# Final Project Report

*Goodreads books data analysis*

**Khalid Alabyad 201-700-500**

**Moaaz Al-Akel 201-600-819**

**Omar Al-Suntawi 201-600-337**

**Reem Farahat 201-700-543**

## INTRODUCTION

This report contains the discussion and analysis of book data collected from a google site which contains data collected from goodreads api.

We used spark library to analyse the data with python code running on google colab.

All the  codes and the results are in a google drive folder here

## Data Info

We downloaded the data using a python script on google colab

The data we collected were divided into 3 sections: Book data, Reviews, interaction

- Books (Met-Data of books): it has several files
  - Detailed book graph
  - Detailed information of author
  - Detailed information of works
  - Detailed information of book series
  - Extracted fuzzy book genres
- Reviews (Book Review Texts):
  - Book reviews (complete)
  - Book reviews (spoiler subset)
  -
- Interactions (User’s book shelves):
  - User book interaction

## Requirements:

- Data analysis
  - Most genre have books
  - Most language for every genre
  - Most number of words for one writer
  - Most books have reviews
  - Most readed books
  - Most “want to read” books

    - Most year have published books
    - Highest book rated for every genre
    - Most double genre for books
    - Most Writer with genre

- Machine Learning:
    - recommendation

## Data Analysis

We wrote functions for every requirement to be used as a map reduce functions with spark

    - Most genre have books

```
#use this function and do map and reduce(wordcount)
def get_AllGenres(book_metadata):
    """
    this function output every genre assigned to every book in the
book_metadata goodreads_books.json.gz
    """
    genre_list=[]
    book_dict=json.loads(book_metadata)
    if len(book_dict['popular_shelves']) != 0:
        for genre in  book_dict['popular_shelves']:
            #it is a list of genres
            genre_list.append(genre['name'])
    return genre_list
```

- most language for every genre

```
# use only this function as it should return for every book a list of
genres associated with the language of the book
def get_Lang_with_Genres(book_metadata):
    """
    this function output every genre assigned to every book in the
book_metadata goodreads_books.json.gz
    """
    genre_list=[]
    lang_genre_list=[]
    book_dict=json.loads(book_metadata)
    book_lang=book_dict['language_code']
    if len(book_dict['popular_shelves']) != 0:
        for genre in  book_dict['popular_shelves']:
            #it is a list of genres
            genre_list.append(genre['name'])
    if len(book_lang)!=0:
        lang_genre_list=[book_lang+'***'+i for i in genre_list]

    return lang_genre_list
```

- Most number of books for one author

```
# this process will be on two Functions
# the first function is the most important as it will just use the writer
id
#the second function will get the the author id associated with the author
name

We ran the job using spark as follows
words=books_data.flatMap(lambda line:myfunction(line))
wordcount=words.map(lambda key:(key,1)).reduceByKey(lambda key,value:
key+value)
data2=authors_data.map(lambda line:get_authorName(ID))
data2.join(wordcount)
out=data2.sortBy(lambda x: x[1][1], ascending=False)
result = out.collect()

def get_authorID(book_metadata):
    """
    this function output every author assigned to every book in the
```

```
book_metadata goodreads_books.json.gz
    """
    authorsID_list=[]
    book_dict=json.loads(book_metadata)
    authors_list=book_dict['authors']
    if len(authors_list) != 0:
        for author in  authors_list:
            #it is a list of genres
            authorsID_list.append(author['author_id'])
    return authorsID_list
def get_authorName(author_data):
    author_dict=json.loads(author_data)
    authorsID_names_list=[]
    id=author_dict['author_id']
    name=author_dict['name']
    authorsID_names_list.append(id+'***',name)
    #return authorsID_names_list
    return (id,name)
```

- Most books have reviews

```
# this Process will be on two functions also
# the first function is the important but it gives the number of reviews
for a book id
# later we can transform the book id onto a book name

def get_bookID_reviews(review):
    """
    this function output every genre assigned to every book , using the
reviews file : goodread_reviews_dedup.json.gz
    """
    BookID_list=[]
    review_dict=json.loads(review)
    book_id=review_dict['book_id']
    if len(book_id) != 0:
        BookID_list.append(book_id)
    return BookID_list
# I found that i can get the reviews number from the book metadata:
def get_books_reviews(book_metadata):
    """
    this function output number of reviews for every book in the
book_metadata goodreads_books.json.gz
    """
    reviews_list=[]
    book_dict=json.loads(book_metadata)
    book_name=book_dict['title_without_series']
    reviews_num=book_dict['text_reviews_count']
    if len(reviews_num) != 0:
            reviews_list.append(book_name+'*-_-*'+reviews_num)
    return reviews_list
```

- Most readed books

```
# i also wrote another code if spark read csv like json (indexing with the column name not buy a number)
# this requirement will take 3 processess
# the first process is the important one as it will return the scv id of the book with the number of reads for it
# the second process should transform the csv id into regular book id then the third will transfrom the book id onto book name


def get_read_books_csvID(interaction):
    """
    this will get the readerd books only , it should be used with the interactions file: goodreads_interactions.csv
    """
    booksIDS_list=[]
    interactiondata=interaction.split(',')
    # is read is the third element
    is_read=int(int(interactiondata[2]))
    # book id is the second element
    book_id=interactiondata[1]
    if is_read ==1:
        booksIDS_list.append(book_id)
    return genre_list

def get_read_books_ID(interaction):
    """
    this will get the readerd books only , it should be used with the interactions file: goodreads_interactions.csv
    """
    booksIDS_list=[]
    interactiondata=json.loads(interaction)
    # is read is the third element
    is_read=interactiondata['is_read']
    # book id is the second element
    book_id=interactiondata['book_id']
    if is_read ==True:
        booksIDS_list.append(book_id)
    return genre_list
```

- Most “want to read” books

```
# this will be on one process only
# i will get the count of the 'to-read' shelf in popular shelves for every
book book_metadata
#this function should return
# remember this funciton will not use code for map:>>>>> words.map(lambda
key:(key,1))
# change map to code:>>>>>>>>     words.map(lambda
key:(key.split('*-_-*')[0],int(key.split('*-_-*')[1])))

def get_AllGenres(book_metadata):
    """
    this functoin gets the number of to-reads for every book in the
book_metadata goodreads_books.json.gz
    """
    books_with_toReadNum_list=[]
    book_dict=json.loads(book_metadata)
    book_name=book_dict['title_without_series']
    if len(book_dict['popular_shelves']) != 0:
        for genre in  book_dict['popular_shelves']:
            #it is a list of genres
            if genre['name']== 'to-read'

books_with_toReadNum_list.append(book_name+'*-_-*'+genre['count'])
    return books_with_toReadNum_list
```

