтинги')

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 program

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  
from sklearn.preprocessing import OneHotEncoder  
  
*# Вмикаємо інтерактивний режим для графіків*plt.ion()  
  
*# --- Завантаження та попередня обробка даних ---  
usecols* = ['Name', 'Rating', 'PublishYear', 'PublisherNaming']  
*# Для тестування завантажуємо лише 10000 рядків  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';', usecols=*usecols*, nrows=10000)  
  
*df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
*df* = *df*.dropna(subset=['Rating'])  
*df*['RatingClass'] = np.where(*df*['Rating'] >= 4.5, 'High', 'Low')  
  
*# --- Обробка PublisherNaming ---  
df*['PublisherNaming'] = *df*['PublisherNaming'].fillna("Unknown")  
*min\_freq* = 10  
*pub\_counts* = *df*['PublisherNaming'].value\_counts()  
*df*['PublisherNaming'] = *df*['PublisherNaming'].apply(lambda *x*: *x* if *pub\_counts*[*x*] >= *min\_freq* else 'Other')  
  
*# --- Формування ознак ---  
imputer* = SimpleImputer(strategy='mean')  
*df*[['PublishYear']] = *imputer*.fit\_transform(*df*[['PublishYear']])  
  
*encoder* = OneHotEncoder(sparse\_output=False, handle\_unknown='ignore')  
*publisher\_encoded* = *encoder*.fit\_transform(*df*[['PublisherNaming']])  
*publisher\_encoded\_df* = pd.DataFrame(*publisher\_encoded*,  
 columns=*encoder*.get\_feature\_names\_out(['PublisherNaming']),  
 index=*df*.index)  
  
*X* = pd.concat([*df*[['PublishYear']], *publisher\_encoded\_df*], axis=1)  
*y* = *df*['RatingClass']  
  
*X\_train*, *X\_test*, *y\_train*, *y\_test* = train\_test\_split(*X*, *y*, test\_size=0.2, random\_state=42)  
  
*# --- Побудова моделей класифікації ---  
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*)  
  
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']))  
  
*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*)  
  
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']))  
  
  
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 Class")  
 plt.ylabel("Actual Class")  
 plt.show()  
  
  
plot\_confusion\_matrix(*cm\_log*, "Confusion Matrix - Logistic Regression")  
plot\_confusion\_matrix(*cm\_tree*, "Confusion Matrix - Decision Tree")  
  
plt.figure(figsize=(6, 4))  
sns.countplot(data=*df*, x='RatingClass', palette='viridis')  
plt.title("Кількість книг за класами RatingClass")  
plt.xlabel("Клас рейтингу")  
plt.ylabel("Кількість книг")  
plt.show()  
  
plt.figure(figsize=(10, 6))  
sns.countplot(data=*df*, x='PublishYear', hue='RatingClass', palette='viridis')  
plt.title("Розподіл класів за роками публікації")  
plt.xlabel("Рік публікації")  
plt.ylabel("Кількість книг")  
plt.xticks(rotation=45)  
plt.legend(title='Rating Class')  
plt.show()  
  
print("\nПерші 10 рядків даних з колонкою 'RatingClass':")  
print(*df*[['Name', 'PublishYear', 'PublisherNaming', 'Rating', 'RatingClass']].head(10))  
  
*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

Connected to pydev debugger (build 233.15619.17)

=== Logistic Regression Results ===

Accuracy: 0.93

Confusion Matrix (Logistic Regression):

[[ 0 141]

[ 0 1859]]

Classification Report (Logistic Regression):

precision recall f1-score support

High 0.00 0.00 0.00 141

Low 0.93 1.00 0.96 1859

accuracy 0.93 2000

macro avg 0.46 0.50 0.48 2000

weighted avg 0.86 0.93 0.90 2000

=== Decision Tree Classifier Results ===

Accuracy: 0.92

Confusion Matrix (Decision Tree):

[[ 1 140]

[ 12 1847]]

Classification Report (Decision Tree):

precision recall f1-score support

High 0.08 0.01 0.01 141

Low 0.93 0.99 0.96 1859

accuracy 0.92 2000

macro avg 0.50 0.50 0.49 2000

weighted avg 0.87 0.92 0.89 2000

Перші 10 рядків даних з колонкою 'RatingClass':

