



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 (recommender system)

Студенти:

Иван Иванов Петров, ЗМ10700035
Петя Ангелова Личева, ЗМ10700022

София, януари 2025 г.



С ъ д ъ р ж а н и е

.....	2
2. Използвани данни.....	5
3. Организация на файловете.....	7
4. Използвани алгоритми.....	8
5. Описание на програмната реализация.....	12
6.	24
7. Конфигурация на проекта.....	28
8. Литература.....	29



1. Формулировка на задачата

Система за препоръки на книги, като препоръките се базират на рейтинг от читателите/потребителите. На данните би могло да се приложи ре-филтрация по възраст на читателите. Основен проблем, който би следвало да се реши преди ре-филтрацията е, че около 40% от читателите не са посочили възрастта си, което води до нужда от поправки/зачистване в данните преди ре-филтрацията, което на по-късен етап установихме, че води до загуба на информация и неточности в модела. Естествено, в следната документация предлагаме пълния процес, през който преминахме в изграждането на система за препоръки на книги.

Забележка: Преди да достигнем крайната реализация на задачата си преминахме през няколко различни потенциални подхода за решаване на подобна задача, но основен подход, на който се спряхме, за реализация на задачата ни е k-NN алгоритъма във варианта му на a-NN (k-NN алгоритъм с приближение). За да изведем съображенията, които ни насочиха да тръгнем в тази посока би било редно да опишем подходите и експериментите, които проведохме, заедно с техните резултати и изводи, както и да покажем с какво a-NN подхода превъзхожда всички останали, изпробвани алгоритми в условностите на задачата ни. Повече информация за процеса на имплементация, както и за преценката за нужния ни алгоритъм може да намерите в **графа Използвани алгоритми** на документацията.

Забележка: Кодът е разработен на Python със съответно необходимите му библиотеки като:

Numpy

Numpy е библиотека за научни изчисления с Python, която предоставя поддръжка за работа с многомерни масиви и набор от математически функции за бързи операции с тези масиви. В кода **Numpy** се използва за:

- Преобразуване на колони в масиви (`np.array()`), които се използват като входни данни за изчисления и модели.



- Комбиниране на колони в масиви (`np.column_stack()`) за създаване на матрици, използвани като характеристики (`features`) в модела.

pandas

pandas е библиотека за манипулация и анализ на данни. Тя предлага структури от данни като `DataFrame`, които улесняват работата с таблични данни. В кода **pandas** се използва за:

- Зареждане на CSV файлове с информация за книги, потребители и рейтинги (`pd.read_csv()`).
- Обединяване на данни от различни източници (`pd.merge()`).
- Филтриране и попълване на липсващи стойности в данните (`dropna()` и `fillna()`).

pprint

Тази библиотека предоставя функции за красиво форматиран печат на сложни структури от данни. В кода се използва `pprint` (преименувана като `pp`) за по-четливо отпечатване на резултатите от определени изчисления, например квантилите на броя на оценките.

sklearn.neighbors

Модулът **sklearn.neighbors** предоставя алгоритми за машинно обучение на базата на съседи на текущо тествания пример. В кода е използван класът **KNeighborsRegressor**, който:

- Използва метода на най-близките съседи за регресия.
- Предсказва стойност на база сходството (напр. близост по рейтинг на книга и възраст на читател).

scikit-learn (от sklearn.neighbors)

Scikit-learn е мощна библиотека за машинно обучение в Python. В кода е използван класът `NearestNeighbors` от модула `neighbors`. Неговата роля е да имплементира алгоритъм за намиране на най-близки съседи (`k-Nearest Neighbors`), който в случая се използва за препоръчване на книги въз основа на сходството между потребителските оценки.

sklearn.metrics



Модулът **sklearn.metrics** предлага функции за оценка на качеството на машинно обучение. В кода се използват:

- `mean_absolute_error` и `mean_squared_error` за изчисляване на грешките в модела.
- Допълнителни функции за оценка като `precision_at_k` и `recall_at_k` се имплементират ръчно, за да измерват качеството на препоръките.

sklearn.model_selection

Модулът **sklearn.model_selection** осигурява инструменти за разделяне на данни и валидиране на модели. В кода **train_test_split**:

- Разделя данните на тренировъчен и тестов набор, което гарантира, че моделът се тренира върху тренировъчното множество данни и се оценява коректно.

scipy.stats

Модулът **scipy.stats** предоставя статистически функции и разпределения. В кода се използва **pearsonr**:

- За изчисляване на коефициента на корелация (по метода на Пиърсън) между рейтингите на книги и възрастта на читателите.
- Това помага да се оцени дали съществува линейна връзка между тези две променливи.

scipy.sparse

Тази библиотека предоставя инструменти за работа с разреждени матрици (matrices with a lot of zeros). Класът `csr_matrix` се използва за създаване на разреждена матрица, която съхранява само ненулевите стойности, което значително оптимизира използването на памет и скоростта на изчисленията при работа с големи набори от данни.

coo_matrix (от scipy.sparse)

Използва се за създаване на разреждена матрица от рейтингите на потребителите за книги. Разрежданата матрица позволява ефективно съхранение и обработка на големи данни.

hnswlib

Използва се за инициализация на индекс за бързо търсене на книги с подобни рейтингови профили. Търсене на препоръки чрез намиране на най-близки съседи по рейтингова матрица.

faiss



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

Използва се за създаване на индекс за популярни книги, като се използват нормализирани рейтингови вектори:

Търсене на препоръки за популярни книги чрез сравнение на рейтинговите вектори.

normalize (от `sklearn.preprocessing`)

Използва се за нормализиране на рейтингите за популярни книги:

- Нормализиране на TF-IDF ембедингите за непопулярни книги:

TfidfVectorizer (от `sklearn.feature_extraction.text`)

Използва се за генериране на ембедингите за непопулярни книги въз основа на техните заглавия.

Annoy

Библиотеката Annoy (Approximate Nearest Neighbors) се използва за бързо намиране на най-близките съседи (nearest neighbors) на книги въз основа на рейтингова матрица. В кода това е реализирано чрез следните стъпки:

1. Инициализация на индекс: AnnoyIndex се инициализира с броя на характеристиките (`num_features`) и метриката за изчисляване на близост. В случая е избрана `angular` метриката (ъглова близост), която е подходяща за намиране на подобия между вектори.
2. Добавяне на данни към индекса: Всеки ред от разредената матрица `sparse_matrix`, представляващ вектор на книга, се добавя към Annoy индекса.
3. Построяване на дървета: Annoy използва дървета (`trees`) за ефективно търсене. В този случай индексът се построява с 10 дървета.
4. Търсене на най-близки съседи: Функцията `get_recommends` извиква метода `get_nns_by_item`, за да намери най-близките съседи на дадена книга (по ISBN). Методът връща както индексите на съседите, така и дистанциите до тях.

warnings

Модулът **warnings** е предназначен за управление на warning-ите в Python. В кода **warnings.filterwarnings('ignore')**:

- Заглушава предупрежденията, които биха могли да разсейват при анализа на резултатите или изпълнението на кода.