- Most year have published books

```python
def get_year(book_metadata):
    """
    this function output every year assigned to every book in the
book_metadata goodreads_books.json.gz
    """
    year_list=[]
    book_dict=json.loads(book_metadata)
    year=book_dict['publication_year']
    if len(year) != 0:
        year_list.append(year)
    return year_list
```

- Most double genre for books

```python
#There are 2 function , the first to get a comination of every 2 genres for
every books
# the second function is not important
from itertools import combinations
def get_2genre_combination(book_fuzzy):
    """
    this function output every 2genre associated for every book in the
book_metadata gooreads_book_genres_initial.json.gz
    """
    genre_syn=[]
    genre_list=[]
    book_dict=json.loads(book_fuzzy)
    if len(book_dict['genres']) >=2 :
        for genre in  book_dict['genres']:
            #it is a list of genres
            genre_list.append(genre)
        #genre_list_mod=[i for i in genre_list if i not in avoid_list]
        genre_combination=combinations(genre_list,2)
        for combination in genre_combination:
            genre_syn.append('***'.join(list(sorted(combination))))

    return genre_syn
```

- Most Writer with genre

```python
# this funciton will use the book metadata and will get every author with every genre in the book
from itertools import product
def get_writerID_with_genre(book_metadata):
    """
    this function output every genre assigned to authors of every book in the book_metadata goodreads_books.json.gz
    """
    writer_genre_list=[]

#avoid_list=['to-read','owned','currently-reading','favorites','books-i-own','kindle','ebook','library','default','owned-books','to-buy','ebooks','wish-list','my-books','my-library','e-book','audiobook','books','i-own','audiobooks','audio','read-in-2016','favourites','e-books','read-in-2015','read-in-2017','own-it','maybe','read-in-2014','abandoned','did-not-finish'] # add others if found

lookup_list=['fiction','romance','adult','non-fiction','fantasy','mystery','novels','nonfiction','history','historical','young-adult','historical-fiction','adult-fiction','adventure','literature','thriller','classics','suspense','paranormal','science-fiction','crime','drama','sci-fi','humor','contemporary-romance','general-fiction','contemporary-fiction','family','school','mystery-thriller','childrens','short-stories','children','biography','classic','literary-fiction','sci-fi-fantasy','horror','supernatural','mysteries','magic','american','children-s','action','kids','mystery-suspense','british','netgalley','science','children-s-books','stand-alone','philosophy','erotica','teen','urban-fantasy','funny','religion','fantasy-sci-fi','realistic-fiction','literary','scifi','politics','memoir','love','war','childrens-books','paranormal-romance','thrillers','humour','europe','friendship','childhood','scifi-fantasy','comics','psychology','short-story','kids-books','women','detective','graphic-novels','action-adventure','crime-fiction','middle-grade']
    genre_list=[]
    authors_list=[]
    book_dict=json.loads(book_metadata)
    if len(book_dict['popular_shelves']) != 0:
        for genre in  book_dict['popular_shelves']:
            #it is a list of genres
            if genre['name'] in lookup_list:
                genre_list.append(genre['name'])
    if len(book_dict['authors'])!=0 and len(genre_list)!=0:
        for author in book_dict['authors']:
            authors_list.append(author['author_id'])
        writer_genre_list=list(product(authors_list,genre_list))

    return writer_genre_list
```