Name ... RatingClass

0 Transformation of Knowledge: Early Manuscripts... ... High

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

2 The Thief, the Cross and the Wheel: Pain and t... ... Low

3 Lost Ramessid and Post-Ramessid Private Tombs ... ... Low

4 Applied Math for Food Service ... Low

5 A Soul Remembers Hiroshima ... High

6 Tornado ... Low

7 She: Understanding Feminine Psychology ... Low

8 Imogene's Antlers ... Low

9 Computer Systems Organization and Architecture ... Low

[10 rows x 5 columns]

Результати класифікації збережено у файл 'classification\_results.csv'

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 a number of years

Description automatically generated with medium confidence

Рис.5. Розподіл класів по рокам

A screenshot of a computer screen

Description automatically generated

Рис.6. Результат класифікації

**Task 3**

Класифікація місяця за CountsOfReview, Rating, PublishYear

Метою даної задачі є побудова класифікаційної моделі, яка на основі характеристик книги – кількості відгуків (CountsOfReview), рейтингу (Rating) та року публікації (PublishYear) – дозволяє прогнозувати місяць публікації (PublishMonth). Аналіз проводиться на сучасних даних, зокрема, на книгах, опублікованих з 2010 року і пізніше. Цей підхід допомагає з’ясувати, як параметри популярності, якості та часу впливають на розподіл книг за місяцями.

### 2. Попередня обробка даних

**Джерело даних:**  
Дані завантажуються з CSV-файлу, який містить інформацію про книги, включаючи назву (Name), кількість відгуків, місяць публікації, рік публікації та рейтинг.

**Обробка та фільтрація:**

* Видаляються записи з пропущеними критичними ознаками (Name, CountsOfReview, PublishMonth, PublishYear, Rating).
* Поля PublishYear та PublishMonth перетворюються на цілі числа.
* Рейтинг обробляється шляхом заміни ком на крапки та перетворення у тип float.
* Для аналізу обираються лише книги, опубліковані з 2010 року і пізніше.

### 3. Формування ознак та підготовка даних

**Цільова змінна:**

* PublishMonth (місяць публікації) – класова змінна, що має 12 можливих значень (від 1 до 12).

**Ознаки для класифікації:**

* CountsOfReview: кількість відгуків, що відображає популярність книги.
* Rating: рейтинг книги, що характеризує її якість або сприйняття аудиторією.
* PublishYear: рік публікації, який враховує ринкові тенденції.

**Підготовка даних:**

* Дані заповнюються за допомогою SimpleImputer (стратегія – середнє значення).
* Ознаки масштабуються за допомогою StandardScaler для покращення роботи моделі.

### 4. Розподіл даних на тренувальну та тестову вибірки

Дані розділяються на тренувальну (80%) та тестову (20%) вибірки зі стратифікацією за місяцем публікації. При цьому зберігаються індекси тестової вибірки, що дозволяє відновити назви книг для звітності.

### 5. Навчання класифікаційної моделі

**Модель:**  
Використовується RandomForestClassifier з 100 деревами. Ця модель була обрана через її здатність ефективно працювати з неструктурованими даними та забезпечувати високу точність, а також дозволяти оцінити важливість ознак.

**Процес навчання:**  
Модель навчається на тренувальній вибірці, після чого на тестовій проводиться прогнозування місяця публікації.

### 6. Оцінка якості моделі

**Базові метрики:**

* **Accuracy:** Загальна точність моделі.
* **Classification Report:** Precision, Recall та F1-score для кожного класу (місяця).
* **Confusion Matrix:** Матриця плутанини, що показує, як модель розподіляє 12 класів.

**Додаткові метрики:**

* Macro, micro та weighted F1-score, Precision і Recall – дозволяють оцінити якість моделі з різних точок зору.
* **Cohen's Kappa:** Оцінка узгодженості між фактичними та прогнозованими класами з урахуванням випадкової згоди.
* **Log Loss:** Вимірює якість прогнозованих ймовірностей.

Результати свідчать про те, наскільки добре модель розрізняє 12 місяців публікації на основі заданих ознак.

### 7. Запис результатів у файл

Результати класифікації зберігаються у CSV-файл "results\_classification.csv". Цей файл містить:

* **Name:** Назва книги.
* **Actual\_PublishMonth:** Фактичний місяць публікації.
* **Predicted\_PublishMonth:** Прогнозований моделью місяць публікації.

Це дозволяє зручно переглядати, які книги були правильно класифіковані, а які – ні.

### 8. Візуалізація результатів

**Матриця плутанини:**  
За допомогою теплової карти побудована матриця плутанини, що показує, як добре модель розрізняє 12 класів (місяців). Вона допомагає виявити, які місяці прогнозуються коректно, а де виникають помилки.

