ADNAN MENDERES UNIVERSITY

CSE424 BIG DATA ANALYSIS

Term Project

Recommendation System with Spark

171805024 – Nagihan Baz

171805071 – Emin Hasanzade

181805052 – Elize Hamitoğlu

181805039 – Osman Melih Tolunay

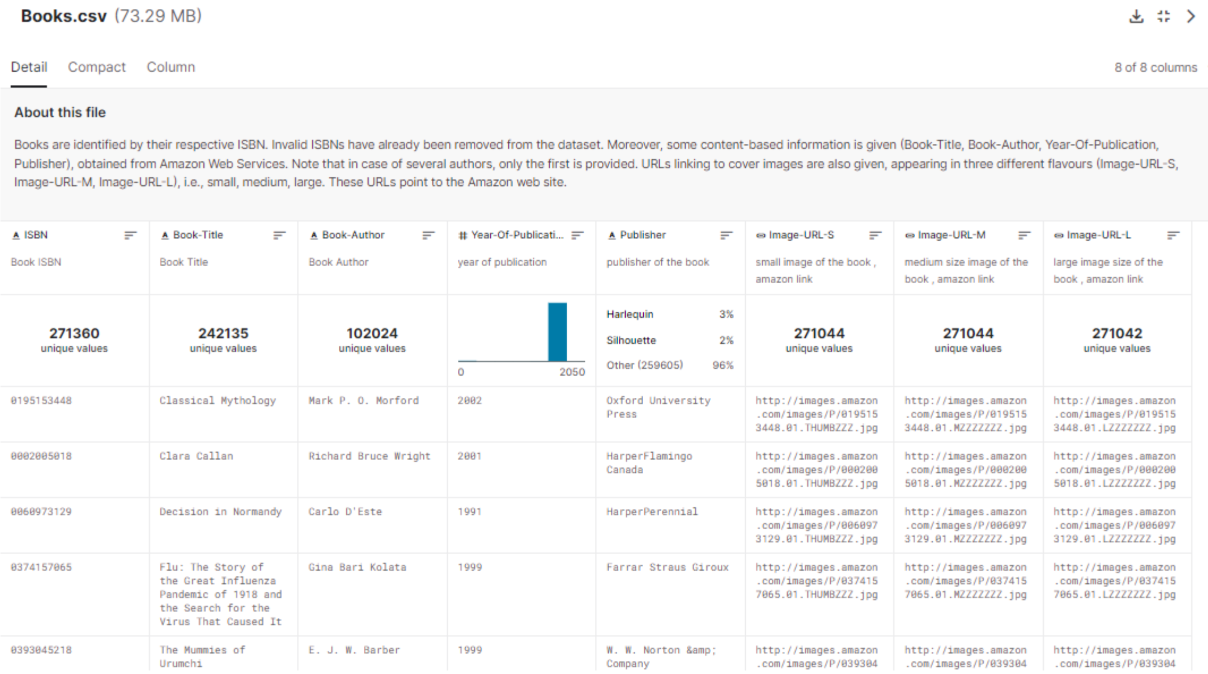
Dataset Information

This dataset is large enough to build good recommendation model and you may face memory issue while training a model using that dataset. Dataset includes books, users and book ratings files.

Books File Description: ISBN, Book-Title, Book-Author, Year-Of-Publication, Publisher, Image-URL-S, Image-URL-M, Image-URL-L.

Book Ratings File Description: User-ID, ISBN, Book-Rating.

Users File Description: User-ID, Location, Age.



