Project: Book recommendation model using K-Nearest Neighbors

Objective: create a book recommendation algorithm using **K-Nearest Neighbors**

We will use the Book-Crossings dataset. This dataset contains 1.1 million ratings (scale of 1-10) of 270,000 books by 90,000 users.

After importing and cleaning the data, use NearestNeighbors from sklearn.neighbors to develop a model that shows books that are similar to a given book. The Nearest Neighbors algorithm measures the distance to determine the "closeness" of instances.

Create a function named get_recommends that takes a book title (from the dataset) as an argument and returns a list of 5 similar books with their distances from the book argument.

This code:

get_recommends("The Queen of the Damned (Vampire Chronicles (Paperback))") should return:

['The Queen of the Damned (Vampire Chronicles (Paperback))', [['Catch 22', 0.7939835159073301], ['The Witching Hour (Lives of the Mayfair Witches)', 0.7448656558990479], ['Interview with the Vampire', 0.7345068454742432], ['The Tale of the Body Thief (Vampire Chronicles (Paperback))', 0.5376338362693787], ['The Vampire Lestat (Vampire Chronicles, Book II)', 0.5177841219902086]]]

- The data returned from get_recommends() is a list. The first element in the list is the book title passed into the function. The second element in the list is a list of five more lists. Each of the five lists contains a recommended book and the distance from the recommended book to the book passed into the function.

If graph the dataset (optional), most books are not rated frequently. To ensure statistical significance, remove from the dataset users with less than 200 ratings and books with less than 100 ratings.

In [ ]:
```
# Import libraries (you may add additional imports but you may not have to)
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

Import data

In [ ]:
```
# get data files
!wget https://cdn.freecodecamp.org/project-data/books/book-crossings.zip

!unzip book-crossings.zip
```

```
--2023-07-03 20:20:08--  https://cdn.freecodecamp.org/project-data/books/book-crossings.zip
Resolving cdn.freecodecamp.org (cdn.freecodecamp.org)... 172.67.70.149, 104.26.3.33, 104.26.2.33
Connecting to cdn.freecodecamp.org (cdn.freecodecamp.org)|172.67.70.149|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26085508 (25M) [application/zip]
Saving to: 'book-crossings.zip'

book-crossings.zip  100%[===================>]  24,88M  2,82MB/s    in 20s

2023-07-03 20:20:28 (1,25 MB/s) - 'book-crossings.zip' saved [26085508/26085508]

Archive:  book-crossings.zip
  inflating: BX-Book-Ratings.csv
  inflating: BX-Books.csv
  inflating: BX-Users.csv
```

In [ ]:
```
books_filename = 'BX-Books.csv'
ratings_filename = 'BX-Book-Ratings.csv'
```

In [ ]:
```
# Import csv data into dataframes
df_books = pd.read_csv(
    books_filename,
    encoding = "ISO-8859-1",
    sep=";",
    header=0,
    names=['isbn', 'title', 'author'],
    usecols=['isbn', 'title', 'author'],
    dtype={'isbn': 'str', 'title': 'str', 'author': 'str'})

df_ratings = pd.read_csv(
    ratings_filename,
    encoding = "ISO-8859-1",
    sep=";",
    header=0,
    names=['user', 'isbn', 'rating'],
    usecols=['user', 'isbn', 'rating'],
    dtype={'user': 'int32', 'isbn': 'str', 'rating': 'float32'})