**Важливість ознак:**  
Графік важливості ознак, побудований за результатами RandomForest, демонструє, які з параметрів (CountsOfReview, Rating, PublishYear) найбільш впливають на прогнозування. Це дозволяє зрозуміти, які характеристики є критично важливими для класифікації.

### 9. Основні висновки та інтерпретація

* **CountsOfReview** є ключовим індикатором популярності книги. Книги з високою кількістю відгуків можуть бути пов’язані з певними місяцями публікації (наприклад, періодами активного продажу).
* **Rating** відображає якість книги, що також впливає на розподіл за місяцями.
* **PublishYear** дозволяє врахувати часові тенденції, оскільки нові книги можуть мати різні ринкові характеристики порівняно зі старішими.

За результатами аналізу важливості ознак видно, яка ознака має найбільший вплив на модель, що допомагає видавцям та маркетологам орієнтувати рекламні кампанії та планувати майбутні випуски. Матриця плутанини підтверджує, що модель має достатню здатність розрізняти 12 класів, хоча завжди є можливості для покращення.

Цей підхід дозволяє не тільки отримати числові результати, але й зрозуміти, які фактори впливають на місяць публікації, що може бути корисним для стратегічного планування в видавничому бізнесі.

**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.preprocessing import StandardScaler  
from sklearn.impute import SimpleImputer  
from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import f1\_score, precision\_score, recall\_score, cohen\_kappa\_score, log\_loss  
  
*# ==============================  
# 1. Завантаження та попередня обробка даних  
# ==============================  
# Завантаження даних із CSV-файлу (переконайтеся, що файл містить: Name, CountsOfReview, PublishMonth, PublishYear, Rating)  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Видаляємо записи з пропущеними критичними ознаками  
df* = *df*.dropna(subset=['Name', 'CountsOfReview', 'PublishMonth', 'PublishYear', 'Rating'])  
  
*# Перетворюємо PublishYear та PublishMonth у цілі числа  
df*['PublishYear'] = *df*['PublishYear'].astype(int)  
*df*['PublishMonth'] = *df*['PublishMonth'].astype(int)  
  
*# Обробка рейтингу: замінюємо кому на крапку та перетворюємо у float  
df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
  
*# ==============================  
# 2. Формування набору ознак та цільової змінної  
# ==============================  
# Цільова змінна: PublishMonth (класи від 1 до 12)  
y* = *df*['PublishMonth']  
  
*# Ознаки для класифікації: CountsOfReview, Rating, PublishYear  
X* = *df*[['CountsOfReview', 'Rating', 'PublishYear']].copy()  
  
*# Заповнюємо можливі пропуски (якщо є)  
imputer* = SimpleImputer(strategy='mean')  
*X* = pd.DataFrame(*imputer*.fit\_transform(*X*), columns=*X*.columns, index=*X*.index)  
  
*# Масштабування ознак  
scaler* = StandardScaler()  
*X\_scaled* = *scaler*.fit\_transform(*X*)  
  
*# ==============================  
# 3. Розподіл даних на тренувальну та тестову вибірки (з поверненням індексів)  
# ==============================  
X\_train*, *X\_test*, *y\_train*, *y\_test*, *idx\_train*, *idx\_test* = train\_test\_split(  
 *X\_scaled*, *y*, *df*.index, test\_size=0.2, random\_state=42, stratify=*y*)  
  
*# ==============================  
# 4. Навчання класифікаційної моделі  
# ==============================  
clf* = RandomForestClassifier(n\_estimators=100, random\_state=42)  
*clf*.fit(*X\_train*, *y\_train*)  
  
*# Прогнозування на тестовій вибірці  
y\_pred* = *clf*.predict(*X\_test*)  
*y\_pred\_proba* = *clf*.predict\_proba(*X\_test*) *# для log loss  
  
# Оцінка якості моделі  
accuracy* = accuracy\_score(*y\_test*, *y\_pred*)  
print("=== Результати класифікації ===")  
print(f"Accuracy: {*accuracy*:.4f}")  
print("\nClassification Report:")  
print(classification\_report(*y\_test*, *y\_pred*))  
print("\nConfusion Matrix:")  
*cm* = confusion\_matrix(*y\_test*, *y\_pred*)  
print(*cm*)  
  
