Semester Project: Building Recommendation Engine on Books datasets

Name	Raja sekhar Pothina, Chandrashekar Akkenapally
Group	Group 1
Email	rp6kp@umsystem.edu, ca7kr@umsystem.edu
Course	CS 5402
Assignment	Semester Project
Date	29-Jul-2022
GitHub link	https://git-classes.mst.edu/rp6kp/sem_proj

Concept Description:

This assignment is about understanding and implement a simple recommendation system using K-Nearest Neighbors,Kmeans, 1R Classifier and Cosine Similarity.

Data Collection:

The data set was provided by our Instructor Mr.Perry Koob
The data given in the dataset has values that can help us to build a recommendation system that can suggest books for someone who has read Stardust by Neil Gaiman.

Example Description

goodreads_book_id
This attribute represent unique id of a book in goodreads website .This is a *Nominal* attribute type and Number data type

---

cover_link
This attribute represent cover link of a book in goodreads website .This is a *Nominal* attribute type and String data type

---

title
This attribute represent title of a book in goodreads website .This is a *Nominal* attribute type and String data type

---

series
This attribute represent series that a book belongs to in goodreads website .This is a *Nominal* attribute type and String data type

---

author
This attribute represent author of a book in goodreads website .This is a *Nominal* attribute type and String data type

---

rating_count
This attribute represent rating given for a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

review_count
This attribute represent reviews given for a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

average_rating
This attribute represent average rating of a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

five_star_ratings
This attribute represent the count of the five star rating given to a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

four_star_ratings
This attribute represent the count of four star rating given to a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

three_star_ratings
This attribute represent the count of three star rating given to a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

two_star_ratings
This attribute represent the count of two star rating given to a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

one_star_ratings
This attribute represent the count of one star rating given to a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

tag_id
This attribute represent id of a tag of a book in goodreads website .This is a *Nominal* attribute type and Number data type

---

tag_count
This attribute represent no of user given the same tag for a book in goodreads website .This is a *Ratio* attribute type and Number data type

---

---

tag_name

This attribute represent tag name given by the user for a book in goodreads website .This is a *Nominal* attribute type and String data type

---

title_with_series_name

This attribute is a derived to identify a book if any two books has same title but belongs to different series eg: books with title *Selected Poems*.This is a *Nominal* attribute type and String data type

---

Attributes level of measurement

Attributes	Level of Measurement
goodreads_book_id	Interval
cover_link	Nominal
title	Nominal
series	Nominal
author	Nominal
rating_count	Ratio
review_count	Ratio
average_rating	Ratio
five_star_ratings	Ratio
four_star_ratings	Ratio
three_star_ratings	Ratio
two_star_ratings	Ratio
one_star_ratings	Ratio
tag_id	Interval
tag_count	Ratio
tag_name	Nominal
title_with_series_name	Nominal

In [ ]:
```
# Imports

import matplotlib as mpl
import importlib
from sklearn.metrics import classification_report
import warnings
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler
from sklearn import neighbors
import seaborn as sns
import pandas as pd
import plotly.express as px
import numpy as np
```

```
import altair as alt
alt.data_transformers.enable('default', max_rows=None)
import matplotlib.pyplot as plt

importlib.reload(mpl)
importlib.reload(plt)
importlib.reload(sns)
params = {
    # "font.family":  "sans-serif",
    # "font.style":   "normal",
    # "font.variant": "normal",
    # "font.weight":  "bold",
    # "font.stretch": "normal",
    'legend.fontsize': 15,
    'axes.labelsize':15,
    'axes.titlesize': 15,
    'xtick.labelsize': 15,
    'ytick.labelsize': 15,
    "figure.titlesize": 20,
    "figure.figsize": (5.0,5.0),
    "figure.dpi":200
}
%matplotlib inline
plt.rcParams.update(params)

# mpl.rcParams.update(mpl.rcParamsDefault)

warnings.filterwarnings('ignore')
```

Data Import and Wrangling

In [ ]:
```
# load dataset
book_tags = pd.read_csv(str(f"../src-data/book_tags.csv"))
goodreads_books_cols = ['goodreads_book_id', 'cover_link', 'title', "series
                        "average_rating", "five_star_ratings", "four_star_r
goodreads_books = pd.read_csv(str(f"../src-data/goodreads_books.csv"),
                              usecols=goodreads_books_cols,encoding="utf-8"
tags = pd.read_csv(str(f"../src-data/tags.csv"))
```

In [ ]:
```
print(f"book_tags: {book_tags.shape}, goodreads_books: {goodreads_books.sha
```

```
book_tags: (9972, 3), goodreads_books: (52201, 13),tags: (83, 2)
```

In [ ]:
```
# merging all the dataframes to form the actual dataframe
books_tagnames = book_tags.merge(tags, on='tag_id')

goodreads_books_tags = goodreads_books.merge(
    books_tagnames, how="outer", on="goodreads_book_id")

goodreads_books_tags.count
# renaming the count column from tag data frame to tag_count for better und
goodreads_books_tags = goodreads_books_tags.rename(columns={"count": "tag_c
goodreads_books_tags.tag_count
goodreads_books_tags.dtypes
```