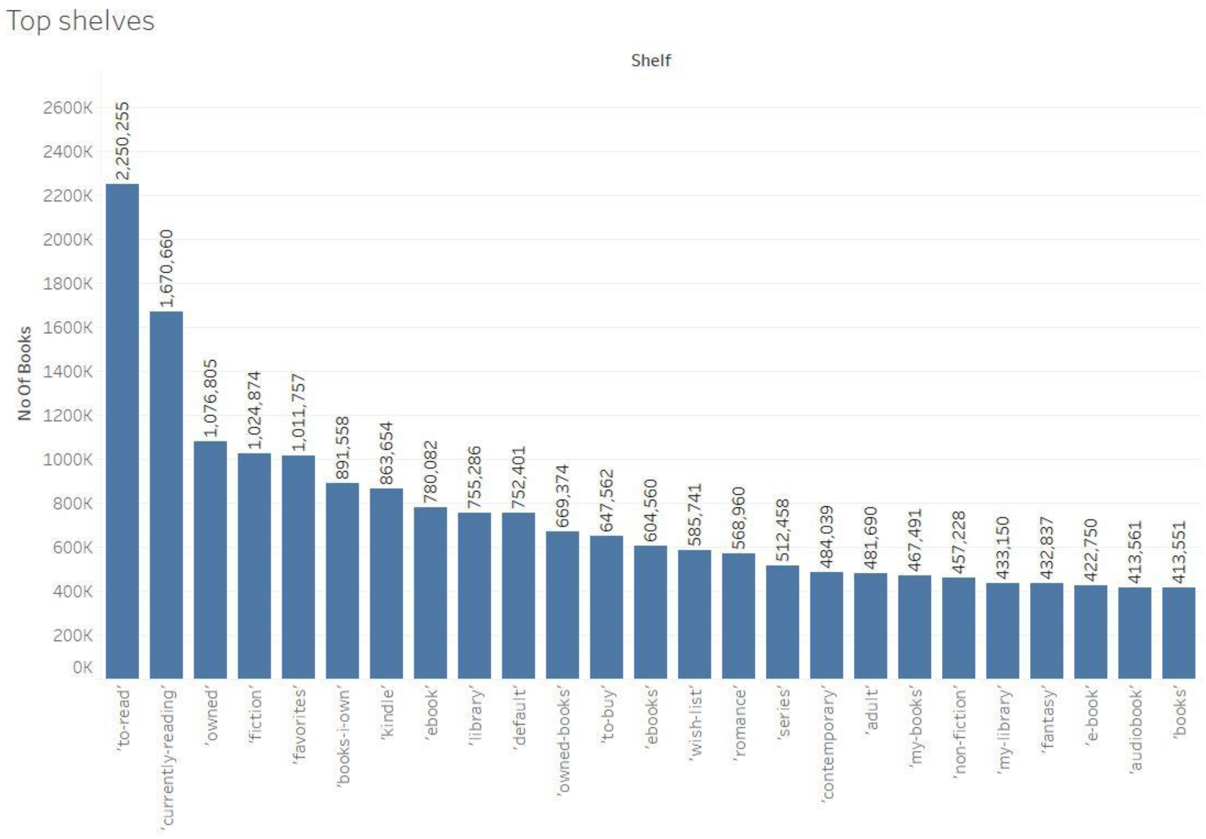
# RESULTS

The results from running spark was a text files

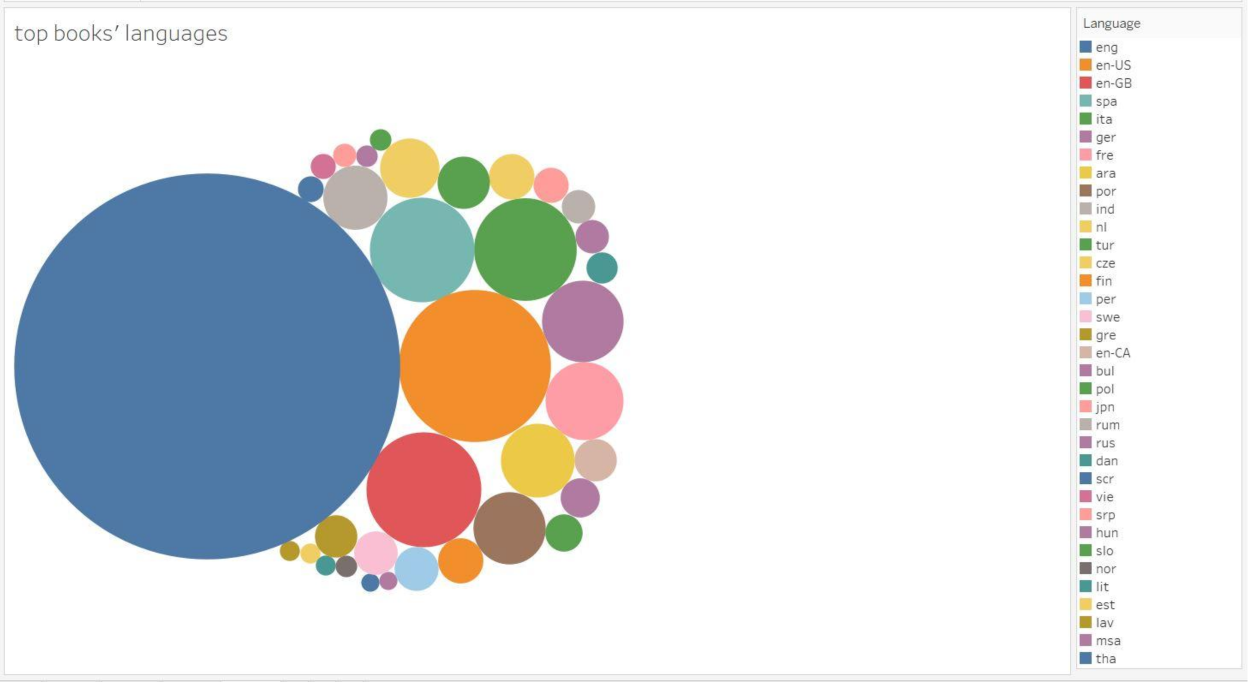
So we converted them to a more clean csv files for better options for plotting the results

The we used Tableau to make plots of the resulting analysis as follows:

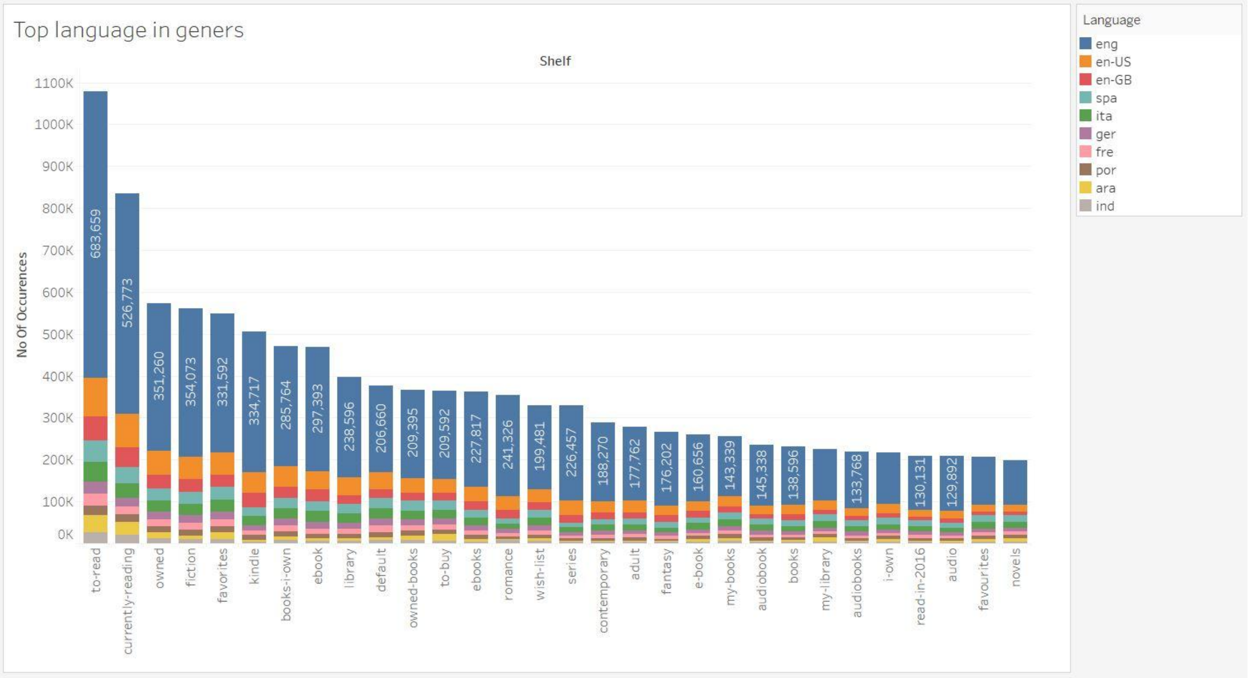
- Most shelves have books

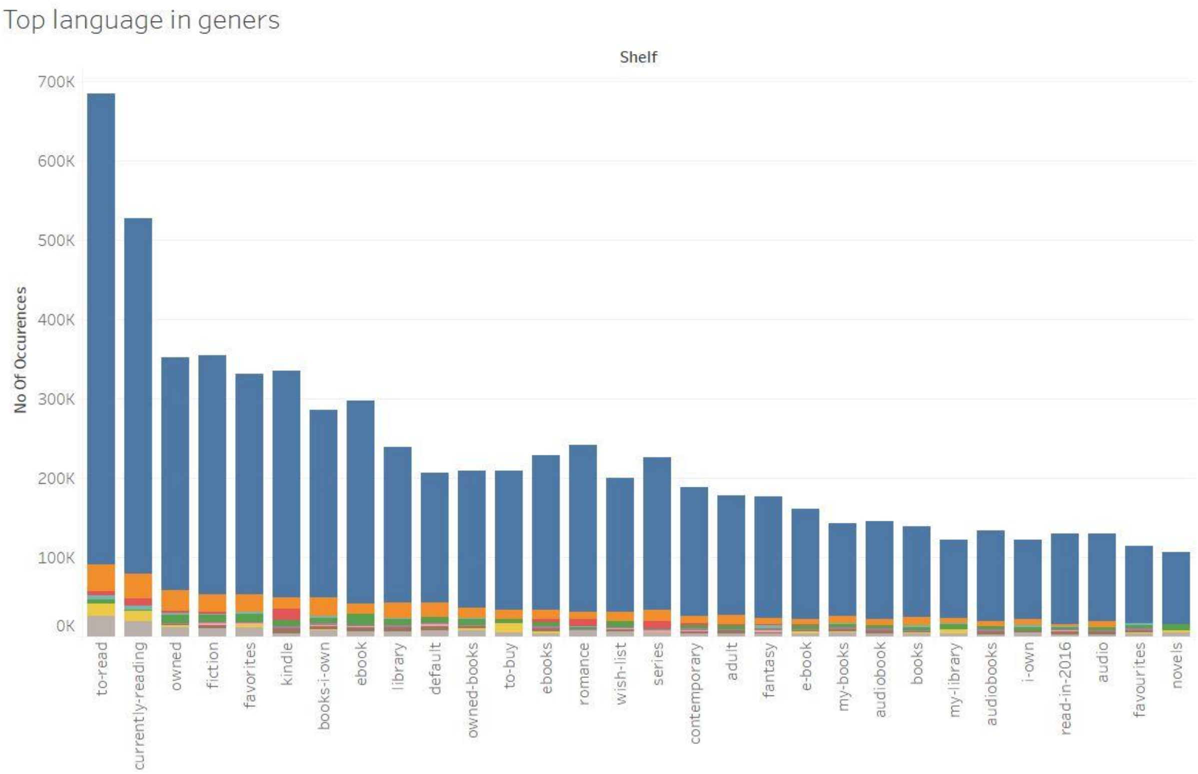


- Most language for Books

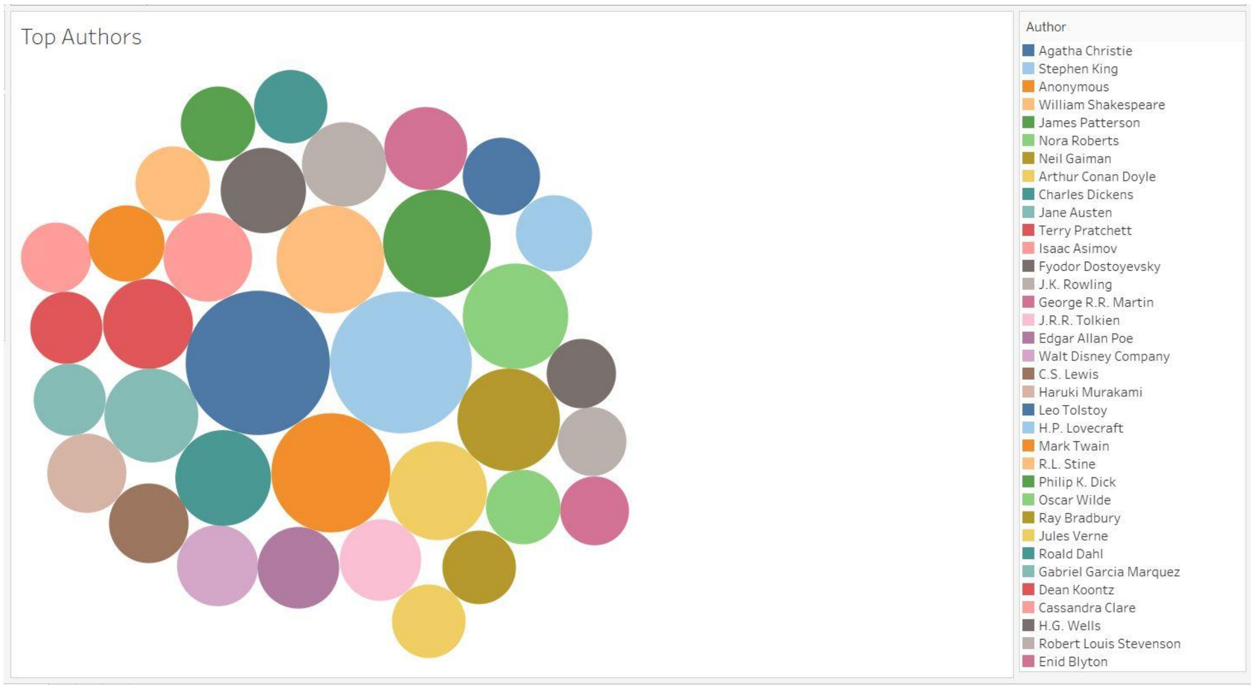


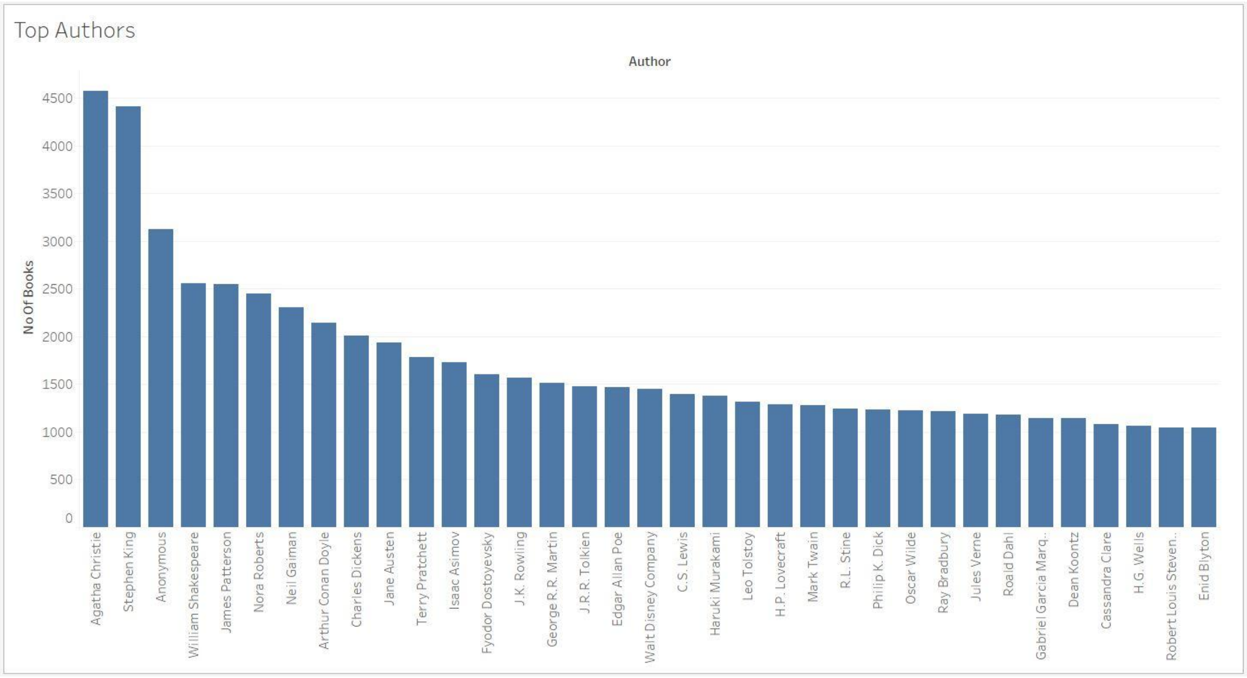
- Most language for every genre





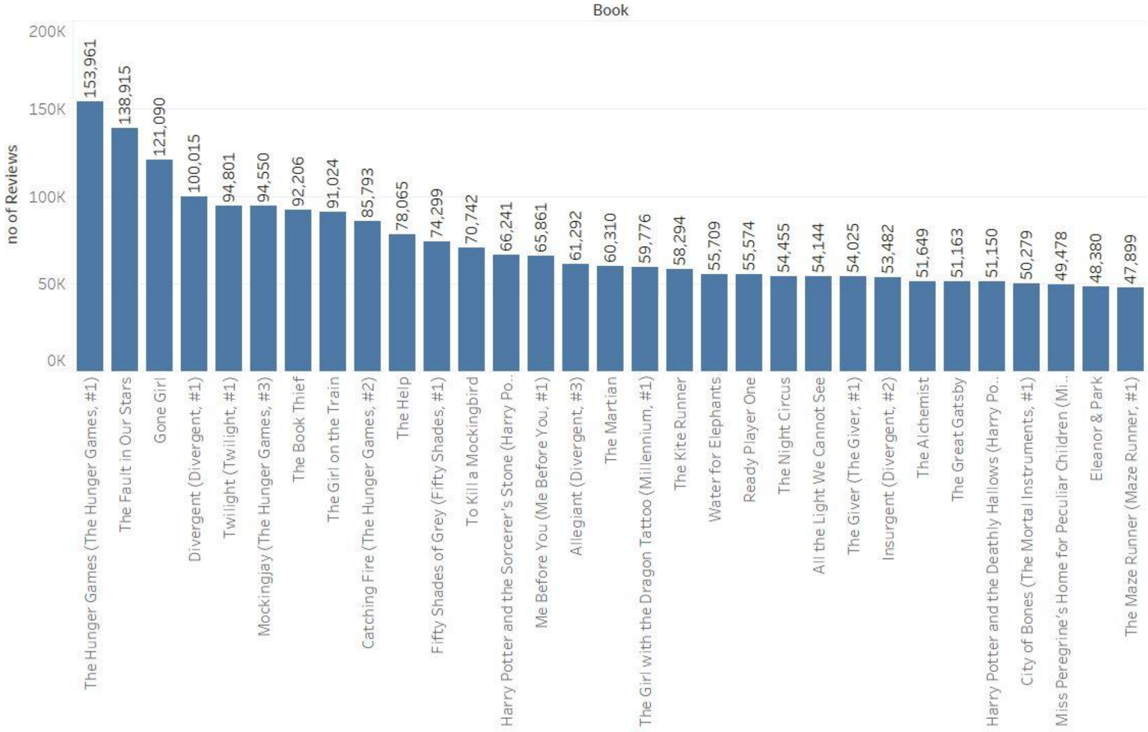
- Most number of books for one author



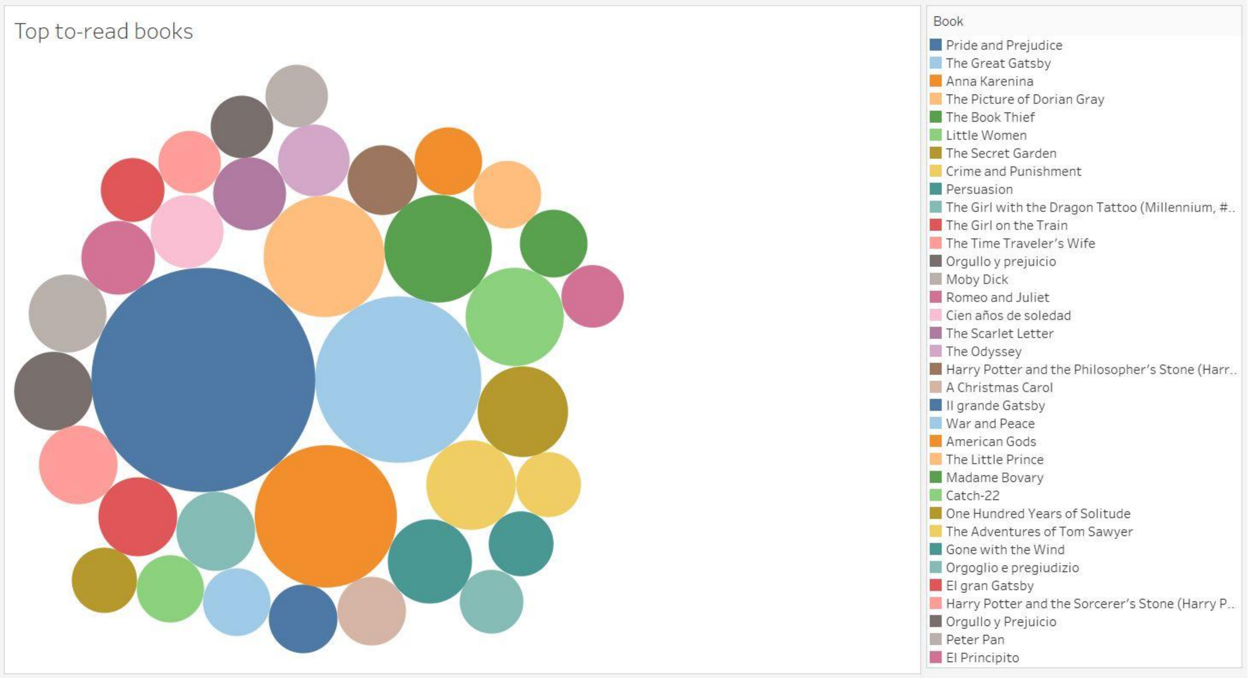


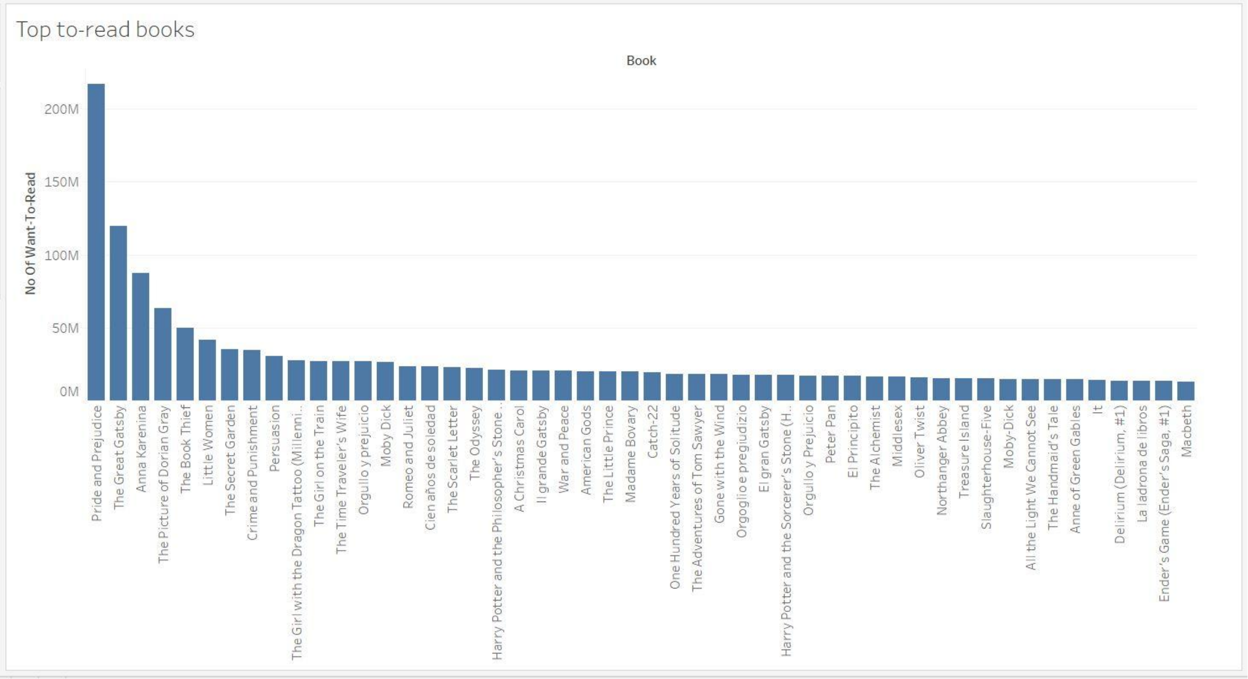
- Most books have reviews

Top books reviews

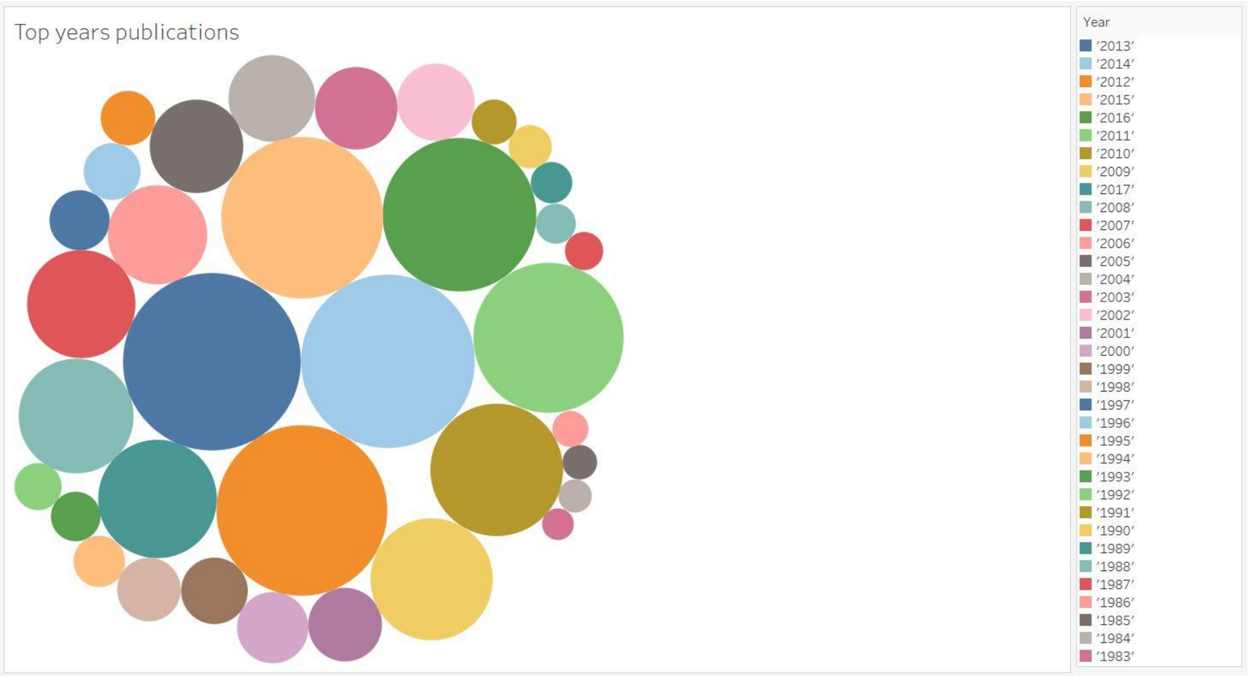


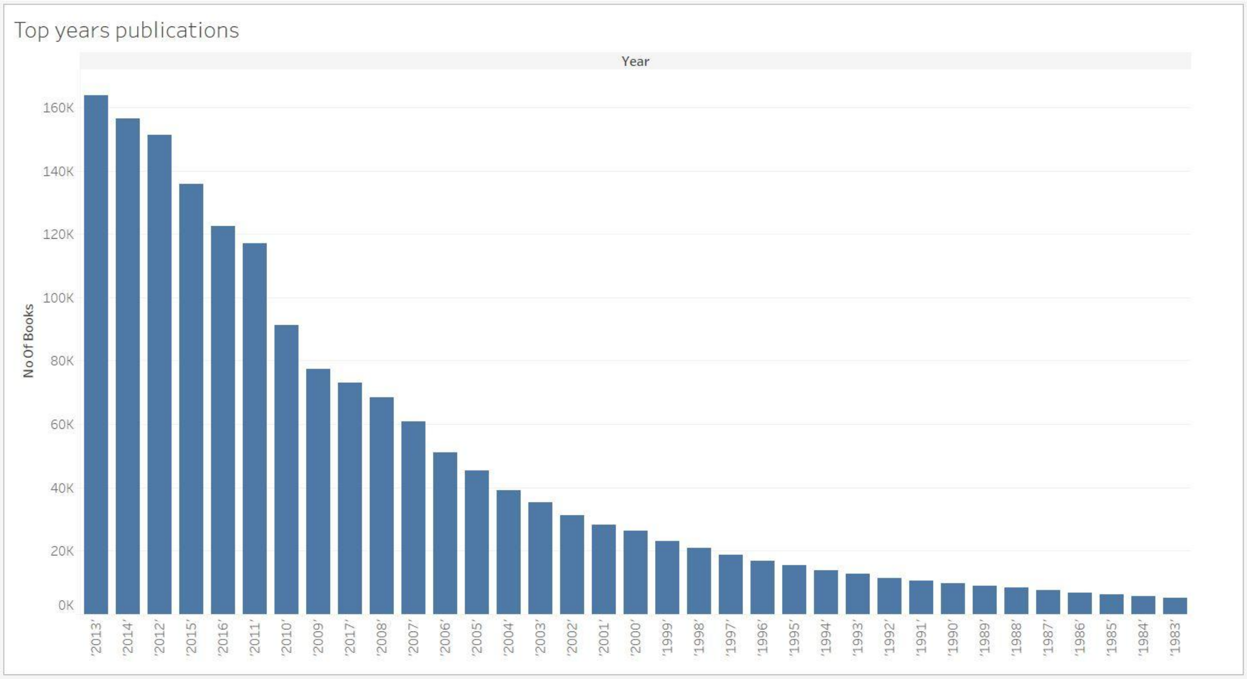
- Most “want to read” books



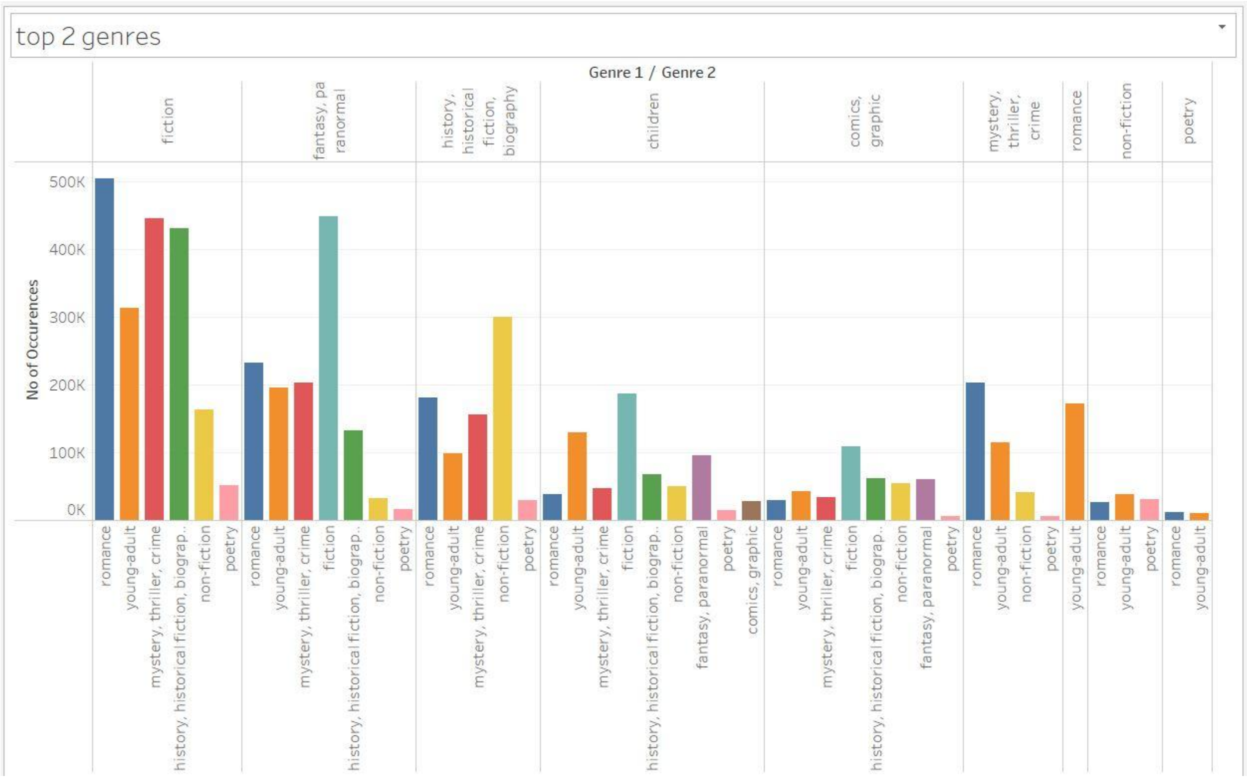


- Most year have published books

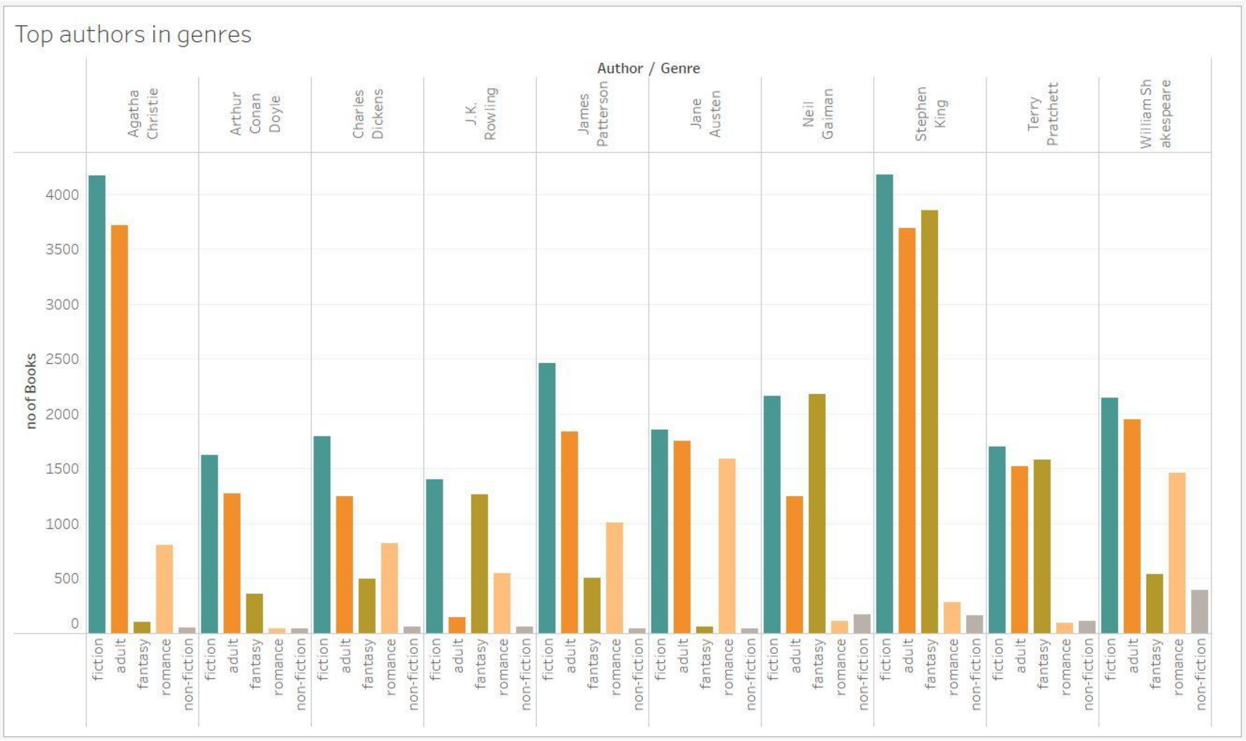