*# Додаткові метрики  
macro\_f1* = f1\_score(*y\_test*, *y\_pred*, average='macro')  
*micro\_f1* = f1\_score(*y\_test*, *y\_pred*, average='micro')  
*weighted\_f1* = f1\_score(*y\_test*, *y\_pred*, average='weighted')  
  
*macro\_precision* = precision\_score(*y\_test*, *y\_pred*, average='macro')  
*micro\_precision* = precision\_score(*y\_test*, *y\_pred*, average='micro')  
*weighted\_precision* = precision\_score(*y\_test*, *y\_pred*, average='weighted')  
  
*macro\_recall* = recall\_score(*y\_test*, *y\_pred*, average='macro')  
*micro\_recall* = recall\_score(*y\_test*, *y\_pred*, average='micro')  
*weighted\_recall* = recall\_score(*y\_test*, *y\_pred*, average='weighted')  
  
*kappa* = cohen\_kappa\_score(*y\_test*, *y\_pred*)  
*ll* = log\_loss(*y\_test*, *y\_pred\_proba*)  
  
print("\nДодаткові метрики:")  
print(f"Macro F1-score: {*macro\_f1*:.4f}")  
print(f"Micro F1-score: {*micro\_f1*:.4f}")  
print(f"Weighted F1-score: {*weighted\_f1*:.4f}")  
print(f"Macro Precision: {*macro\_precision*:.4f}")  
print(f"Micro Precision: {*micro\_precision*:.4f}")  
print(f"Weighted Precision: {*weighted\_precision*:.4f}")  
print(f"Macro Recall: {*macro\_recall*:.4f}")  
print(f"Micro Recall: {*micro\_recall*:.4f}")  
print(f"Weighted Recall: {*weighted\_recall*:.4f}")  
print(f"Cohen's Kappa: {*kappa*:.4f}")  
print(f"Log Loss: {*ll*:.4f}")  
  
*# ==============================  
# 5. Запис результатів класифікації у файл  
# ==============================  
results\_df* = pd.DataFrame({  
 'Name': *df*.loc[*idx\_test*, 'Name'],  
 'Actual\_PublishMonth': *y\_test*,  
 'Predicted\_PublishMonth': *y\_pred*})  
*results\_df*.to\_csv('results\_classification.csv', index=False, encoding='utf-8-sig')  
print("\nРезультати класифікації з назвою роботи, фактичним та прогнозованим місяцем збережено у файл 'results\_classification.csv'.")  
  
*# ==============================  
# 6. Візуалізація результатів  
# ==============================  
# 6.1 Матриця плутанини*plt.figure(figsize=(10, 8))  
sns.heatmap(*cm*, annot=True, fmt='d', cmap='Blues',  
 xticklabels=sorted(*y*.unique()), yticklabels=sorted(*y*.unique()))  
plt.xlabel("Прогнозований місяць")  
plt.ylabel("Фактичний місяць")  
plt.title("Матриця плутанини для класифікації місяців публікації")  
plt.show()  
  
*# 6.2 Візуалізація важливості ознак  
importances* = *clf*.feature\_importances\_  
*feature\_names* = *X*.columns  
*indices* = np.argsort(*importances*)[::-1]  
  
plt.figure(figsize=(8, 6))  
sns.barplot(x=*importances*[*indices*], hue=*importances*[*indices*], y=[*feature\_names*[*i*] for *i* in *indices*], palette='viridis')  
plt.title("Важливість ознак")  
plt.xlabel("Важливість")  
plt.ylabel("Ознака")  
plt.show()  
  
*# 6.3 Інтерпретація результатів класифікації*print("\n=== Інтерпретація результатів класифікації ===")  
print("Модель класифікує книги за місяцем публікації на основі наступних параметрів:")  
print(" - CountsOfReview: кількість відгуків (популярність книги).")  
print(" - Rating: рейтинг книги (якість сприйняття).")  
print(" - PublishYear: рік публікації (тенденції та тренди ринку).")  
print("\nВажливість ознак:")  
for *i*, *name* in enumerate(*feature\_names*[*indices*]):  
 print(f" {*i*+1}. {*name*}: важливість = {*importances*[*indices*][*i*]:.3f}")  
print("\nМатриця плутанини демонструє, наскільки добре модель розрізняє 12 класів (місяців).")