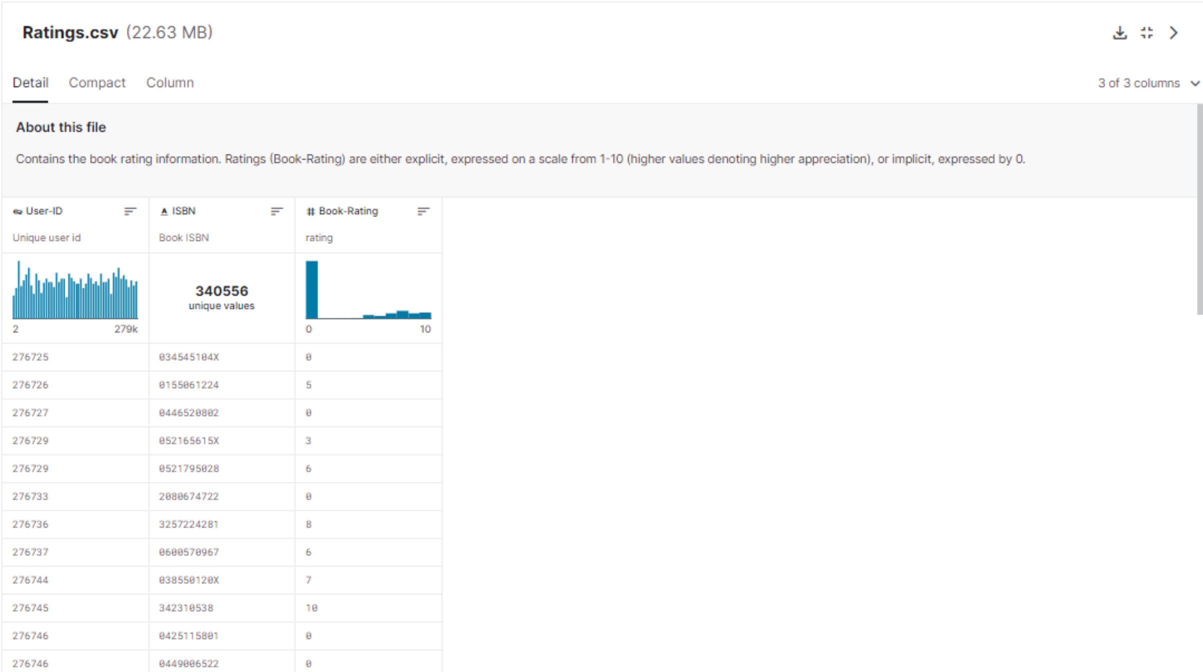
Books.csv (73.29 MB)

Detail Compact Column — 8 of 8 columns

About this file

Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (Book-Title, Book-Author, Year-Of-Publication, Publisher), obtained from Amazon Web Services. Note that in case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (Image-URL-S, Image-URL-M, Image-URL-L), i.e., small, medium, large. These URLs point to the Amazon web site.

ISBN	Book-Title	Book-Author	Year-Of-Publicati...	Publisher	Image-URL-S	Image-URL-M	Image-URL-L
Book ISBN	Book Title	Book Author	year of publication	publisher of the book	small image of the book , amazon link	medium size image of the book , amazon link	large image size of the book , amazon link
271360 unique values	242135 unique values	102024 unique values	0 – 2050	Harlequin 3%; Silhouette 2%; Other (259605) 96%	271044 unique values	271044 unique values	271042 unique values
0195153448	Classical Mythology	Mark P. O. Morford	2002	Oxford University Press	http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg	http://images.amazon.com/images/P/0195153448.01.MZZZZZZZ.jpg	http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg
0002005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg	http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg	http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg
0060973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg	http://images.amazon.com/images/P/0060973129.01.MZZZZZZZ.jpg	http://images.amazon.com/images/P/0060973129.01.LZZZZZZZ.jpg
0374157065	Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It	Gina Bari Kolata	1999	Farrar Straus Giroux	http://images.amazon.com/images/P/0374157065.01.THUMBZZZ.jpg	http://images.amazon.com/images/P/0374157065.01.MZZZZZZZ.jpg	http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg
0393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	http://images.amazon.com/images/P/039304	http://images.amazon.com/images/P/039304	http://images.amazon.com/images/P/039304



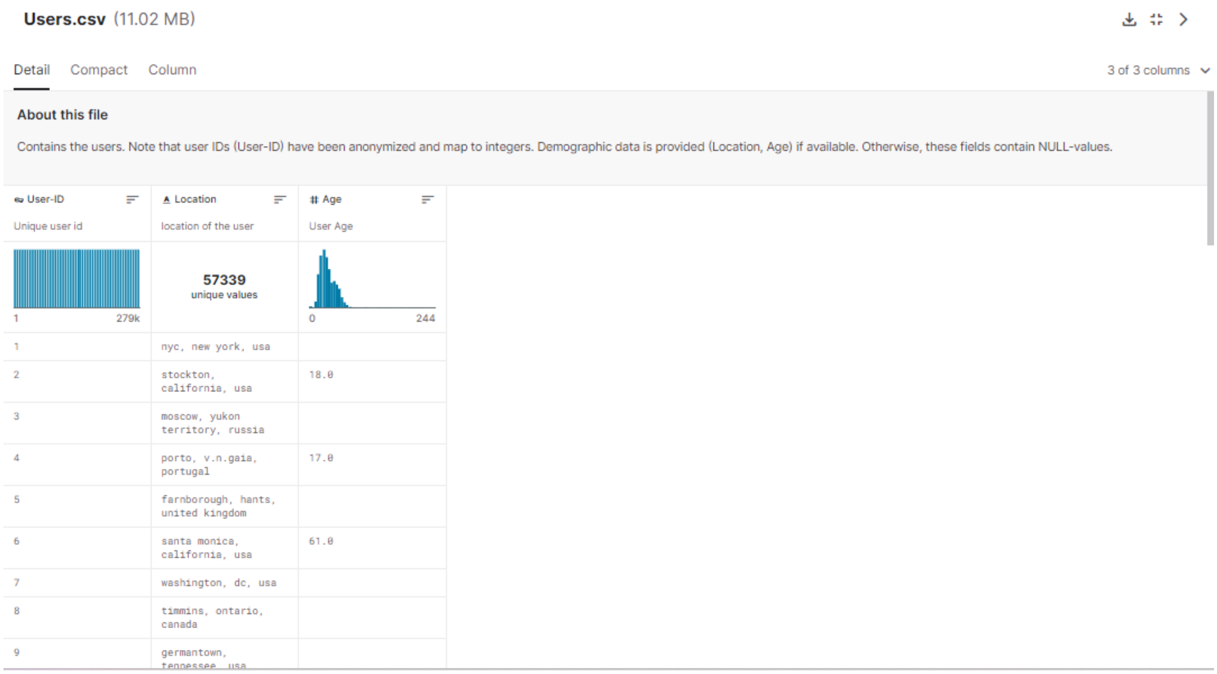
Ratings.csv (22.63 MB)