Тези библиотеки съвместно осигуряват основата за предварителна обработка, статистически анализ, машинно обучение и оценка на резултатите.



2. Използвани данни

Описаните по-долу **CSV таблици** са взети от **dataset-a** в <https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset>, а този ресурс е предоставен и в **графа Литература** на документацията под наименованието **Сетове с данни**.

A. Users

User-ID	Location	Age
1	nyc, new york, usa	
2	stockton, california, usa	18
3	moscow, yukon territory, russia	
4	porto, v.n.gaia, portugal	17
5	farnborough, hants, united kingdom	
6	santa monica, california, usa	61
7	washington, dc, usa	
8	timmins, ontario, canada	
9	germantown, tennessee, usa	
10	albacete, wisconsin, spain	26

Таблица, която съдържа информация за потребителите, наричани още читатели в рамките на нашата система. Информацията, която се съхранява за читателите е техните уникални идентификационни номера (**User-ID**), локациите им (**Location**) и възрастта им. Потребителските идентификатори са анонимизирани и съпоставими с цели числа. Данните за местоположението на читателите и тяхната възраст се предоставят, ако има такива. В противен случай тези полета приемат NULL стойности. Това предполага, че ако те бъдат включвани в методите за категоризация и препоръки на книги на други потребители, трябва да се подсигурирм за коректността на данните и евентуално да ги почистим от NULL стойности, тъй като те не внасят никаква информация и пречат на процеса по препоръка на книги.

B. Books

ISBN	Book-Title	Book-Author	Year-Of-Put	Publisher	Image-URL	Image-URL	Image-URL-L			
0195153448	Classical My	Mark P. O. F	2002	Oxford Univ	http://imag	http://imag	http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg			
0002005018	Clara Callan	Richard Bru	2001	HarperFlam	http://imag	http://imag	http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg			
0060973125	Decision in I	Carlo D'Este	1991	HarperPere	http://imag	http://imag	http://images.amazon.com/images/P/0060973129.01.LZZZZZZZ.jpg			
0374157065	Flu: The Sto	Gina Bari Kc	1999	Farrar Strau	http://imag	http://imag	http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg			
0393045218	The Mummi	E. J. W. Barl	1999	W. W. Nort	http://imag	http://imag	http://images.amazon.com/images/P/0393045218.01.LZZZZZZZ.jpg			
0399135782	The Kitchen	Amy Tan	1991	Putnam Pub	http://imag	http://imag	http://images.amazon.com/images/P/0399135782.01.LZZZZZZZ.jpg			
0425176428	What If?: T	Robert Cow	2000	Berkley Pub	http://imag	http://imag	http://images.amazon.com/images/P/0425176428.01.LZZZZZZZ.jpg			
0671870432	PLEADING C	Scott Turov	1993	Audioworks	http://imag	http://imag	http://images.amazon.com/images/P/0671870432.01.LZZZZZZZ.jpg			
0679425608	Under the B	David Cordi	1996	Random Ho	http://imag	http://imag	http://images.amazon.com/images/P/0679425608.01.LZZZZZZZ.jpg			



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

Таблица, която предоставя информация за книгите, от които имаме в наличност и ще можем да препоръчваме на нашите читатели. Книгите са уникално идентифицируеми от техните **ISBN** номера като трябва предварително да се погрижим, че всички ISBN номера са уникални и валидни. Таблицата също така предоставя информация за съдържанието на книгата като заглавие на книгата (**Book-Title**), автор на книгата (**Book-Author**), година на публикация на книгата (**Year-Of-Publication**) и издателство на книгата (**Publisher**). **Тази информация е предоставена от Amazon Web Services.** Трябва да се отбележи също така, че при наличие на съавторство над някоя книга, само имената на първият автор са предоставени. В таблицата също така се съхранява информация за линка към корицата на съответната книга (линк към уеб сайта на Amazon). Тези линкове може да се показват под различни форми (Image-URL-S или къси, Image-URL-M или средно дълги, Image-URL-L или дълги).

C. Ratings

User-ID	ISBN	Book-Rating
276725	034545104X	0
276726	0155061224	5
276727	0446520802	0
276729	052165615X	3
276729	0521795028	6
276733	2080674722	0
276736	3257224281	8
276737	0600570967	6
276744	038550120X	7
276745	342310538	10

Таблица, която описва оценките на книгите. В нея се съдържа информация кой потребител (**User-ID**) коя книга (**ISBN**) а е оценил и каква е била самата оценка (число в интервала от 1 до 10, вписана в полето **Book-Rating**). Ако някоя книга все още не е била оценена от даден потребител, оценката, която автоматично се вписва в полето за оценка (**Book-Rating**) е 0.

След като Ви запознахме достатъчно подробно със съдържанието на CSV таблиците и Ви разяснихме компонентите, от които са съставени данните, които системата ни ще има за цел да обработи и да извлече достатъчно изчерпателни препоръки на книги към своите потребители (, наричани още читатели) от тях, идва ред да Ви запознаем с **Организацията на файловете** в следващата графа от този документ.

3. Организация на файловете

По-надолу описваме структурата на проекта си - неговите папки, както и съдържанието им, за прегледност и по-голямо разбиране при работа с кода на нашата система.














- **docs** - папка, която съдържа документацията на проекта (, тоест текущо разглеждания от Вас файл), както и папка **resources** и прилежащите й файлове, необходими за изграждането на тази документация.
 - **resources** - папка, съдържаща всички необходими ресурсни файлове за създаването на тази документация (например скрийншотите, включени в нея).
- **public**- папка, съдържаща глобално достъпни (и от двамата разработчици) файлове, необходими за осмислянето на задачата и построението на решението, такова, каквото е изложено в този документ.
- **src** - папка, съдържаща данните за обработка, изходния код на системата и Jupyter notebook-a, описващ достатъчно подробно кода и прилежащите му примери. Цялото това съдържание условно се разделя в следните три папки:
 - **code** - папка, която съдържа изходния код на всички проведени експерименти, описани в този документ, както и изходния код на крайния алгоритъм, използван за решаването на проблема.
 - **data** - папка, съдържаща всички CSV таблици, използвани в процеса на разработка на системата.
 - **jupyter** - папка, съдържаща ipynb файла на въпросния Jupyter notebook, използван за демонстрация на работата на системата и документация на самата демонстрация.

След като Ви предоставихме нагледна информация за структурата и съдържанието на папките в нашата система, идва ред да Ви запознаем с **Използваните алгоритми** в следващата графа от този документ.