Лістинг та результати роботи

=== Результати класифікації ===

Accuracy: 0.1223

Classification Report:

precision recall f1-score support

1 0.18 0.43 0.26 8761

2 0.10 0.06 0.08 4268

3 0.10 0.07 0.08 5244

4 0.10 0.08 0.09 5307

5 0.08 0.06 0.07 4925

6 0.08 0.06 0.07 4857

7 0.07 0.04 0.05 4125

8 0.08 0.06 0.07 4666

9 0.11 0.10 0.10 6093

10 0.10 0.10 0.10 5927

11 0.08 0.06 0.07 4684

12 0.10 0.10 0.10 5518

accuracy 0.12 64375

macro avg 0.10 0.10 0.09 64375

weighted avg 0.10 0.12 0.11 64375

Confusion Matrix:

[[3744 325 489 464 385 420 286 399 608 626 333 682]

[1285 271 322 287 246 229 155 204 344 355 226 344]

[1613 245 380 377 312 267 207 284 465 447 246 401]

[1512 248 348 416 319 286 224 267 504 475 286 422]

[1418 215 328 333 295 270 231 276 452 458 266 383]

[1543 206 303 301 253 275 230 261 419 400 275 391]

[1266 165 253 253 230 251 179 229 379 374 209 337]

[1270 225 292 312 286 270 215 282 431 437 258 388]

[1748 241 361 389 359 339 286 364 617 546 329 514]

[1691 270 321 387 366 306 266 342 582 586 314 496]

[1406 195 251 310 250 264 176 275 425 456 273 403]

[2046 207 270 289 255 276 207 257 442 460 256 553]]

Додаткові метрики:

Macro F1-score: 0.0949

Micro F1-score: 0.1223

Weighted F1-score: 0.1056

Macro Precision: 0.0991

Micro Precision: 0.1223

Weighted Precision: 0.1049

Macro Recall: 0.1017

Micro Recall: 0.1223

Weighted Recall: 0.1223

Cohen's Kappa: 0.0264

Log Loss: 9.5025

Результати класифікації з назвою роботи, фактичним та прогнозованим місяцем збережено у файл 'results\_classification.csv'.

=== Інтерпретація результатів класифікації ===

Модель класифікує книги за місяцем публікації на основі наступних параметрів:

- CountsOfReview: кількість відгуків (популярність книги).

- Rating: рейтинг книги (якість сприйняття).

- PublishYear: рік публікації (тенденції та тренди ринку).

Важливість ознак:

1. Rating: важливість = 0.548

2. CountsOfReview: важливість = 0.283

3. PublishYear: важливість = 0.169

Матриця плутанини демонструє, наскільки добре модель розрізняє 12 класів (місяців).

A screenshot of a computer

Description automatically generated

Рис.7. Результат класифікації

A table with numbers and a number on it

Description automatically generated

Рис.8. Confusion Matrix для класифікатора

A graph with different colored squares

Description automatically generated

Рис.9. Розподіл важливості ознак

**Task 4**

Побудова кореляційної матриці між параметрами даних

**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=';')

* **Що робиться:**  
  Завантажується датасет із CSV‑файлу, де стовпці розділені символом «;».
* **Мета:**  
  Отримати дані для подальшого аналізу.

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

* **Що робиться:**  
  Значення в стовпці Rating обробляються: коми замінюються на крапки (щоб десяткові значення були коректно розпізнані) і дані перетворюються у тип float.
* **Мета:**  
  Забезпечити коректне числове представлення рейтингу.

numeric\_cols = ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth']

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

df = df.dropna(subset=numeric\_cols)

* **Що робиться:**
  + Стовпці, які є критично важливими для аналізу, перетворюються у числовий тип (якщо дані могли бути прочитані як рядки).
  + Рядки, де хоча б одна з цих змінних має пропущене значення, видаляються.
* **Мета:**  
  Забезпечити, щоб усі обрані для аналізу числові змінні мали правильний тип та не містили пропусків.

### 2. Додаткові характеристики

df['TitleLength'] = df['Name'].fillna("").apply(len)

* **Що робиться:**  
  Для кожного запису обчислюється довжина назви книги. Якщо значення в стовпці Name пропущене, воно замінюється на порожній рядок.
* **Мета:**  
  Отримати числову характеристику, що може відображати стиль, жанрові особливості або маркетингові тренди (наприклад, короткі або довгі назви).

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

* **Що робиться:**  
  Для стовпця PublisherNaming (назва видавництва) заповнюються пропуски за допомогою моди (найчастіше зустрічаємого значення).
* **Мета:**  
  Забезпечити повноту даних для подальшого аналізу залежностей за видавництвом.

