```
import os
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
```

```
import zipfile

with zipfile.ZipFile('BX-CSV-Dump.zip', 'r') as zipref:
    zipref.extractall('datasets')
```

Создание рекомендательной системы

Датасет книг с оценками пользователей

```
path = './datasets/'
ratings = pd.read_csv(path + 'BX-Book-Ratings.csv', encoding='windows-1251', sep=';')
books = pd.read_csv(path + 'BX-Books.csv', encoding='windows-1251', sep=';',
                   usecols=['ISBN',
                            'Book-Title',
                            'Book-Author',
                            'Year-Of-Publication',
                            'Publisher',
                            'Image-URL-S',
                            'Image-URL-M',
                            'Image-URL-L'])
users = pd.read_csv(path + 'BX-Users.csv', encoding='windows-1251', sep=';')

ratings = ratings.rename(columns={'Book-Rating': 'Rating'})
ratings["Rating"] = ratings["Rating"].astype("int8")
books.drop(['Image-URL-S', 'Image-URL-M', 'Image-URL-L'], axis=1, inplace=True)
books = books.rename(columns={'Book-Title': 'Title', 'Book-Author': 'Author'})
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWa
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Средние значения и количество

```
avg_user_rating = ratings.groupby('User-ID')['Rating'].mean()
cnt_user_rating = ratings.groupby('User-ID')['Rating'].count()

avg_book_rating = ratings.groupby('ISBN')['Rating'].mean()
cnt_book_rating = ratings.groupby('ISBN')['Rating'].count()
```

```
avg_user_rating.name = 'avg_rating'
cnt_user_rating.name = 'N_ratings'
avg_book_rating.name = 'avg_rating'
cnt_book_rating.name = 'N_ratings'
```

```
users = users.merge(avg_user_rating, on=['User-ID'])
users = users.merge(cnt_user_rating, on=['User-ID'])

books = books.merge(avg_book_rating, on=['ISBN'])
books = books.merge(cnt_book_rating, on=['ISBN'])
```

KNN model - модель ближайших соседей (рекомендация похожей книги)

Найти наиболее похожую книгу (используя оценки пользователей) и порекомендовать ее.

Выбираем книги у которых более 20 оценок от пользователей

```
pd_matrix = \
    pd.merge(books.loc[books["N_ratings"] > 20, "ISBN"],
             ratings, how="left", left_on="ISBN", right_on="ISBN").drop_duplicates()

pd_matrix = pd_matrix.pivot(index='ISBN', columns='User-ID', values='Rating').fillna(0).as
```

```
# Сжимаем матрицу
matrix = csr_matrix(pd_matrix.values)
```

```
# Создаем модель
N_predicted_neighbours = 11
KNN = NearestNeighbors(metric='cosine', n_neighbors=N_predicted_neighbours, n_jobs=-1)
# Обучение модели
KNN.fit(matrix)
```

```
     NearestNeighbors(metric='cosine', n_jobs=-1, n_neighbors=11)
```

```
# Предсказание
distances, indices = KNN.kneighbors(matrix)
distances.shape, indices.shape
```

```
     ((6863, 11), (6863, 11))
```

```
# Предсказание 11 книг
```

```
print(f"Because you liked {books.loc[books['ISBN'] == pd_matrix.index[indices[489][0]], 'T
print()
for i in range(1, N_predicted_neighbours):
    print(f"{books.loc[books['ISBN'] == pd_matrix.index[indices[489][i]], 'Title'].values[
```

```
Because you liked Princess in the Spotlight (The Princess Diaries, Vol. 2) you may li

The Princess Diaries with distance 0.502.
Gossip Girl #1 : A Novel by Cecily von Ziegesar (Gossip Girl) with distance 0.752.
Knocked Out by My Nunga-Nungas : Further, Further Confessions of Georgia Nicolson (Co
Irish Chain (Benni Harper Mysteries (Paperback)) with distance 0.841.
Whatever Happened to Janie? with distance 0.851.
Amanda's Wedding: A Novel with distance 0.861.
Emily of New Moon with distance 0.867.
The Second Summer of the Sisterhood with distance 0.872.
It's Always Something with distance 0.875.
The Book of Three (Chronicles of Prydain (Paperback)) with distance 0.882.
```

```
def recommend_similar_book(isbn, indices, ratings_matrix, books_table, N_recommendations=1
    """
    Recommends a book title.