4. Използвани алгоритми

За решаването на задачата сме използвали няколко алгоритъма - разновидности на k-NN алгоритъма с цел проучване на потенциални възможности за модела и избиране на най-добрия възможен в конкретиката на задачата ни и dataset-овете, описани по-горе. В следващите редове ще опишем всеки един от изпробваните модели, заедно с проблемите, които сме срещнали при тяхната имплементация, възможните им решения и скрийншоти, показващи нагледно тяхната работа.

k-NN

		Items					
							
Users		10	-1	8	10	9	4
		8	9	10	-1	-1	8
		10	5	4	9	-1	-1
		9	10	-1	-1	-1	3
		6	-1	-1	-1	8	10

User-item Interaction matrix

Описание на алгоритъма

k-NN алгоритъма е класификационен алгоритъм, който се базира на изчисления на разстоянията между текущо изследвания тестови пример и неговите k на брой най-близки съседи, така че да се открият най-добрите примери от някаква извадка данни и според тази класификация, те да бъдат препоръчани на потребителя на системата (в случая говорим за книги, които трябва да се препоръчат на определена група читатели със сходства помежду си). За тази цел ни трябва определена метрика, която често пъти е Евклидово или Манхатанско разстояние между дадените тестови примери и техните най-близки съседи (за простота).

- Евклидово разстояние: Ако **точката A(x₁, y₁)** и **точката B(x₂, y₂)**, то

$$d_{AB} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- Манхатанско разстояние: Ако **точката A(x₁, y₁)** и **точката B(x₂, y₂)**, то

$$d_{AB} = |x_1 - x_2| + |y_1 - y_2|$$

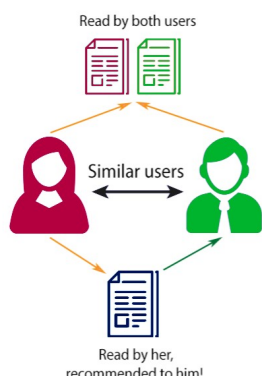
Тези метрики обаче не са достатъчно добри за нашата задача, защото те допускат дори и вземането на всички примери (в случая книги), които са на горе-долу еднакво разстояние от текущо тествания, а **cosine метриката** - взема предвид и ъгъла, под който се намира съседа на текущо тествания пример спрямо текущо тествания пример и въз основа на това колко голям е ъгълът решава кой съсед (в случая книга) да добави в препоръчаните.



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

k-NN алгоритъм с регресорна препоръка на книги и грануларно изчисление на дистанциите (между текущо изследвания пример и неговите k на брой най-близки съседи)

COLLABORATIVE FILTERING



Описание на алгоритъма

- Идеята зад този алгоритъм е да се използва колаборативно филтриране за препоръчване на книги на потребителите, като се вземат предвид техните оценки на книги и възраст. Това се постига чрез изчисляване на сходството между потребителите, като се използва Евклидово разстояние, което комбинира две характеристики: оценките на книгите и възрастта на потребителите. Целта е да се намерят потребители със сходни предпочитания (като се сравняват оценките им за книги) и да се препоръчат книги, които тези сходни потребители са харесали. Освен това, възрастта на потребителя е допълнителен фактор, който може да повлияе на препоръките, защото различни възрастови групи могат да имат различни литературни предпочитания. Алгоритъмът използва метода на най-близките съседи (k-NN), за да определи кои потребители имат най-сходни вкусове на базата на оценките на книги и възрастта им. Също така, чрез Евклидовото разстояние, се придава по-голямо значение на оценките на книгите, отколкото на възрастта, което отразява важността на самите предпочитания в сравнение с демографската информация. Идеята е, че чрез тази комбинирана оценка на сходствата, моделът може да предостави по-точни и персонализирани препоръки за книги на потребителите, които може да не са срещали, но са им сходни с други потребители, които имат подобни литературни вкусове. k-NN алгоритъм с линейна регресия за оценката влиянието на годините на читателите върху рейтинга на книгите и грануларни изчисления на разстоянията между текущо изследваната книга и нейните k на брой най-близки съседи чрез определена метрика (Евклидова, Манхатански разстояния, cosine и други) - стъпките са сходни като тези на горния алгоритъм, но с включване на корелация между рейтингите и възрастите на читателите, регресия при положителна оценка за линейна връзка между двата компонента и грануларни изчисления, базирани на следната формула:



$$d_w(X, Y) = \sum_{i=1}^n w_i * (x_i - y_i)^2$$

а-NN алгоритъм (оптимизиран k-NN алгоритъм)

Описание на алгоритъма

Приблизителният най-близък съсед (ANN) е важен проблем в машинното обучение и има множество приложения в различни области. Значението на търсенето на ANN се крие в способността му да намира ефективно приблизителни решения на проблема с най-близките съседи, което може да доведе до значително ускоряване и спестяване на памет в сравнение с търсенето на точни най-близки съседи. Очевидно има компромис между точността на приблизителните резултати и времето, което се изразходва за изчисления. В някои задачи наборът от данни е толкова голям, че напълно точното търсене е дори невъзможно в разумно време, а в други - разликата между точния най-близък съсед и приблизителния съсед, който е близо, може да не бъде забележима. ANN се използва в различни приложения:

- **Компютърно зрение**
- **Извличане на информация**
- **Системи за препоръки**
- **Изчислителна биология**

Основни разновидности на ANN алгоритъма

HNSW

Основна идея

Алгоритъмът HNSW (Hierarchical Navigable Small World) е разширена версия на алгоритъма NSW (Navigable Small World), който се базира на концепцията за "малък свят". Тази концепция предполага, че в графовете всяка връзка е краткосрочна към съседите (около логаритмично количество стъпки), но също така има и няколко дългосрочни връзки, които позволяват ефективно глобално търсене. Алгоритъмът HNSW изгражда многопластова йерархична структура от графове, която позволява ефективно индексирание и търсене на най-близки съседи в високоизмерни пространства.

FAISS

Основна идея

FAISS ускорява търсенето на най-близки съседи в големи набори от данни, като използва характеристиките на високомерните пространства. Традиционните методи като изчерпателното търсене (brute-force) или K-D дърветата са бавни и паметно интензивни при обработка на такива данни. FAISS предлага алгоритми, създадени специално за ефективно търсене в подобни пространства.



Основна идея

Annoy изгражда йерархична дървовидна структура, наречена *Annoy дърво*, за да организира данните. Това позволява ефективно търсене на приблизително най-близки съседи. Алгоритъмът жертва малка част от точността на резултатите, за да постигне значително по-бързо време за търсене и по-ниска консумация на памет.

5. Описание на програмната реализация

k-NN

Алгоритъмът в стъпки

1. Зареждане и обработка на данни

- Зареждане на данните от файлове `Ratings.csv`, `Books.csv` и `Users.csv` в **Pandas DataFrame**.
- Обработка на липсващи стойности:
 - ◆ Попълване на липсващи стойности с фиксирани стойности като "NaN" или -1.
 - ◆ Изтриване на редове с липсващи стойности в оценките (`df_ratings.dropna()`).

2. Комбиниране на данни

- Обединяване на данни за оценки (`df_ratings`) и книги (`df_books`) чрез общата колона `ISBN`.
- Премахване на ненужни колони, като например `Book-Author`.

