СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

КУРСОВ ПРОЕКТ
ПО СИСТЕМИ, ОСНОВАНИ НА ЗНАНИЯ

Тема:

Система за генериране на препоръка за закупуване на книги

Студенти:
Александра Георгиева Бамбалдокова, III гр., ф.н 72088
Антон Мариов Янчев, III гр., ф.н 72086
Мария Димитрова Червенкова, III гр., ф.н. 72098

София, януари 2023 г.

1. Формулировка на задачата

Реализиране на система за генериране на препоръка за закупуване на книги. Системата трябва по дадена книга от използвания набор от данни да направи предложения на потребителя за подходящи книги, базирани на рейтинги от всички потребители.

2. Използвани алгоритми

Проучване и запознаване с темата: [1] *How Did We Build Book Recommender Systems in An Hour Part 2 — k Nearest Neighbors and Matrix Factorization*.

Набор от данни: [2] *Book-Crossing Dataset*.

За решение на задачата използваме к-най-близки съседи.

[3] Алгоритъмът k-най-близки съседи (KNN) е метод за класифициране на данни за оценка на вероятността точка от данни да стане член на една или друга група въз основа на групата, към която принадлежат най-близките до нея точки от данни.

Алгоритъмът на k-най-близкия съсед е вид алгоритъм за контролирано машинно обучение, използван за решаване на проблеми с класификация и регресия. Въпреки това, той се използва главно за проблеми с класификацията. KNN е мързелив и непараметричен алгоритъм.

Нарича се алгоритъм за мързеливо обучение или мързелив учащ се, защото не извършва никакво обучение, когато предоставяте данните за обучение. Вместо това, той просто съхранява данните по време на обучението и не извършва никакви изчисления. Той не изгражда модел, докато не бъде извършено запитване към набора от данни. Това прави KNN идеален за извличане на данни.

Смята се за непараметричен метод, тъй като не прави никакви предположения относно основното разпределение на данните. Просто казано, KNN се опитва да определи към коя група принадлежи точка от данни, като разглежда точките от данни около нея.

3. Описание на програмната реализация

❖ Представяне на набора от данни

Наборът от данни *Book-Crossing* се състои от 3 таблици.

BX-Users

Съдържа потребителите, които се представят чрез User-ID, Location, Age. Потребителските идентификатори („User-ID“) са анонимизирани и се съпоставят с

цели числа. Предоставят се демографски данни („Местоположение“, „Възраст“), ако са налични. В противен случай тези полета съдържат NULL-стойности.

```
users = pd.read_csv('data/BX-Users.csv', sep=";", on_bad_lines='skip', encoding='latin-1', low_memory=False)
print(users.shape)
print(users.columns)
```

```
(278858, 3)
Index(['User-ID', 'Location', 'Age'], dtype='object')
```

Таблицата изглежда по следния начин (първите 5 реда за пример):

	User-ID	Location	Age
0	1	nyc, new york, usa	NaN
1	2	stockton, california, usa	18.0
2	3	moscow, yukon territory, russia	NaN
3	4	porto, v.n.gaia, portugal	17.0
4	5	farnborough, hants, united kingdom	NaN

BX-Books

Книгите се идентифицират със съответния им ISBN номер. Невалидните ISBN номера вече са премахнати от набора от данни. Освен това е дадена информация, базирана на съдържание („Book-Title “, „Book-Author“, „Year-Of-Publication “, „Publisher“), получена от Amazon Web Services. Имайте предвид, че в случай на няколко автора се предоставя само първият. Дадени са и URL адреси, водещи към изображения на корицата, които се появяват в три различни вида („URL-Image-S“, „URL-Image-M“, „URL-Image-L“), т.е. малък, среден, голям. Тези URL адреси сочат към уеб сайта на Amazon.

```
books = pd.read_csv('data/BX-Books.csv', sep=";", on_bad_lines='skip', encoding='latin-1', low_memory=False)
print(books.shape)
print(books.columns)
```

```
(271360, 8)
Index(['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher',
       'Image-URL-S', 'Image-URL-M', 'Image-URL-L'],
      dtype='object')
```