Detail Compact Column — 3 of 3 columns

About this file

Contains the book rating information. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0.

User-ID	ISBN	Book-Rating
Unique user id	Book ISBN	rating
2 – 279k	340556 unique values	0 – 10
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
276746	0425115801	0
276746	0449006522	0



Reads files from specified path using textFile with 'sc' method. Returns an RDD where each item represents a row in the file.

```
In [3]: dfUser = sc.textFile("C:/Users/USER/Desktop/dataset/Users.csv")
        dfBooks = sc.textFile("C:/Users/USER/Desktop/dataset/Books.csv")
        dfRatings = sc.textFile("C:/Users/USER/Desktop/dataset/Book-Ratings.csv")
```

Split DF's for each fields

```
In [4]: users_fields = dfUser.map(lambda lines: lines.replace('"','').split(";"))
        books_fields = dfBooks.map(lambda lines: lines.replace('"','').split(";"))
        rating_fields = dfRatings.map(lambda lines: lines.replace('"','').split(";"))
```

This code creates a new RDD, clearing and transforming the values for further analysis.

```
In [5]: def handle_isbn(lst2):
            lst = lst2.collect()
            arr=[]
            arr2=[]
            for i in lst:
                arr.append(re.sub("[^0-9]",'1',i))
            for i in arr:
                if len(i)>=9:
                    arr2.append(int(i[len(i)-9:]))
            return sc.parallelize(arr2)

        def handle_isbn_str(line):
            line1 = re.sub("[^0-9]",'1',line)
            if len(line1)>=9:
                line2 = int(line1[len(line1)-9:])
                return int(line2)
            else:
                return int(line1)
```

USERS Fields & BOOKS Fields & BOOK Rating Fields

```
In [7]: users_id = users_fields.map(lambda field: field[0])
        users_location = users_fields.map(lambda field: field[1])
        users_age = users_fields.map(lambda field: field[2])

        books_isbn = handle_isbn(books_fields.map(lambda field: field[0]))
        books_name = books_fields.map(lambda field: field[1])
        books_writer = books_fields.map(lambda field: field[2])
        books_press = books_fields.map(lambda field: field[3])

        rating_users_id = rating_fields.map(lambda field: float(field[0]))
        rating_books_isbn = handle_isbn(rating_fields.map(lambda field: field[1]))
        rating_books_rating = rating_fields.map(lambda field: float(field[2]))
```

Calculate baseline statistics for filtered age values.

In [8]:
```
age_list = users_age.map(lambda item: handle_age(item))
age_list_filtered = age_list.filter(lambda item: item <= 90 and item >=0 )
age_counts = collections.OrderedDict(sorted(age_list_filtered.countByValue().items()))
age_list_filtered.stats()
```

Out[8]:
```
(count: 167666, mean: 34.54562045972393, stdev: 13.769988967606224, max: 90.0, min: 0.0)
```

USERS' Countries Location & TOP 10 Countries That The Most Users Data Collected

In [10]:
```
location_fields = users_location.map(lambda x: x.replace('"""','').split(','))
location_country = location_fields.map(lambda x: x[-1])

countryCounts = location_country.map(lambda word: (word,1)).reduceByKey(lambda a,b: a + b)
orderedCountryCounts=countryCounts.sortBy(lambda x : x[1],ascending=False)
print(f'TOP 10 Countries That The Most Data Collected: ')
orderedCountryCounts.take(10)
```

```
TOP 10 Countries That The Most Data Collected:
```

Out[10]:
```
[(' usa', 76495),
 (' spain', 12691),
 (' united kingdom', 12541),
 (' canada', 11385),
 (' germany', 10642),
 (' australia', 8896),
 (' italy', 5377),
 (' france', 3037),
 (' portugal', 2721),
 (' new zealand', 2026)]
```

Calculates basic statistics for books ratings.