3. Изчисляване на популярност на книгите

- Групиране на данни, за да се преброи броят на рейтингите за всяка книга.
- Обединяване на тези резултати с оригиналния набор от данни.

4. Филтриране на популярни книги

- Изчисляване на прагово ниво за популярност на книгите въз основа на 90-ия перцентил от броя на рейтингите.

5. Създаване на матрица за препоръки

- Преобразуване на данните в **pivot таблица**, където редовете са ISBN на книги, колоните са потребителски идентификатори, а стойностите са рейтинги.
- Преобразуване на тази матрица в рядко представяне (**CSR matrix**), за да се спести памет.

6. Създаване и обучение на модел



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

- Инициализиране на модел **NearestNeighbors** с метрична функция "косинусова подобие".
 - Обучение на модела с данните от CSR матрицата.
7. Препоръчване на книги
- Дефиниране на функцията `get_recommends`, която приема заглавие на книга като вход, изчислява най-близките съседи на тази книга, и връща списък от препоръчани книги.
 - Използване на функцията за препоръки.

Алгоритъмът в код

Кодът на този експеримент с коментари към него може да откриете във файла **knn.py** в директорията **src/code**.

Резултати

```
0.90    136.0
0.91    150.0
0.92    167.0
0.93    184.0
0.94    209.0
0.95    236.0
0.96    277.0
0.97    350.0
0.98    420.0
0.99    568.0
Name: RatingCount, dtype: float64
'Northwest Wines and Wineries is not in the top books'
```

На горния скрийншот са показани броят рейтинги на топ 10 най-четени книги и съобщение, че няма препоръчани книги.

```
0.90    136.0
0.91    150.0
0.92    167.0
0.93    184.0
0.94    209.0
0.95    236.0
0.96    277.0
0.97    350.0
0.98    420.0
0.99    568.0
Name: RatingCount, dtype: float64
['1558745157',
 [ ["Left Behind: A Novel of the Earth's Last Days (Left Behind No. 1)",
    0.9381235623430844],
   ['Night Sins', 0.9352435834429496],
   ['On the Street Where You Live', 0.9258003569774721],
   ['A Child Called \\It\\": One Child\\'s Courage to Survive"',
    0.7356135842239919],
   ["The Lost Boy: A Foster Child's Search for the Love of a Family",
    3.3306690738754696e-16]]]
```



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

На горния скрийншот са показани броят рейтинги на топ 10 най-четени книги и препоръки за други книги, базирани на дадена от потребителя книга.

Изводи

При наблюдение на големи масиви от данни, каквито са представените във въпросните CSV таблици, алгоритъмът значително се забавя в изчисленията си и препоръките не са достатъчно персонализирани, за да можем да твърдим, че този модел е най-доброто решение. Това ни наведе на няколко оптимизационни идеи. На първо място решихме да използваме автоматичен подбор на структурата (на база колко мерни са наборите данни), която да използва k -NN модела, вместо възможностите:

- brute - brute-force алгоритъм, който изчислява разстоянията между всяка двойка обекти, за оптималност използва векторизирани изчисления и е ефективен за всякакъв вид метрика. Подходящ е при много малки или много големи обеми данни.
- ball-tree - балансирано дърво, което разделя пространството на сфери и е подходящо за работа с големи или високоразмерни набори от данни. Този алгоритъм е подходящ за Евклидова или друга подобна метрика.
- kd-tree - оптимизиран алгоритъм за работа с големи или високоразмерни набори от данни. Може да бъде по-бърз от другите алгоритми при тези размерности. Добре се справя с Евклидови и Манхатънски разстояния, но не работи добре с неквадратни метрични функции, като косинусово подобие.

На второ място, решихме да търсим начин за персонализирана препоръка на книги на база рейтингите на книгите и възрастовите групи, които се оформят като тенденции сред читателите, използващи нашата система. Естествено тук се пороци въпроса: Ако досега можехме да представим алгоритъма в двумерна матрица, в която индексите на редовете са идентификационните номера на потребителите/читателите ни, а индексите на колоните са ISBN номерата на книгите, а стойностите в матрицата са оценките, които потребителите са давали за дадена книга. То сега как да изобразим положението, при което ще отправяме препоръки по 2 показателя? Отговорът, който намерихме беше свързан с представянето на линейната връзка между двата показателя, по които ще препоръчваме книги и така стигнахме до идеята за следващия ни експериментален алгоритъм.

k -NN алгоритъм с регресорна препоръка на книги и грануларно изчисление на дистанциите (между текущо изследвания пример и неговите k на брой най-близки съседи)

Алгоритъмът в стъпки

1. Включване на необходимите библиотеки и пакети.



2. Зареждане на данните.
3. Обединяване на данните за рейтингите на книгите (всички данни от таблицата с рейтингите) с годините на читателите.
4. Почистване на данните от невалидни такива (NaN стойности).
5. Подравняване на размерите на данните.
6. Изчисляване на корелацията по Пиърсън за определяне колко смислена би била препоръка на книга на базата на възрастта на читателя и рейтинга на книгите.
7. Подготовка на данните и разделянето им на тренировъчни и тестови множества данни.
8. Създаване и трениране на модела с тренировъчните множества данни.
9. Грануларно изчисление на разстоянията на текущо изследвания пример и неговите k на брой най-близки съседи.
10. Препоръка на книга.
11. Оценка на качеството на модела.

Алгоритъмът в код

Кодът на този експеримент с коментари към него може да откриете във файла **knn_with_regression_and_granular_computation.py** в директорията **src/code**.

Резултати

- **Наблюдавани проблеми:** Основен проблем, който срещнахме при този подход е, че трябваше да се установи дали между двата параметъра - години на читателя и рейтинг на книгата съществува линейна връзка, която да ни позволява да използваме регресия за оценка на книгите и потенциални препоръки към читателите. За целта обаче първо трябваше да зачистим данните от незначещи стойности (като NaN стойности). При анализ на данните от трите таблици открихме, че около 40% от читателите не са посочили своята възраст, което доста намали шансовете да постигнем извода за връзка между въпросните два параметъра, на които искахме да базираме нашите препоръки. Все пак проведохме този експеримент като проверихме коефициента за корелация по метода на Пиърсън.
- **Корелация по Пиърсън:** След като проверихме връзката на рейтинга на книгите и възрастта на читателите, установихме, че p -value-то е равно на 0.0, което от своя страна показва, че между двата компонента не е налична линейна връзка и следователно регресията тук не би довела до добри препоръки на книги.

```
Percentage of records without NaN values: 73.08%
Pearson Correlation: -0.0374251240504987
P-value: 0.0
```




➤ **Скриншот на препоръките на книги на базата на рейтингите на книгите и възрастта на читателите:**