За целта на проекта премахваме колоните с Image-URL, понеже не са необходими за системата. Също така преименуваме колоните с по-кратки имена за прегледност.

```
# Leaving only columns needed for analysis
books = books[['ISBN', 'Book-Title', 'Book-Author', 'Year-Of-Publication', 'Publisher']]

# Renaming column names
books.rename(columns={"Book-Title": 'Title',
                      'Book-Author': 'Author',
                      "Year-Of-Publication": 'Year',
                      "Publisher": "Publisher"}, inplace=True)
```

Преработената таблица изглежда по следния начин (първите 5 реда за пример):

	ISBN	Title	Author	Year	Publisher
0	0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press
1	0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada
2	0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux
4	0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company

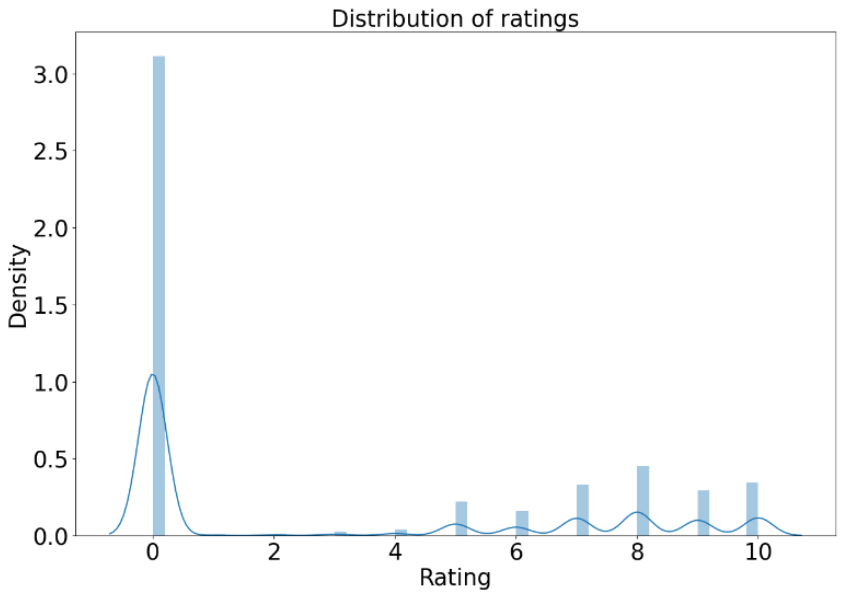
BX-Book-Ratings

Съдържа информация за отзив на книгата. Отзивите (`Book-Rating`) са или изрични, изразени по скала от 1-10 (по-високите стойности означават по-висок отзив), или имплицитни, изразени с 0.

Таблицата изглежда по следния начин (първите 5 реда за пример):

	User-ID	ISBN	Book-Rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6

Разпределение на отзивите на всички потребители:



Забелязваме, че отзивите са разпределени неравномерно и мнозинството от тях са 0. По тази причина за по-добри резултати в системата премахваме всички потребители с по-малко от 300 отзива.

```
best_user_ratings = ratings['User-ID'].value_counts() > 300
ratings = ratings[ratings['User-ID'].isin(best_user_ratings[best_user_ratings].index)]
```

❖ Решение на задачата

1) Съединяваме таблиците Books и Ratings по ISBN на книгите.

```
ratings_with_books = ratings.merge(books, on='ISBN')
```