### 3. Обчислення кореляційних матриць для числових змінних

#### 3.1 Створення розширеного набору числових характеристик

num\_features = ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth', 'TitleLength']

* **Що робиться:**  
  Створюється список змінних, за якими буде обчислено кореляцію. До цього набору входять стандартні числові ознаки та додаткова характеристика довжини назви книги.
* **Мета:**  
  Вивчити взаємозв’язки між усіма ключовими числовими показниками.

#### 3.2 Обчислення Pearson кореляції

pearson\_corr = df[num\_features].corr(method='pearson')

* **Що робиться:**  
  Обчислюється матриця кореляції методом Pearson, яка вимірює лінійну залежність між змінними.
* **Мета:**  
  Виявити, чи існує сильний лінійний зв’язок між будь-якими парами змінних.

#### 3.3 Обчислення Spearman кореляції

spearman\_corr = df[num\_features].corr(method='spearman')

* **Що робиться:**  
  Обчислюється матриця кореляції методом Spearman, що вимірює монотонну залежність (не обов’язково лінійну).
* **Мета:**  
  Отримати інформацію про залежності, які можуть бути нелінійними, але все одно монотонними.

#### 3.4 Обчислення матриці взаємної інформації (Mutual Information)

mi\_matrix = pd.DataFrame(index=num\_features, columns=num\_features)

for col1 in num\_features:

for col2 in num\_features:

mi = mutual\_info\_regression(df[[col1]], df[col2], random\_state=42)

mi\_matrix.loc[col1, col2] = mi[0]

mi\_matrix = mi\_matrix.astype(float)

* **Що робиться:**  
  Для кожної пари числових змінних обчислюється взаємна інформація, що є мірою залежності між ними, яка може виявити нелінійні зв’язки.
* **Мета:**  
  Додатково оцінити залежності, які можуть бути невловимими для традиційних методів кореляції.

### 4. Аналіз PublisherNaming (категорійна змінна)

#### 4.1 Використання χ²‑тесту

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"])

ct = pd.crosstab(df['PublisherNaming'], df['CountsOfReviewCategory'])

chi2, p\_val, dof, expected = chi2\_contingency(ct)

* **Що робиться:**
  + Значення CountsOfReview розбиваються на категорії за допомогою функції pd.cut.
  + Побудовується крос-таблиця між видавництвом та категоріями кількості відгуків.
  + Застосовується χ²‑тест для визначення статистичної залежності між PublisherNaming та категоріями кількості відгуків.
* **Мета:**  
  Перевірити, чи існує статистично значуща залежність між видавництвом та популярністю книги (за категоріями відгуків).

#### 4.2 Взаємна інформація для PublisherNaming

le = LabelEncoder()

df['PublisherEncoded'] = le.fit\_transform(df['PublisherNaming'])

mi\_publisher = {}

for feature in ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth', 'TitleLength']:

mi = mutual\_info\_regression(df[[feature]], df['PublisherEncoded'], random\_state=42)

mi\_publisher[feature] = mi[0]

* **Що робиться:**
  + За допомогою LabelEncoder категоріальна змінна PublisherNaming перетворюється на числовий формат.
  + Для кожної числової характеристики обчислюється взаємна інформація між нею та видавництвом.
* **Мета:**  
  Оцінити, наскільки PublisherNaming впливає на інші характеристики датасету.

### 5. Візуалізація кореляційних матриць та парних залежностей

#### 5.1 Теплові карти для матриць кореляцій

plt.figure(figsize=(10, 8))

sns.heatmap(pearson\_corr, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

plt.title('Pearson Correlation Matrix (Extended)')

plt.tight\_layout()

plt.show()

plt.figure(figsize=(10, 8))

sns.heatmap(spearman\_corr, annot=True, cmap='coolwarm', vmin=-1, vmax=1)

plt.title('Spearman Correlation Matrix (Extended)')

plt.tight\_layout()

plt.show()

plt.figure(figsize=(10, 8))

sns.heatmap(mi\_matrix, annot=True, cmap='viridis')

plt.title('Mutual Information Matrix (Extended)')

plt.tight\_layout()

plt.show()

* **Що робиться:**  
  Побудовано три теплові карти, що наочно демонструють коефіцієнти Pearson і Spearman кореляцій, а також матрицю взаємної інформації.
* **Мета:**  
  Надати візуальне уявлення про силу та напрямок залежностей між числовими змінними.