- Most double genre for books



- Most author with genre



## Machine Learning:

Collaborative filtering is the method utilized, which is often used in recommender systems.   Model-based collaborative filtering is now supported by spark.ml, in which users and products are defined by some variables that can be used to predict missing entries. The alternating least squares (ALS) approach is used by spark.ml to learn these latent components.

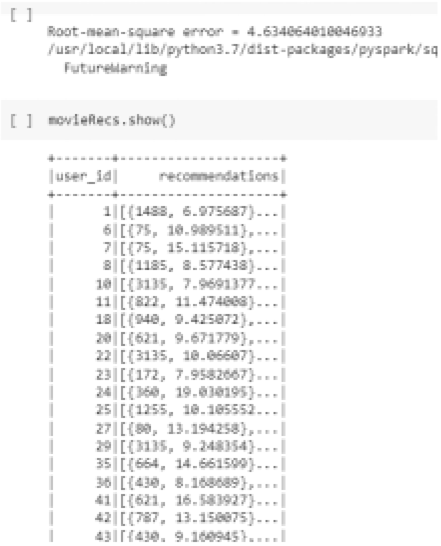
Steps:

```
·  Build the recommendation model using ALS on the training data
· Evaluate the model by computing the RMSE on the test data
```

Output:

```
·
```

```
 Generate top 10 movie recommendations for each user