Снимката по-долу предоставя препоръките на книги за двама читателя с тотално различни характеристики - единият читател е на 33 години и предпочита книги с рейтинг 3 / 10, а другият читател е на 25 години и предпочита книги с рейтинг 5 / 10. Информацията, която се предоставя на читателя при препоръка е името и автора на книгата.

```
Recommendations for a 33 years old user who wants to read a book with rating 3:
Book Title: The Deadhouse, Author: Linda Fairstein
Book Title: Death a L'Orange, Author: Nancy Fairbanks
Book Title: Energy Up!: Shed Pounds, Get Fit, Gain Stamina, and Turn on Your Power With This Unique Program, Author: High Voltage
Book Title: The Billionaire Bridegroom (Passion), Author: Emma Darcy
Book Title: The Memory of Running, Author: Ron McLarty

Recommendations for a 25 years old user who wants to read a book with rating 5:
Book Title: Heretics of Dune, Author: Frank Herbert
Book Title: Flirting with Pete : A Novel, Author: Barbara Delinsky
Book Title: Call It Sleep, Author: Henry Roth
Book Title: Marabou Stork Nightmares, Author: Irvine Welsh
Book Title: Billy, Author: Whitley Strieber
```

- **Оценка на качеството на модела:** След като открихме, че този подход не би бил удачен в нашата задача, ние решихме да продължим с него с напълно експериментални цели, за да затвърдим или евентуално опровергаем това ни заключение. Завършихме подхода по описаните по-горе стъпки и в последната стъпка при оценката на качеството на модела достигнахме до извода, че едва 40% от препоръките са релевантни на предпочитанията на читателите и че около 67% от релевантните препоръки са включени в топ 5 препоръчвани на отделния читател книги, тоест между 1 и 2 от 5 препоръчани на читателя книги наистина съответства на неговите предпочитания.

```
Precision@5: 0.4
Recall@5: 0.6666666666666666
```

Изводи

Този резултат само затвърждава по-ранно изведение (чрез корелацията по метода на Пиърсън) извод, че един такъв модел не би бил достатъчно ефективен при препоръките на книги, следователно следва да се мисли в друга посока, например оптимизирания k-NN алгоритъм (, познат още като Approximate K Nearest Neighbour - a-NN алгоритъм, който ще разгледаме малко по-надолу).



Основни разновидности на ANN алгоритъма

HNSW

Алгоритъмът HNSW в стъпки

1. Зареждане на данните

- Зареждаме данни за рейтингите на книги, книгите и потребителите от CSV файлове.
- Попълваме липсващи стойности с подходящи стойности, напр. NaN за книги и -1 за потребители.

2. Обработка на данните

- Обединяваме данните за рейтинги с информацията за книгите по колоната ISBN.
- Премахваме ненужни колони като "Book-Author".
- Изчисляваме броя на рейтингите за всяка книга, използвайки групиране по ISBN.

3. Филтриране

- **Популярни книги:** Оставяме само книгите, които са били оценени поне определен брой пъти (например 136).
- **Активни потребители:** Задържаме само потребителите, които са оценили поне 5 книги.

4. Създаване на мапинг за потребители и книги

- Създаваме речници за съпоставяне на уникални User-ID и ISBN със съответните им индекси.
- Добавяме нови колони в данните с индексите за потребителите и книгите.

5. Създаване на разрежена матрица

- Създаваме разрежена матрица (`sparse_matrix`), където редовете съответстват на книги, колоните – на потребители, а стойностите – на оценките.

6. Инициализация на HNSW индекс



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

- Инициализираме структура за търсене HNSW (`hnswlib.Index`) с:
 - **Косинусово разстояние** за измерване на сходство.
 - Размерност, равна на броя на потребителите.

- Зареждаме данните в HNSW индекса.

7. Функция за препоръки

- **Вход:** ISBN на книга и брой препоръчани книги (`k_neighbors`).
- **Проверка:** Проверява дали дадената книга е в популярния набор.
- **Извличане на вектор:** Намираме реда в матрицата, съответстващ на ISBN, за да използваме неговия вектор.
- **HNSW заявка:** Изпращаме вектора към HNSW индекса за намиране на най-близките `k` съседни.
- **Резултат:** Съставяме списък с препоръчани книги на базата на намерените индекси.

8. Тестване на алгоритъма

- Примерна заявка се прави с конкретен ISBN, за да се видят препоръките.

Алгоритъмът HNSW в код

Кодът на този експеримент с коментари към него може да откриете във файла `ann_hnsw.py` в директорията `src/code`.

Faiss

Алгоритъмът Faiss в стъпки

1. Зареждане на данни

- Четем данни за рейтингите, книгите и потребителите от CSV файлове.
- Попълваме липсващите стойности със стойности като `NaN` за книги и `-1` за потребители.
- Изтриваме празните редове от данните за рейтингите.

2. Обединение на данни

- Обединяваме данните за рейтингите с информация за книгите, използвайки `ISBN` като обща ключова колона.
- Премахваме ненужни колони като "Book-Author".



3. Изчисление на популярността

- Групираме данните по ISBN, за да преброим броя на оценките (`RatingCount`) за всяка книга.
- Обединяваме информацията за броя на оценките с първоначалните данни.

4. Филтриране на книги

- **Популярни книги:** Избираме книги с броя на оценките по-голям или равен на праг (например 136).
- **Непопулярни книги:** Избираме книги с по-малък брой оценки, но по-голям от 0.

5. Създаване на ембединг за популярни книги

- Създаваме **пивотна таблица**, където редовете са книги, колоните – потребители, а стойностите – оценки.
- Попълваме липсващите стойности с 0.
- **Нормализираме** векторите (редовете) на таблицата, за да използваме косинусово сходство.

6. Индексиране на популярни книги с Faiss

- Създаваме Faiss индекс за търсене с **косинусово сходство**.
- Добавяме нормализираните вектори на популярните книги в индекса.

7. Обработка на непопулярни книги с TF-IDF

- Извличаме заглавията на непопулярните книги.
- Създаваме TF-IDF представяне на текстовите заглавия.
- Нормализираме ембедингите и създаваме Faiss индекс за тях.

8. Препоръки

- **Популярни книги:**
 - Намираме индекса на дадена книга в популярните ембединг данни.
 - Извличаме k най-близки съседни от Faiss индекса за популярни книги.
- **Непопулярни книги:**
 - Намираме индекса на дадена книга в TF-IDF ембедингите.
 - Извличаме k най-близки съседни от Faiss индекса за непопулярни книги.
- **Fallback (резервен случай):**



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

- Ако книгата не съществува в популярните или непопулярните, връщаме най-популярните книги като препоръка.

9. Функции за помощ

- `get_title_isbn`: Намира заглавието на книга по нейния ISBN.
- `get_most_popular`: Връща k най-популярни книги, когато не е възможно да се намерят други препоръки.

10. Тестване

- Извикваме `get_recommends` с ISBN на популярна и непопулярна книга, за да видим препоръките.

Този алгоритъм комбинира два подхода:

- **Ембединг и Faiss за популярни книги** (по оценки).