In [12]:
```
rating_books_rating.stats()
```

Out[12]:
```
(count: 397245, mean: 7.601852760890583, stdev: 1.8412729800421157, max: 10.0, min: 1.0)
```

Computer Information

In [2]:
```
import socket
import platform
hostname = socket.gethostname()
ip = socket.gethostbyname(hostname)
uname = platform.uname()
print(f"Hostname: {hostname}")
print(f"Ip address: {ip}")
print(f"System: {uname.system}")
print(f"Release: {uname.release}")
print(f"Version: {uname.version}")
print(f"Machine: {uname.machine}")
print(f"Processor: {uname.processor}")
```

```
Hostname: LAPTOP-UEQCSDT9
Ip address: 192.168.0.20
System: Windows
Release: 10
Version: 10.0.22621
Machine: AMD64
Processor: Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
```

MSE (Mean Squared Error)

In [93]:
```
print("Mean Squared Error Scores For Each Model:", )
c = 0
for i in modelsArray:
    print(f'Model {arr[c]} = {i[-2]}')
    c+=1
```

```
Mean Squared Error Scores For Each Model:
Model 10_10_01 = 53.31545601995979
Model 10_50_01 = 18.02675073795958
Model 10_200_01 = 21.837385101121505
Model 10_10_1 = 10.94068188592083
Model 10_50_1 = 13.861513642908823
Model 10_200_1 = 10.534852160638245
Model 50_10_01 = 36.69128644997727
Model 50_50_01 = 16.16370876131336
Model 50_200_01 = 17.51192248537928
Model 50_10_1 = 10.267777032909267
Model 50_50_1 = 11.94532434385151
Model 50_200_1 = 9.970174517018798
Model 200_10_01 = 32.01450161397515
Model 200_10_1 = 14.741212207519643
Model 200_50_01 = 16.57593515367748
Model 200_50_1 = 10.222840141615677
Model 200_200_01 = 11.863358392374645
Model 200_200_1 = 9.980126049794423
```

To determine the best model for the dataset based on the Mean Squared Error (MSE) scores provided, we need to identify the model with the lowest MSE score. A lower MSE indicates better accuracy and closer predictions to the actual values.

From the given scores, the model with the lowest MSE is "Model 200_200_1" with an MSE of 9.980126049794423. This model has the smallest error, indicating it provides the closest predictions to the actual values in the dataset. Therefore, based on the provided MSE scores, "Model 200_200_1" is considered the best model for the dataset. It achieves the lowest MSE, indicating better accuracy in predicting the target variable compared to other models.

Root Mean Squared Error

In [94]:
```
print("Root Mean Squared Error Scores For Each Model:", )
c = 0
for i in modelsArray:
    print(f'Model {arr[c]} = {i[-1]}')
    c+=1
```

```
Root Mean Squared Error Scores For Each Model:
Model 10_10_01 = 7.301743354840664
Model 10_50_01 = 4.245792121378481
Model 10_200_01 = 4.673048801491539
Model 10_10_1 = 3.3076701597832923
Model 10_50_1 = 3.723105376283194
Model 10_200_1 = 3.2457436991602164
Model 50_10_01 = 6.057333278760321
Model 50_50_01 = 4.020411516413881
Model 50_200_01 = 4.184724899605621
Model 50_10_1 = 3.2043372220958997
Model 50_50_1 = 3.4562008540956515
Model 50_200_1 = 3.1575583156956575
Model 200_10_01 = 5.658135878005684
Model 200_10_1 = 3.8394286303458802
Model 200_50_01 = 4.0713554442811155
Model 200_50_1 = 3.197317647906707
Model 200_200_01 = 3.444322631864594
Model 200_200_1 = 3.1591337499058856
```

To determine the best model for the dataset based on the Root Mean Squared Error (RMSE) scores provided, we need to identify the model with the lowest RMSE score. A lower RMSE indicates better accuracy and closer predictions to the actual values.

From the given scores, the model with the lowest RMSE is "Model 200_200_1" with an RMSE of 3.1591337499058856. This model has the smallest error, indicating it provides the closest predictions to the actual values in the dataset. Therefore, based on the provided RMSE scores, "Model 200_200_1" is considered the best model for the dataset. It achieves the lowest RMSE, indicating better accuracy in predicting the target variable compared to other models.

Work Sharing Policy

Nagihan does the initial research of the project and finds some sample code and different datasets and is responsible for ALS learning and merging the dataset, removing punctuation, removing blank and repetitive items. (Extracted and filtered RDDs.)

Elize created a histogram for the average age, printed the location of the users' countries and the top 10 countries with the most user data collected by code.

Emin was responsible for the cosine similarity function and built the model using the ALS train dataset.

Melih was helpful in researching and writing all the codes in general.