df_books.head()
```

Out[ ]:

	isbn	title	author
0	0195153448	Classical Mythology	Mark P. O. Morford
1	0002005018	Clara Callan	Richard Bruce Wright
2	0060973129	Decision in Normandy	Carlo D'Este
3	0374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata
4	0393045218	The Mummies of Urumchi	E. J. W. Barber

In [ ]:
```
df_ratings.head()
```

Out[ ]:

	user	isbn	rating
0	276725	034545104X	0.0
1	276726	0155061224	5.0
2	276727	0446520802	0.0
3	276729	052165615X	3.0
4	276729	0521795028	6.0

NULL data

In [ ]:
```
df_books.isnull().sum()
```

Out[ ]:
```
isbn     0
title    0
author   1
dtype: int64
```

In [ ]:
```
df_ratings.isnull().sum()
```

Out[ ]:
```
user      0
isbn      0
rating    0
dtype: int64
```

In [ ]:
```
df_books.dropna(inplace=True)
```

In [ ]:
```
df_books.isnull().sum()
```

Out[ ]:
```
isbn      0
title     0
author    0
dtype: int64
```

Remove users with less than 200 ratings

In [ ]:
```
df_ratings.shape
```

Out[ ]:
```
(1149780, 3)
```

In [ ]:
```
ratings = df_ratings['user'].value_counts()
ratings.sort_values(ascending=False).head()
```

Out[ ]:
```
11676     13602
198711     7550
153662     6109
98391      5891
35859      5850
Name: user, dtype: int64
```

In [ ]:
```
len(ratings[ratings < 200])
```

Out[ ]:
```
104378
```

In [ ]:
```
df_ratings['user'].isin(ratings[ratings < 200].index).sum()
```

Out[ ]:
```
622224
```

In [ ]:
```
df_ratings_rm = df_ratings[
    ~df_ratings['user'].isin(ratings[ratings < 200].index)
]
df_ratings_rm.shape
```

Out[ ]:
```
(527556, 3)
```

Remove books with less than 100 ratings

In [ ]:
```
ratings = df_ratings['isbn'].value_counts() # we have to use the original df_ratings to pass the challenge
ratings.sort_values(ascending=False).head()
```

Out[ ]:
```
0971880107    2502
0316666343    1295
0385504209     883
0312195516     732
0060928336     723
Name: isbn, dtype: int64
```

In [ ]:
```
len(ratings[ratings < 100])
```

Out[ ]:
```
339825
```

In [ ]:
```
df_books['isbn'].isin(ratings[ratings < 100].index).sum()
```

Out[ ]:
```
269442
```

In [ ]:
```
df_ratings_rm = df_ratings_rm[
    ~df_ratings_rm['isbn'].isin(ratings[ratings < 100].index)
]
df_ratings_rm.shape
```

Out[ ]:
```
(49781, 3)
```

In [ ]:
```
# These should exist
books = ["Where the Heart Is (Oprah's Book Club (Paperback))",
         "I'll Be Seeing You",
         "The Weight of Water",
         "The Surgeon",
         "I Know This Much Is True"]

for book in books:
    print(df_ratings_rm.isbn.isin(df_books[df_books.title == book].isbn).sum())
```

```
183
75
49
57
77
```

Prepare dataset for KNN

In [ ]:
```
df = df_ratings_rm.pivot_table(index='user',columns='isbn',values='rating').fillna(0).T
df.head()
```

Out[ ]:

user	254	2276	2766	2977	3363	4017	4385	6242	6251	6323	...	274004	274061	274301	274308	274808	275970	277427	277478	277639	278418
isbn																					
0025477098	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	0.0
0060008032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0060096195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
006016848X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0060173289	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 888 columns

In [ ]:
```
df.index = df.join(df_books.set_index('isbn'))['title']
```

In [ ]:
```
df = df.sort_index()
df.head()
```

Out[ ]:

user	254	2276	2766	2977	3363	4017	4385	6242	6251	6323	...	274004	274061	274301	274308	274808	275970	277427	277478	277639	278418
title																					
1984	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1st to Die: A Novel	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2nd Chance	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2nd Chance	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 888 columns

In [ ]:
```
df.loc["The Queen of the Damned (Vampire Chronicles (Paperback))"][:5]
```

Out[ ]:
```
user
254     0.0
2276    0.0
2766    0.0
2977    0.0
3363    0.0
Name: The Queen of the Damned (Vampire Chronicles (Paperback)), dtype: float32
```

KNN model

In [ ]:
```
model = NearestNeighbors(metric='cosine')
model.fit(df.values)
```

Out[ ]:
```
NearestNeighbors(metric='cosine')
```

get_recommends()

In [ ]:
```
df.loc[0].shape
```

Out[ ]:
```
(888,)
```

In [ ]:
```
title = 'The Queen of the Damned (Vampire Chronicles (Paperback))'
df.loc[title].shape
```

Out[ ]:
```
(888,)
```

In [ ]:
```
distance, indice = model.kneighbors([df.loc[title].values], n_neighbors=6)