Ембедингът (на английски: *embedding*) е начин за представяне на обекти, като текст, изображения или други видове данни, в числова форма. Това е вид векторно представяне, при което сложни обекти се преобразуват в математически вектори в многомерно пространство. Ембедингите улесняват компютърните алгоритми да обработват и сравняват тези обекти.

- **Математически вектор**: Ембедингът е вектор от числа (напр. $[0.1, 0.5, -0.3, 0.8]$), който описва обекта в многомерно пространство.
 - **Съхранява информация**: Ембедингът улавя най-важната информация за обекта (като смисъл, контекст или сходство).
 - **Позволява сравнение**: Близостта между два ембединга в пространството показва колко си приличат техните обекти.
- **TF-IDF** (от английски: *Term Frequency-Inverse Document Frequency*) е статистическа техника, която се използва в текстовата обработка и извличането на информация. Тя измерва значимостта на дадена дума в определен документ спрямо цялата колекция от документи (наречена "корпус").

TF-IDF комбинира две основни метрики:

1. TF (Term Frequency):

- Измерва колко често дадена дума се среща в един документ.
- Формула:

$$T(w, d) = \frac{\text{Брой срещания на думата } w \text{ в документа } d}{\text{Общ брой думи в документа } d}$$



2. IDF (Inverse Document Frequency):

- Измерва колко рядко се среща дадена дума в корпуса от документи.
- Формула:

$$IDF(w) = \log\left(\frac{N}{1 + \text{Брой документи, съдържащи думата } w}\right)$$

- N е общият брой документи в корпуса.
- 1 + брой документи, съдържащи думата добавя "1", за да се избегне деление на 0.

3. TF-IDF:

- Това е произведението на TF и IDF:

$$TF-IDF(w, d) = TF(w, d) \cdot IDF(w)$$

- Думи, които са често срещани в документа, но рядко срещани в останалите документи, получават висока стойност.

Начин на работа: Подчертава думи, които са специфични за даден документ, като намалява значението на думи, които са общи в много документи (напр. "и", "или", "но").

Алгоритъмът Faiss в код

Кодът на този експеримент с коментари към него може да откриете във файла **ann_faiss.py** в директорията **src/code**.

ANNOY

Алгоритъмът ANNOY в стъпки

1. Индексиране (Indexing):

- Избира се случайна точка за разделяне по случайно избран размер на данни, която създава кореновия възел на дървото.
- Данните се разделят рекурсивно в по-малки региони чрез избиране на нови точки за разделяне на различни размери по всяко ниво на дървото.
- Стремежът е да се разпределят данните равномерно през дървото, с цел балансирана структура на дървото.

2. Баланс (Balancing):



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

· Алгоритъмът гарантира, че дървото остава балансирано, като поддържа максимален размер за всеки възел.

- Ако броят на точките в даден възел надвишава максималния размер, възелът се разделя на два нови дъщерни възела чрез избиране на нова точка за разделяне.
- Процесът продължава рекурсивно, докато не се създадат всички листни възли в дървото.

3. Запитвания (Querying):

· Алгоритъмът започва от кореновия възел и сравнява точката за запитване с точката за разделяне в текущия възел.

- Въз основа на това сравнение алгоритъмът избира дъщерния възел, който е по-близо до точката за запитване.
- Процесът продължава рекурсивно надолу през дървото, като се избира дъщерният възел, който е по-близо до точката за запитване, докато не се достигне листен възел.

4. Приблизени резултати (Approximate results):

· След като се достигне листен възел, алгоритъмът извлича данните в този възел като начален набор от приблизителни най-близки съседи.

- Алгоритъмът усъвършенства приближените резултати чрез брутално търсене в ограничен радиус около точката за запитване в листния възел.
- Радиусът се определя от максималното разстояние между точката за запитване и приблизителните най-близки съседи, като алгоритъмът актуализира резултатите с всякакви по-близки точки, открити по време на търсенето в радиуса.

Алгоритъмът ANNOY в код

Кодът на този експеримент с коментари към него може да откриете във файла `ann_annoy.py` в директорията `src/code`.

Сравнение между ANNOY, HNSW и FAISS

Критерий	ANNOY	HNSW	FAISS
Точност	Висока, но контролирана от броя на дърветата.	Много висока, благодарение на малкия свят и дългосрочните връзки.	Много висока за големи и високодименсионални данни.
Скорост на търсене	Бърза, но зависи от броя на	Много бърза благодарение на	Изключително бърза при големи



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

	дърветата.	оптимизираното търсене в графи.	данни с GPU оптимизация.
Ефективност при добавяне на нови данни	Изисква пълно изграждане на дърветата при нови данни.	Поддържа динамични актуализации (добавяне и премахване на точки).	Изисква частична или пълна прекалибрация на индексите за някои типове.
Използване на памет	Ниско, създаден за работа с данни на диск.	Средно до високо, особено за големи графи.	Високо, изисква целият индекс да бъде в паметта за максимална ефективност. Сложна настройка, особено за специфични архитектури (GPU, специализирани индекси).
Простота на внедряване	Много лесен за настройка и използване.	Изисква по-сложна конфигурация и параметризация.	
Мащабируемост	Подходящ за малки до средни набори от данни (до милиони елементи).	Добра за средни до големи набори (до десетки милиони точки).	Отлична за много големи набори (стотици милиони до милиарди точки).
Поддръжка на специални функции	Основен фокус върху баланс скорост/точност, без сложни функции.	Предоставя гъвкавост за настройка на точността и динамика на търсене.	Предлага разширени функции като мулти-GPU поддръжка, мулти-проба и индексирание на редки данни.
Приложимост за препоръки за книги	Отличен за начални и средни по мащаб проекти с балансиран подход.	Подходящ за по-големи проекти, които изискват много висока точност и динамика.	Идеален за изключително мащабни приложения или сложни системи с високодименсионални данни.

За повече информация, относно този алгоритъм и неговите разновидности, вижте следната [статия](#).



Резултати

Изводи

След всички тези проведени експерименти над множествата от данните, разгледани и описани в графа **Използвани алгоритми** и графа **Използвани данни** на документацията, се спираме на ANN (approximate nearest neighbor) алгоритъма като най-оптимално решение за идентифициране на най-добрата препоръка на база експериментите ни. По-надолу в документацията описваме крайното решение на проблема ни и основни моменти от кода в графа **Описание на програмната реализация** от документацията, а в графа **Примери, илюстриращи работата на програмната система** от документацията предоставяме скрийншоти на резултатите след изпълнението на конкретните примери заедно с разяснения по тях.

Забележка: Предвид средно големите данни, които се обработват в системата ни, неналичието на условие за изключителна точност на прогнозите (каквато би се изисквало в един медицински софтуер, свързан с предсказания на симптоми на заболявания например) и неналичието на толкова ефективна изчислителна мощ (, каквато би била необходима за реализацията на някоя друга от разновидностите на ANN алгоритъма), ние предпочетохме да използваме ANN алгоритъма в смисъла на неговата разновидност като най-уместна.