#### 5.2 Pair Plot

sns.pairplot(df[num\_features])

plt.suptitle('Pair Plot for Extended Numeric Features', y=1.02)

plt.show()

* **Що робиться:**  
  Створюється pair plot для всіх числових характеристик. Цей графік показує діаграми розподілу для кожної змінної та діаграми розсіювання для кожної пари змінних.
* **Мета:**  
  Візуально оцінити парні залежності та можливі закономірності між характеристиками.  
  Зауваження: Якщо даних дуже багато, генерація pair plot може займати багато часу, тому можна обмежити вибірку.

### Висновки

1. **Кореляції між характеристиками:**
   * **Pearson та Spearman:** Матриці показують, наскільки сильно лінійно або монотонно пов’язані такі параметри, як кількість відгуків, рейтинг, рік та місяць публікації, а також довжина назви книги.
   * **Mutual Information:** Виявляє нелінійні залежності між змінними.
2. **Аналіз PublisherNaming:**
   * За допомогою χ²‑тесту було перевірено статистичну залежність між видавництвом і категоріями кількості відгуків.
   * Взаємна інформація з числовими характеристиками дає уявлення про вплив видавництва на інші показники.
3. **Візуальна презентація:**
   * Теплові карти та pair plot дозволяють наочно оцінити залежності між характеристиками, що важливо для подальшого сегментування та прогнозування.
   * Отримані графіки можна використовувати для прийняття бізнес-рішень, таких як оптимізація рекламних кампаній або планування нових випусків.

**Код програми**

import pandas as *pd*import numpy as *np*import matplotlib.pyplot as *plt*import seaborn as *sns*from scipy.stats import chi2\_contingency, spearmanr  
from sklearn.feature\_selection import mutual\_info\_regression  
from sklearn.preprocessing import LabelEncoder  
  
*# ------------------------------  
# 1. Завантаження та попередня обробка даних  
# ------------------------------  
# Завантаження даних із CSV-файлу  
df* = pd.read\_csv('CSV\_BI\_Lab1\_data\_source.csv', sep=';')  
  
*# Обробка рейтингу: заміна ком на крапки та перетворення в float  
df*['Rating'] = *df*['Rating'].replace({',': '.'}, regex=True).astype(float)  
  
*# Перетворення важливих колонок у числовий формат  
numeric\_cols* = ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth']  
*df*[*numeric\_cols*] = *df*[*numeric\_cols*].apply(pd.to\_numeric, errors='coerce')  
  
*# Видаляємо рядки з пропущеними значеннями для цих колонок  
df* = *df*.dropna(subset=*numeric\_cols*)  
  
*# ------------------------------  
# 2. Додаткові характеристики  
# ------------------------------  
# 2.1 Довжина назви книги (TitleLength)  
df*['TitleLength'] = *df*['Name'].fillna("").apply(len)  
  
*# 2.2 Для видавництва (PublisherNaming) переконаємося, що немає пропусків  
df*['PublisherNaming'] = *df*['PublisherNaming'].fillna(*df*['PublisherNaming'].mode()[0])  
  
*# ------------------------------  
# 3. Обчислення кореляційних матриць для числових змінних  
# ------------------------------  
# Розширений набір числових характеристик  
num\_features* = ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth', 'TitleLength']  
  
*# 3.1 Pearson кореляція  
pearson\_corr* = *df*[*num\_features*].corr(method='pearson')  
print("Pearson Correlation Matrix (Extended):")  
print(*pearson\_corr*)  
  
*# 3.2 Spearman кореляція  
spearman\_corr* = *df*[*num\_features*].corr(method='spearman')  
print("\nSpearman Correlation Matrix (Extended):")  
print(*spearman\_corr*)  
  
*# 3.3 Mutual Information (для нелінійних залежностей)  
mi\_matrix* = pd.DataFrame(index=*num\_features*, columns=*num\_features*)  
for *col1* in *num\_features*:  
 for *col2* in *num\_features*:  
 *mi* = mutual\_info\_regression(*df*[[*col1*]], *df*[*col2*], random\_state=42)  
 *mi\_matrix*.loc[*col1*, *col2*] = *mi*[0]  
*mi\_matrix* = *mi\_matrix*.astype(float)  
print("\nMutual Information Matrix (Extended):")  
print(*mi\_matrix*)  
  