print(distance)
print(indice)
```

```
[[0.         0.51784116 0.51763384 0.73450685 0.74486566 0.7939835 ]]
[[412 668 408 272 667 638]]
```

In [ ]:
```
df.iloc[indice[0]].index.values
```

Out[ ]:
```
array(['The Queen of the Damned (Vampire Chronicles (Paperback))',
       'The Vampire Lestat (Vampire Chronicles, Book II)',
       'The Tale of the Body Thief (Vampire Chronicles (Paperback))',
       'Interview with the Vampire',
       'The Witching Hour (Lives of the Mayfair Witches)', 'Catch 22'],
      dtype=object)
```

In [ ]:
```
pd.DataFrame({
    'title': df.iloc[indice[0]].index.values,
    'distance': distance[0]
}) \
.sort_values(by='distance', ascending=False)
```

Out[ ]:

	title	distance
5	Catch 22	0.793984
4	The Witching Hour (Lives of the Mayfair Witches)	0.744866
3	Interview with the Vampire	0.734507
2	The Tale of the Body Thief (Vampire Chronicles...	0.537634
1	The Vampire Lestat (Vampire Chronicles, Book II)	0.517841
0	The Queen of the Damned (Vampire Chronicles (P...	0.000000

In [ ]:
```
# function to return recommended books - this will be tested
def get_recommends(title = ""):
  try:
    book = df.loc[title]
  except KeyError as e:
    print('The given book', e, 'does not exist')
    return

  distance, indice = model.kneighbors([book.values], n_neighbors=6)

  recommended_books = pd.DataFrame({
      'title': df.iloc[indice[0]].index.values,
      'distance': distance[0]
  }) \
  .sort_values(by='distance', ascending=False) \
  .head(5).values

  return [title, recommended_books]
```

In [ ]:
```
get_recommends("The Queen of the Damned (Vampire Chronicles (Paperback))")
```

Out[ ]:
```
["The Queen of the Damned (Vampire Chronicles (Paperback))",
 array([['Catch 22', 0.7939835159073301],
        ['The Witching Hour (Lives of the Mayfair Witches)',
         0.7448656558990479],
        ['Interview with the Vampire', 0.7345068454742432],
        ['The Tale of the Body Thief (Vampire Chronicles (Paperback))',
         0.5376338362693787],
        ['The Vampire Lestat (Vampire Chronicles, Book II)',
         0.5177841219902086]], dtype=object)]
```

Prediction

In [ ]:
```
books = get_recommends("Where the Heart Is (Oprah's Book Club (Paperback))")
print(books)

def test_book_recommendation():
  test_pass = True
  recommends = get_recommends("Where the Heart Is (Oprah's Book Club (Paperback))")
  if recommends[0] != "Where the Heart Is (Oprah's Book Club (Paperback))":
    test_pass = False
  recommended_books = ["I'll Be Seeing You", 'The Weight of Water', 'The Surgeon', 'I Know This Much Is True']
  recommended_books_dist = [0.8, 0.77, 0.77, 0.77]
  for i in range(5):
    if recommends[1][i][0] not in recommended_books:
      test_pass = False
    if abs(recommends[1][i][1] - recommended_books_dist[i]) >= 0.05:
      test_pass = False


test_book_recommendation()
```

```
["Where the Heart Is (Oprah's Book Club (Paperback))", array([["I'll Be Seeing You", 0.8016210754448853],
       ['The Weight of Water', 0.7708583474151241],
       ['The Surgeon', 0.7699429931574756],
       ['I Know This Much Is True', 0.7677875266838074],
       ['The Lovely Bones: A Novel', 0.7234046234924316]], dtype=object)]
```