· Generate top 10 user recommendations for each movie
```



```
[ ]
    Root-mean-square error = 4.634064010046933
    /usr/local/lib/python3.7/dist-packages/pyspark/sq
      FutureWarning
```

```
[ ]  movieRecs.show()
```

```
    +--------+--------------------+
    |user_id|     recommendations|
    +--------+--------------------+
    |       1|[{1488, 6.975687}...|
    |       6|[{75, 10.989511},...|
    |       7|[{75, 15.115718},...|
    |       8|[{1185, 8.577438}...|
    |      10|[{3135, 7.9691377...|
    |      11|[{822, 11.474008}...|
    |      18|[{940, 9.425072},...|
    |      20|[{621, 9.671779},...|
    |      22|[{3135, 10.06607}...|
    |      23|[{172, 7.9582667}...|
    |      24|[{360, 19.030195}...|
    |      25|[{1255, 10.105552...|
    |      27|[{80, 13.194258},...|
    |      29|[{3135, 9.248354}...|
    |      35|[{664, 14.661599}...|
    |      36|[{430, 8.168689},...|
    |      41|[{621, 16.583927}...|
    |      42|[{787, 13.150075}...|
    |      43|[{430, 9.160945},...|
```

- **DATASET PREPROCESSING**

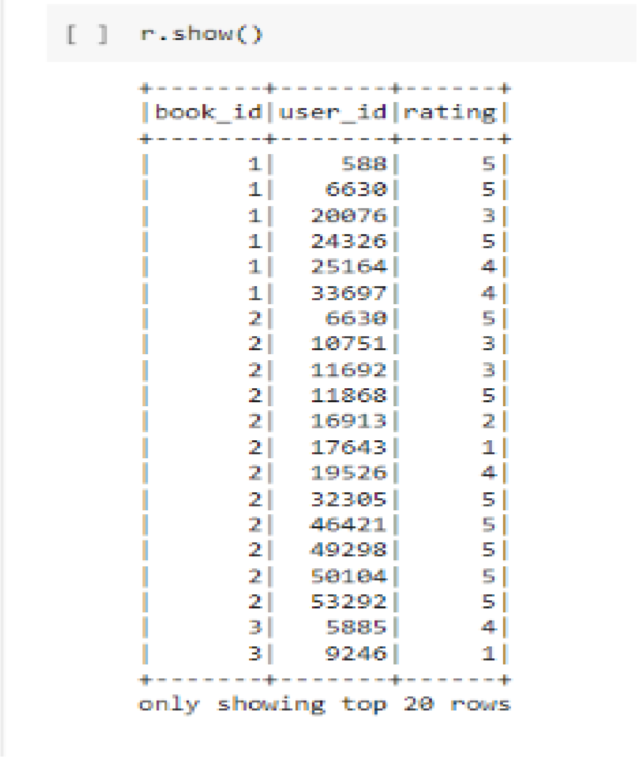
The only needed features from the whole dataset are the book_id, user_id, and rating but they needed to be processed:

1. They were read as a data frame.

2. They were strings and were casted as the input columns for the ALS must be

integers.

3. The user_id was combination of numbers and words so this column was mapped to be only integers.



```
[ ]  r.show()
```

```
+-------+-------+------+
|book_id|user_id|rating|
+-------+-------+------+
|      1|    588|     5|
|      1|   6630|     5|
|      1|  20076|     3|
|      1|  24326|     5|
|      1|  25164|     4|
|      1|  33697|     4|
|      2|   6630|     5|
|      2|  10751|     3|
|      2|  11692|     3|
|      2|  11868|     5|
|      2|  16913|     2|
|      2|  17643|     1|
|      2|  19526|     4|
|      2|  32305|     5|
|      2|  46421|     5|
|      2|  49298|     5|
|      2|  50104|     5|
|      2|  53292|     5|
|      3|   5885|     4|
|      3|   9246|     1|
+-------+-------+------+
only showing top 20 rows
```