*# ------------------------------  
# 4. Аналіз PublisherNaming (категорійна змінна)  
# ------------------------------  
# Використаємо χ²-тест для перевірки залежності між PublisherNaming та, наприклад, категоріями кількості відгуків.  
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"])  
*ct* = pd.crosstab(*df*['PublisherNaming'], *df*['CountsOfReviewCategory'])  
*chi2*, *p\_val*, *dof*, *expected* = chi2\_contingency(*ct*)  
print("\nChi-squared test between PublisherNaming and CountsOfReviewCategory:")  
print(f"Chi-squared: {*chi2*:.2f}, P-value: {*p\_val*:.4f}")  
  
*# Також можна обчислити взаємну інформацію між PublisherNaming та іншими числовими змінними,  
# спершу проведемо Label Encoding для PublisherNaming  
le* = LabelEncoder()  
*df*['PublisherEncoded'] = *le*.fit\_transform(*df*['PublisherNaming'])  
*mi\_publisher* = {}  
for *feature* in ['CountsOfReview', 'Rating', 'PublishYear', 'PublishMonth', 'TitleLength']:  
 *mi* = mutual\_info\_regression(*df*[[*feature*]], *df*['PublisherEncoded'], random\_state=42)  
 *mi\_publisher*[*feature*] = *mi*[0]  
print("\nMutual Information between PublisherNaming and numeric features:")  
print(*mi\_publisher*)  
  
*# ------------------------------  
# 5. Візуалізація кореляційних матриць та парних залежностей  
# ------------------------------*plt.figure(figsize=(10, 8))  
sns.heatmap(*pearson\_corr*, annot=True, cmap='coolwarm', vmin=-1, vmax=1)  
plt.title('Pearson Correlation Matrix (Extended)')  
plt.tight\_layout()  
plt.show()  
  
plt.figure(figsize=(10, 8))  
sns.heatmap(*spearman\_corr*, annot=True, cmap='coolwarm', vmin=-1, vmax=1)  
plt.title('Spearman Correlation Matrix (Extended)')  
plt.tight\_layout()  
plt.show()  
  
plt.figure(figsize=(10, 8))  
sns.heatmap(*mi\_matrix*, annot=True, cmap='viridis')  
plt.title('Mutual Information Matrix (Extended)')  
plt.tight\_layout()  
plt.show()

Лістинг та результати роботи

Pearson Correlation Matrix (Extended):

CountsOfReview Rating ... PublishMonth TitleLength

CountsOfReview 1.000000 0.029469 ... 0.000254 -0.033354

Rating 0.029469 1.000000 ... 0.028181 -0.117861

PublishYear 0.027245 0.063448 ... 0.025060 0.033002

PublishMonth 0.000254 0.028181 ... 1.000000 0.001346

TitleLength -0.033354 -0.117861 ... 0.001346 1.000000

[5 rows x 5 columns]

Spearman Correlation Matrix (Extended):

CountsOfReview Rating ... PublishMonth TitleLength

CountsOfReview 1.000000 0.219715 ... 0.022108 -0.180757

Rating 0.219715 1.000000 ... 0.024683 -0.023168

PublishYear 0.133969 0.018285 ... -0.008807 0.038829

PublishMonth 0.022108 0.024683 ... 1.000000 0.005145

TitleLength -0.180757 -0.023168 ... 0.005145 1.000000

[5 rows x 5 columns]

Mutual Information Matrix (Extended):

CountsOfReview Rating ... PublishMonth TitleLength

CountsOfReview 2.868436 0.268690 ... 0.007086 0.018916

Rating 0.267621 4.587653 ... 0.004906 0.021107

PublishYear 0.012535 0.009206 ... 0.030276 0.003006

PublishMonth 0.006827 0.006825 ... 2.463545 0.000487

TitleLength 0.020534 0.021429 ... 0.000565 4.604793

[5 rows x 5 columns]

Chi-squared test between PublisherNaming and CountsOfReviewCategory:

Chi-squared: 85099.15, P-value: 0.0000

Mutual Information between PublisherNaming and numeric features:

{'CountsOfReview': np.float64(0.21483633370941035), 'Rating': np.float64(0.20455304509410333), 'PublishYear': np.float64(0.267398331443748), 'PublishMonth': np.float64(0.12098122060613381), 'TitleLength': np.float64(0.11317918426782203)}

A screenshot of a graph

Description automatically generated

Рис.10. Матриця кореляцій Пірсона

A screenshot of a graph

Description automatically generated

Рис.11. Кореляційна матриця Спірмана

A screenshot of a computer screen

Description automatically generated

Рис.12. Кореляційна матриця взаємної інформації (для нелінійних взаємозв’язків між змінними)

**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.