6. Примери, илюстриращи работата на програмната система

В тази глава от документацията ще Ви представим резултатите, които постигнахме при експериментите, които проведохме заедно с коментари и изводи, до които стигнахме при наблюдението на различните реализации на системата.

k-NN

След стартиране на k-NN алгоритъма, това което получаваме като резултати са книгите предоставени на скрийншота по-долу. Проблемът на този алгоритъм е, че е бавен и оценките на книгите, които да се препоръчват на читателите е крайно неперсонализирана, защото използваме единствено рейтинга на книгите като характеристика, която да указва кои книги са по-предпочитани от други. Затова решихме, че ще ни е необходим алгоритъм, който да взема предвид годините на читателите като второстепенен критерий за персонализация на оценката. Естествено, ако имахме още информация за книгите (например техните жанрове), книгите можеха да се филтрират и по тенденциозно най-харесваните жанрове, което да ни доведе до по-висока персонализация (препоръките са изведени по предварително зададен ISBN на книга):



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

```
['1558745157',  
[["Left Behind: A Novel of the Earth's Last Days (Left Behind No. 1)",  
0.9381235623430844],  
['Night Sins', 0.9352435834429496],  
['On the Street Where You Live', 0.9258003569774721],  
['A Child Called \\It\\': One Child's Courage to Survive",  
0.7356135842239919],  
["The Lost Boy: A Foster Child's Search for the Love of a Family",  
3.3306690738754696e-16]]]
```

k-NN алгоритъм с регресорна препоръка на книги и грануларно изчисление на дистанциите (между текущо изследвания пример и неговите k на брой най-близки съседи)

При стартиране на k-NN алгоритъма с регресионна оценка на връзката между рейтингите на книгите с възрастовите групи, обособили се като тенденциозни при нашите читатели, получаваме следната препоръка на книги (резултатите са изведени по години на читателя и предпочитан рейтинг на книгите):

```
Recommendations for a 33 years old user who wants to read a book with rating 3:  
Book Title: The Deadhouse, Author: Linda Fairstein  
Book Title: Death a L'Orange, Author: Nancy Fairbanks  
Book Title: Energy Up!: Shed Pounds, Get Fit, Gain Stamina, and Turn on Your Power With This Unique Program, Author: High Voltage  
Book Title: The Billionaire Bridegroom (Passion), Author: Emma Darcy  
Book Title: The Memory of Running, Author: Ron McLarty  
  
Recommendations for a 25 years old user who wants to read a book with rating 5:  
Book Title: Heretics of Dune, Author: Frank Herbert  
Book Title: Flirting with Pete : A Novel, Author: Barbara Delinsky  
Book Title: Call It Sleep, Author: Henry Roth  
Book Title: Marabou Stork Nightmares, Author: Irvine Welsh  
Book Title: Billy, Author: Whitley Strieber
```

На първите няколко реда виждаме препоръките за книги на читател на 33 години, който обича да чете книги с оценка 3 / 10, а на следващите няколко реда виждаме препоръките за книги на читател на 25 години, който обича да чете книги с рейтинг 5 / 10. Тук проблемът, с който се сблъскахме беше, че след тест за корелация между тези 2 характеристики открихме, че линейната връзка не съществува (според метода на Pearson, p-value-то беше равно на 0) и следователно не би следвало изобщо да има логична причина за един такъв модел.

```
Pearson Correlation: -0.0374251240504987  
P-value: 0.0
```

Тук ние обаче решихме да експериментираме и все пак да доразвием този модел без значение коефициента на корелацията и резултатите. Горните препоръки са плод именно на това наше решение. Освен тези препоръки обаче ние решихме да оценим модела:

```
y_pred = knn.predict(X_test)  
y_true = y_test[:10] # Top-10 true ratings (or actual ratings for evaluation).  
y_pred_top_k = y_pred[:10] # Top-10 predicted ratings.  
  
precision = precision_at_k(y_true, y_pred_top_k, k=5)  
recall = recall_at_k(y_true, y_pred_top_k, k=5)
```



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

```
print(f"Precision@5: {precision}")  
print(f"Recall@5: {recall}")
```

Това, което получихме при оценката е:

```
Precision@5: 0.4  
Recall@5: 0.6666666666666666
```

Като тези резултати могат да се интерпретират, че около 40% от препоръчаните книги са релевантни на предпочитанията на въпросния читател и около 67% от всички релевантни предложения за читателя са включени в показаните по-горе препоръки. Това само засили мнението ни, че този модел не е достатъчно подходящ за крайно решение и така продължихме със следващите алгоритми - оптимизации на k-NN алгоритъма като за препоръка **решихме крайно да използваме само рейтинга на книгите**, поради липса на достатъчно допълнителна информация и контекст, които да ни позволяват да изследваме връзките им с рейтинга и да бъдат направени по-точни и смислени препоръки.

a-NN

След тези 2 експеримента ни хрумна идеята, че трябва да използваме някоя от оптимизациите на k-NN алгоритъма. Чудейки се коя точно оптимизация ни трябва, след четене на различни статии из Интернет (**упоменати в Ресурсите в края на този документ**), стигнахме до решението да продължим да експериментираме и да описваме какво сме наблюдавали.

HNSW

След имплементирането на HNSW алгоритъма и неговото стартиране, получихме следните препоръки за книги (резултатите са изведени по предварително въведен ISBN на книга):

```
'1558745157',  
[[['Night Sins', '055356451X'], 0.9014358],  
 [['Icy Sparks', '0142000205'], 0.90126425],  
 [['Left Behind: A Novel of the Earth's Last Days (Left Behind No. 1)',  
   '0842329129'],  
   0.89481175],  
 [['On the Street Where You Live', '0671004530'], 0.8627482],  
 [['A Child Called \\It\\": One Child's Courage to Survive"', '1558743669'],  
   0.63628924]]]  
'0330281747',  
[[['Wild Animus', '0971880107'], 0],  
 [['The Lovely Bones: A Novel', '0316666343'], 0],  
 [['The Da Vinci Code', '0385504209'], 0],  
 [['Divine Secrets of the Ya-Ya Sisterhood: A Novel', '0060928336'], 0],  
 [['The Red Tent (Bestselling Backlist)', '0312195516'], 0]]]
```

Наблюденията ни тук бяха, че отново имаше забавяне в работата на алгоритъма при работа с големи масиви данни, каквито са налице в **dataset-a, предоставен ни от Kaggle**. Друг проблем, който се наблюдава при всички дотук имплементирани алгоритми е, че матриците от изследваните данни изследват само множеството на



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

популярните книги (тези в топ 10%), което също намалява точността на препоръката, изключвайки голяма част от книгите (, смятани за непопулярни).