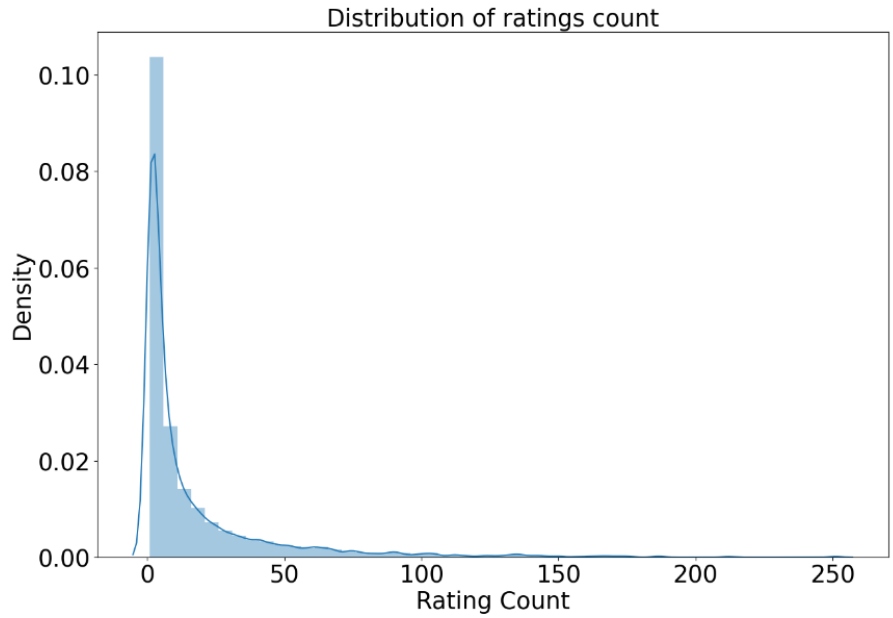
2) Създаваме таблица, която съпоставя име на книга с общ брой отзиви за нея.

```
ratings_count = ratings_with_books.groupby('Title')['Book-Rating'].count().reset_index()
ratings_count.rename(columns={'Book-Rating': 'Ratings-Count'}, inplace=True)
```

3) Съединяваме новосъздадената таблица с предишната по заглавие на книгите.

```
merged_with_ratings_and_books = ratings_with_books.merge(ratings_count, on='Title')
```

Дистрибуция на броя на отзивите:



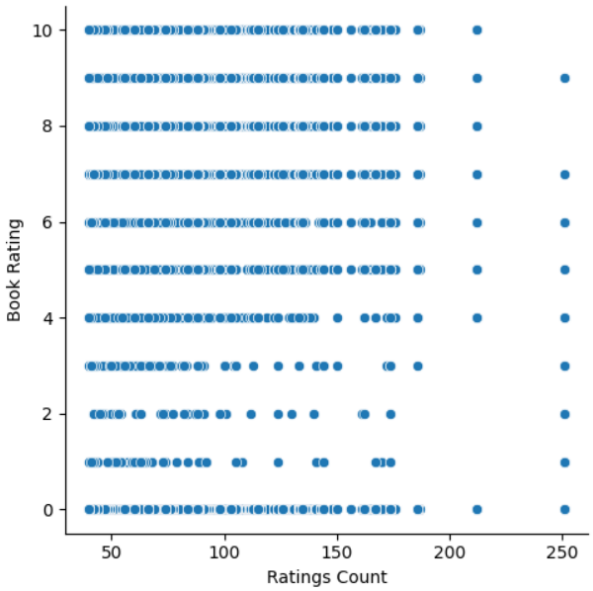
Забелязваме, че мнозинството книги имат по 0 отзива и за подобряване точността на системата премахваме всички книги с по-малко от 40 отзива.

```
final_rating = merged_with_ratings_and_books[merged_with_ratings_and_books['Ratings-Count'] >= 40]
```

4) Премахваме повтарящите се отзиви от един и същ потребител на една и съща книга.

```
final_rating.drop_duplicates(['User-ID', 'Title'], inplace=True)
```

Връзка между брой отзиви и отзиви на книгите:



Забелязваме, че няма връзка между атрибутите Book Rating и Ratings Count.