    Parameters
    ----------
    ISBN: str
        ISBN of a book a user liked
    indices: np.array
        indices of ratings_matrix as predicted by KNN
    ratings_matrix: pd.DataFrame
        user-book-rating matrix with ratings as values
    N_recommendations: int (default 1)
        How many books to recommend?
    distances: np.array
        How distant are books from each other by KNN?
    """
    # Возврат рекомендации
    print(f"Because you liked {books_table.loc[books_table['ISBN'] == ratings_matrix.index
    print()
    for i in range(1, 1+N_recommendations):
        if distances:
            print(f"{books_table.loc[books_table['ISBN'] == ratings_matrix.index[indices[i
        else:
            print(f"{books_table.loc[books_table['ISBN'] == ratings_matrix.index[indices[i
```

```
recommend_similar_book(489, indices, pd_matrix, books)
```

```
Because you liked Princess in the Spotlight (The Princess Diaries, Vol. 2) you may li

The Princess Diaries.
```

▾ KNN model - предлагает любимую книгу пользователя

На основе книги, которую мы задаем, модель находит похожих пользователей и предлагает другие книги на основании их прочтений

```
# Создание модели
KNN2 = NearestNeighbors(metric='cosine', n_neighbors=20, n_jobs=-1)


# Обучение
KNN2.fit(matrix.T)
```

```
NearestNeighbors(metric='cosine', n_jobs=-1, n_neighbors=20)
```

```
%%time

# Предсказание
distances2, indices2 = KNN2.kneighbors(matrix.T)
```

```
CPU times: user 2min 51s, sys: 3.13 s, total: 2min 54s
Wall time: 2min 7s
```

```
def recommend_favourite_book_of_similar_user(userID, indices, ratings_matrix, users_table,
    """
    Recommends a book title based on favourite books of ten most similar users.

    The order of books is following:
    Take the most similar user, sort his books by rating,
    exclude everything the current predicted user already read.
    Output books one by one.
    If there is only a few books from the most similar user and
    we run out of books, take next similar user and output
    his favorite books in a similar fashion.

    Parameters
    ----------
    userID: int
        ID of a user we want a recommendation for
    indices: np.array
        indices of ratings_matrix as predicted by KNN
    ratings_matrix: pd.DataFrame
        user-book-rating matrix with ratings as values
    users_table: pd.DataFrame
        Information about users
    books_table: pd.DataFrame
        Information about books
    ratings_table: pd.DataFrame
        Information about ratings
    N_recommendations: int (default 1)
        How many books to recommend?
    distances: np.array
        How distant are books from each other by KNN?
    """
    selected_index = ratings_matrix.columns.get_loc(userID)
    already_read_book_isbns = list(ratings_table.loc[ratings_table["User-ID"] == userID, "
    not_read_books = ratings_table.loc[~ratings_table["ISBN"].isin(already_read_book_isbns
    books_to_recommend = list()
    for i in range(1,10):
        similar_user_index = indices[selected_index][i]
        similar_user_ID = ratings_matrix.columns[similar_user_index]
        possible_to_recommend = not_read_books.loc[not_read_books["User-ID"] == similar_us
        possible_to_recommend = possible_to_recommend.sort_values(by="Rating", ascending=F
        for a, row in possible_to_recommend.iterrows():
            books_to_recommend.append(books_table.loc[books["ISBN"] == row["ISBN"], "Title
            if len(books_to_recommend) > N_recommendations-1:
                break
        if len(books_to_recommend) > N_recommendations-1:
            break
    print(f"Based on users who like similar books as you, you may like:")
    print()
    for book_name in books_to_recommend:
        print(book_name)


recommend_favourite_book_of_similar_user(175002,
                                         indices2,
                                         pd_matrix,
                                         users,
                                         books,
                                         ratings,
                                         N_recommendations=3,
                                         distances=distances2)
```

```
Based on users who like similar books as you, you may like:

The First Immortal
Nightswimmer: A Novel
Rockets, Redheads &amp; Revolution
```

● ✕