FAISS

Предимство на FAISS алгоритъма пред дотук имплементираните е именно разглеждането както на популярните, така и на непопулярните книги, следователно този алгоритъм покрива най-много случаи по най-оптимален начин. Оптималността му се дължи на това, че **използва TF-IDF (, подход описан по-горе в този документ)**. на кратко чрез TF-IDF алгоритъмът токенизира заглавията на непопулярните книги и оптимизира размера на матрицата на непопулярните книги. Естествено освен този алгоритъм, ние решихме да имплементираме и ANNOY с цел сравнение на резултатите и производителността, имайки по-широк поглед върху проблема да подберем най-доброто крайно решение.

```
['1558745157',  
[[['A Child Called \\It\\": One Child\\'s Courage to Survive"', '1558743669'],  
0.26438642],  
[['On the Street Where You Live', '0671004530'], 0.07419965],  
[['Night Sins', '055356451X'], 0.064756416],  
[['Left Behind: A Novel of the Earth's Last Days (Left Behind No. 1)",  
'0842329129'],  
0.061876435],  
[['L Is for Lawless', '0449221490'], 0.061851475]]]  
['0330281747',  
[[['The crystal bucket: Television criticism from the Observer, 1976-79',  
'0224018906'],  
0.7108932],  
[['The Box: An Oral History of Television, 1920-1961', '0140252657'],  
0.6335434],  
[['Channels of Discourse, Reassembled: Television and Contemporary Criticism',  
'0807843741'],  
0.61701965],  
[['The Television Detectives, 1948-1978', '0498022366'], 0.57000047],  
[['The Houdini Box', '0679814299'], 0.54777056]]]
```

ANNOY

След имплементацията на ANNOY алгоритъма и неговото стартиране, наблюдавахме следните резултати:

```
['1558745157',  
[[['L Is for Lawless', '0449221490'], 1.3697799444198608],  
[['Left Behind: A Novel of the Earth's Last Days (Left Behind No. 1)",  
'0842329129'],  
1.3697617053985596],  
[['Night Sins', '055356451X'], 1.3676575422286987],  
[['On the Street Where You Live', '0671004530'], 1.3607354164123535],  
[['A Child Called \\It\\": One Child\\'s Courage to Survive"', '1558743669'],  
1.2129415273666382]]]  
['0330281747',  
[[['Wild Animus', '0971880107'], 0],  
[['The Lovely Bones: A Novel', '0316666343'], 0],  
[['The Da Vinci Code', '0385504209'], 0],  
[['Divine Secrets of the Ya-Ya Sisterhood: A Novel', '0060928336'], 0],  
[['The Red Tent (Bestselling Backlist)', '0312195516'], 0]]]
```



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

Това, което се вижда тук е, че след тестването на толкова много и различни алгоритми с еднакви данни, започва да има прилика в препоръките. Това е съвсем нормално, но в началото, когато решавахме задачата с алгоритми като чист k-NN и k-NN с регресия такава тенденция не се наблюдаваше. Това е още една причина да смятаме, че сме на прав път, използвайки оптимизациите на k-NN алгоритъма. Естествено не можем да твърдим, че и ANNOY е най-доброто решение поради бавнодействието му в сравнение пред FAISS и поради лошото му справяне с така наречените “непопулярни книги” и засега като най-оптимално и най-точно решение на проблем като този е алгоритъма FAISS. Това не означава, че ако имаме повече данни, на които да базираме по-многофакторна препоръка, задължително отново бихме избрали FAISS, понеже в една такава хипотетична ситуация щяхме да имаме нуждата от много по-задълбочен анализа на взаимовръзките в графите от данни.

7. Конфигурация на проекта

В тази графа на документацията ще Ви запознаем със стъпките, необходими за успешно конфигуриране и стартиране на изходния код на нашата система на Вашите машини, независимо дали използвате Windows, MacOS или Linux като операционна система.

Windows / MacOS / Linux

Стъпка 1: Направете директория, в която искате да разположите кода ни.

Стъпка 2: Клонирайте следното repository: <https://github.com/Nv4n/soz-project-2024-2025.git> на вашата машина в току-що създадената от Вас директория. Клонирането може да стане през:

- терминала със следната команда: `git clone` **със задаване на директория.** Когато клонирате хранилище, можете директно да зададете целева директория, където да бъде изтеглено:

`git clone <URL-на-репото> <път-към-директорията>`

- UI-я на някое IDE (например Visual Studio Code).

Стъпка 3: Инсталирайте някой Python Extension Pack и необходимите библиотеки, описани по-нагоре в този документ.

Стъпка 4: Отворете някой от файловете с изходния код, стартирайте го и експериментирайте.

Забележка: И за трите операционни системи общите стъпки за конфигурация са еднакви. Разликите могат да идват от това какъв терминал използва самата операционна система или как се извикват конкретни команди.



8. Литература

- **Сетове с данни (16 януари 2025 г.):**
<https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset>
- **Kaggle notebook за по-добро разбиране на данните (16 януари 2025 г.):**
<https://www.kaggle.com/code/arashnic/recom-i-data-understanding-and-simple-recomm>
- **Ръководство за създаване на система за препоръки в 1 час (част 1 - 17 януари 2025 г.):** <https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-the-fundamentals-dfee054f978e>
- **Ръководство за създаване на система за препоръки в 1 час (част 2 - 17 януари 2025 г.):** <https://towardsdatascience.com/how-did-we-build-book-recommender-systems-in-an-hour-part-2-k-nearest-neighbors-and-matrix-c04b3c2ef55c>
- **Еволюция на алгоритмите за системи, отправящи препоръки (част 1 - 18 януари 2025 г.):** <https://medium.com/@anicomanesh/evolution-of-recommendation-algorithms-part-i-fundamentals-and-classical-recommendation-bb1c0bce78a9>
- **Алгоритъм за k най-близки съседа (19 януари 2025 г.):**
<https://medium.com/@leam.a.murphy/personalized-book-recommendations-with-k-nearest-neighbors-442ce4dad44c>
- **Система за препоръки на книги чрез k-NN алгоритъма (19 януари 2025 г.):**
<https://www.geeksforgeeks.org/recommender-systems-using-knn/>
- **Евклидово разстояние (20 януари 2025 г.):**
<https://www.geeksforgeeks.org/euclidean-distance/>
- **Манхатанско разстояние (20 януари 2025 г.):**
<https://www.datacamp.com/tutorial/manhattan-distance>
- **Три начина за регресионна система за препоръки (20 януари 2025 г.):**
<https://www.sciencedirect.com/science/article/abs/pii/S0020025516301669>
- **Грануларно изчисление на разстоянията (20 януари 2025 г.):**
<https://www.sciencedirect.com/topics/computer-science/granular-computing>
- **ANN алгоритъм и неговите разновидности (21 януари 2025 г.):**
<https://rtriangle.hashnode.dev/approximate-nearest-neighbors-algorithms-and-libraries>
- **Снимки за илюстрация на използваните алгоритми (22 януари 2025 г.):**
<https://www.almabetter.com/bytes/articles/types-of-recommendation-systems-how-they-work-and-their-use-cases>