5) Създаваме обобщена таблица с колони идентификаторите на потребителите, редове- заглавия на книгите. Стойностите на таблицата ще са отзивите, които съответният потребителя е дал на дадена книга.

```
book_pivot = final_rating.pivot_table(columns='User-ID', index='Title', values='Book-Rating')
print(book_pivot.head())
```

```
User-ID                254     2276    3363    ...  275970  277427  278418
Title                                          ...
1984                     9.0     NaN     NaN   ...     0.0     NaN     NaN
1st to Die: A Novel      NaN     NaN     NaN   ...     NaN     NaN     NaN
2nd Chance               NaN    10.0     NaN   ...     NaN     NaN     NaN
4 Blondes                NaN     NaN     NaN   ...     NaN     NaN     NaN
84 Charing Cross Road    NaN     NaN     NaN   ...    10.0     NaN     NaN
```

Заменяме стойностите NaN (липса на отзив от даден потребител за книга) с 0, защото алгоритъмът, който ще използваме по-късно, пресмята разстояние между съседни стойности.

6) Трансформираме обобщената таблица в SciPy разредена матрица за по-ефикасни пресмятания.

```
book_sparse = csr_matrix(book_pivot)
```

7) Прилагаме алгоритъма k- най-близки съседи върху получената разредена матрица. Избираме алгоритъма, чрез който да намерим най-близките съседи, да е brute (алгоритъм с брутална сила) и напасваме модела.

```
model = NearestNeighbors(algorithm='brute')
model.fit(book_sparse)
```

- Тестов модел на системата

```
def recommend_book(book_name):
    book_id = np.where(book_pivot.index == book_name)[0][0]
    distance, suggestion = model.kneighbors(book_pivot.iloc[book_id, :].values.reshape(1, -1), n_neighbors=11)
    for i in range(len(suggestion)):
        books = book_pivot.index[suggestion[i]]
        for j in books:
            if j == book_name:
                print(f"You searched '{book_name}'\n")
                print("The suggestion books are: \n")
            else:
                print(j)
```

Създаваме тестов модел на системата, който приема име на книга и връща предложения за книги на потребителя.

1) Запазваме номера на реда, на който се намира подадената книгата като book_id, за да го използваме за търсене в алгоритъма.
2) Прилагаме алгоритъма к- най-близки съседи за намерения индекс и избираме броя на книгите, които искаме системата да предлага на потребителите да е 10.
3) Извеждаме предложените книги.

4. Примери, илюстриращи работата на програмната система

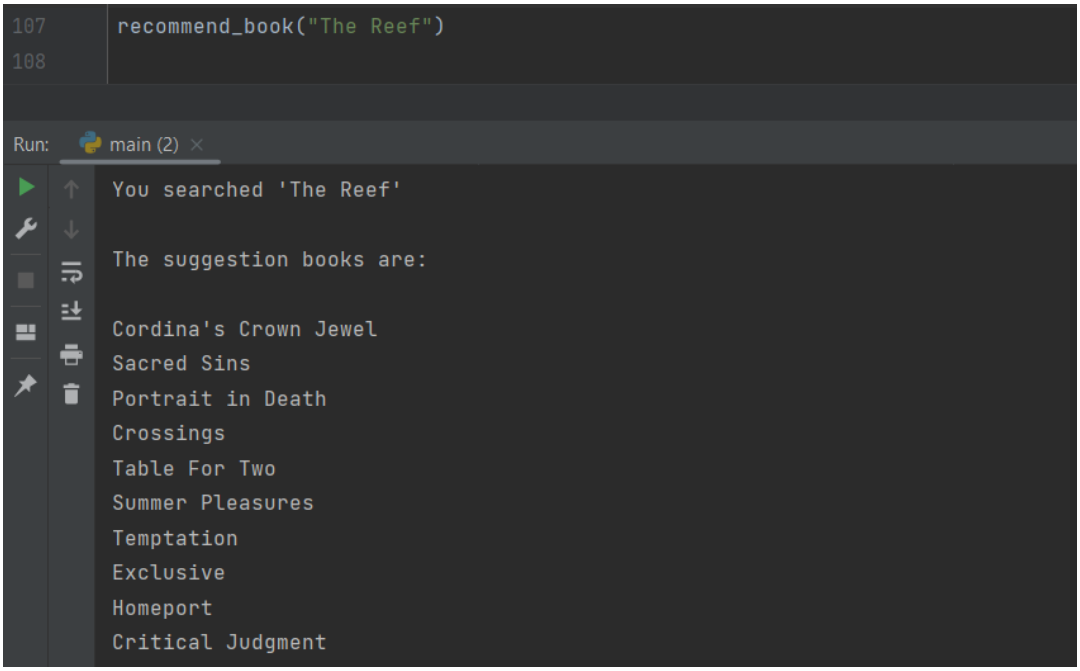
```
107    recommend_book("Harry Potter and the Chamber of Secrets (Book 2)")
```

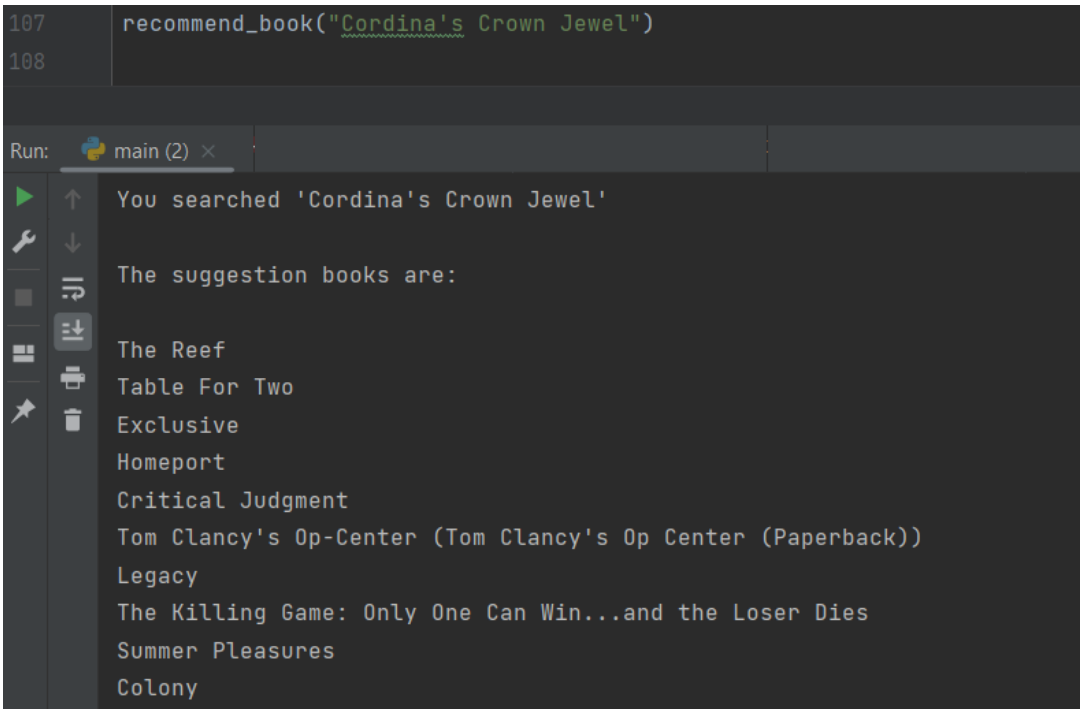
Run: main (2)

```
You searched 'Harry Potter and the Chamber of Secrets (Book 2)'

The suggestion books are:

Harry Potter and the Prisoner of Azkaban (Book 3)
Harry Potter and the Goblet of Fire (Book 4)
Harry Potter and the Sorcerer's Stone (Book 1)
Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
Tom Clancy's Op-Center (Tom Clancy's Op Center (Paperback))
Exclusive
The Cat Who Tailed a Thief
Weep No More My Lady
The Reef
Bittersweet
```





5. Литература

1) Susan Li, How Did We Build Book Recommender Systems in An Hour Part2 — k Nearest Neighbors and Matrix Factorization, Towards Data Science, 2017, 28.12.2022г.
2) Cai-Nicolas Ziegler, DBIS Freiburg, Book-Crossing Dataset, Institut für Informatik, Universität Freiburg, 2004, 28.12.2022г.
3) Amal Joby, What is K-Nearest Neighbor? An ML Algorithm to Classify Data, 2021, 28.12.2022г.