Out[ ]:
```
goodreads_book_id      object
title                  object
series                 object
cover_link             object
author                 object
rating_count          float64
review_count          float64
average_rating        float64
five_star_ratings     float64
four_star_ratings     float64
three_star_ratings    float64
two_star_ratings      float64
one_star_ratings      float64
tag_id                float64
tag_count             float64
tag_name               object
dtype: object
```

In [ ]:
```
goodreads_books_tags.head(10)
```

Out[ ]:

	goodreads_book_id	title	series	cover_link	author	ra
0	630104	Inner Circle	(Private #5)	https://i.gr-assets.com/images/S/compressed.ph...	Kate Brian, Julian Peploe	
1	9487	A Time to Embrace	(Timeless Love #2)	https://i.gr-assets.com/images/S/compressed.ph...	Karen Kingsbury	
2	6050894	Take Two	(Above the Line #2)	https://i.gr-assets.com/images/S/compressed.ph...	Karen Kingsbury	
3	39030	Reliquary	(Pendergast #2)	https://i.gr-assets.com/images/S/compressed.ph...	Douglas Preston, Lincoln Child	
4	998	The Millionaire Next Door: The Surprising Secr...	NaN	https://i.gr-assets.com/images/S/compressed.ph...	Thomas J. Stanley, William D. Danko	
5	311164	Black Sheep	NaN	https://i.gr-assets.com/images/S/compressed.ph...	Georgette Heyer	
6	32105	Sylvester	NaN	https://i.gr-assets.com/images/S/compressed.ph...	Georgette Heyer, Joan Wolf	
7	377993	Joe	NaN	https://i.gr-assets.com/images/S/compressed.ph...	Larry Brown	
8	71292	Asterix the Gaul	(AstÃ©rix #1)	https://i.gr-assets.com/images/S/compressed.ph...	RenÃ© Goscinny, Albert Uderzo, Anthea Bell, De...	
9	821003	When We Were Very Young	(Winnie-the-Pooh #3)	https://i.gr-assets.com/images/S/compressed.ph...	A.A. Milne, Ernest H. Shepard	

In [ ]:
```
# Understand the data tail() returns the last 10 rows of the dataset
goodreads_books_tags.tail(10)
```

Out[ ]:

	goodreads_book_id	title	series	cover_link	author	rating_count	review_count	averag
57621	25041504	NaN	NaN	NaN	NaN	NaN	NaN	
57622	25816688	NaN	NaN	NaN	NaN	NaN	NaN	
57623	25837341	NaN	NaN	NaN	NaN	NaN	NaN	
57624	26192646	NaN	NaN	NaN	NaN	NaN	NaN	
57625	29925715	NaN	NaN	NaN	NaN	NaN	NaN	
57626	11149	NaN	NaN	NaN	NaN	NaN	NaN	
57627	32067	NaN	NaN	NaN	NaN	NaN	NaN	
57628	5544	NaN	NaN	NaN	NaN	NaN	NaN	
57629	23093367	NaN	NaN	NaN	NaN	NaN	NaN	
57630	23093367	NaN	NaN	NaN	NaN	NaN	NaN	

In [ ]:
```
# for understanding the shape of dataset
# print("Shape: ", goodreads_books_tags.shape)
print(
    f"There are {goodreads_books_tags.shape[1]} columns in this dataset and
```

```
There are 16 columns in this dataset and 57631 rows in the dataset
```

In [ ]:
```
print(goodreads_books_tags.dtypes)
```

```
goodreads_book_id      object
title                  object
series                 object
cover_link             object
author                 object
rating_count          float64
review_count          float64
average_rating        float64
five_star_ratings     float64
four_star_ratings     float64
three_star_ratings    float64
two_star_ratings      float64
one_star_ratings      float64
tag_id                float64
tag_count             float64
tag_name               object
dtype: object
```

Handling Duplicate and Missing Values

Handling row having same title but different tag-names to have tag-names which has max count

In [ ]:
```
idx = book_tags.groupby(["goodreads_book_id"])["count"].transform(
    max) == book_tags['count']
book_tags = book_tags[idx]

books_tagnames = book_tags.merge(tags, on='tag_id')
```

```
print("books_tagname shape: ",books_tagnames.shape)
goodreads_books_tags = goodreads_books.merge(
    books_tagnames, how="outer", on="goodreads_book_id")

print("goodreads_books_tags: ",goodreads_books_tags.shape)
```

```
books_tagname shape:  (7048, 4)
goodreads_books_tags:  (54707, 16)
```

In [ ]:
```
columns = goodreads_books_tags.columns

def show_missing_info(columns):
    for a in columns:
        if goodreads_books_tags[a].isnull().any().sum():
            print(a, ' has: ', goodreads_books_tags[a].isnull(
            ).sum(), ' missing values')
        else:
            print(a,'has no missing values')

show_missing_info(columns)
```

```
goodreads_book_id has no missing values
title  has:  2508  missing values
series  has:  31388  missing values
cover_link  has:  3115  missing values
author  has:  2508  missing values
rating_count  has:  2508  missing values
review_count  has:  2508  missing values
average_rating  has:  2508  missing values
five_star_ratings  has:  2508  missing values
four_star_ratings  has:  2508  missing values
three_star_ratings  has:  2508  missing values
two_star_ratings  has:  2508  missing values
one_star_ratings  has:  2508  missing values
tag_id  has:  47659  missing values
count  has:  47659  missing values
tag_name  has:  47659  missing values
```

Books without title are removed from the dataframe by merging with left

In [ ]:
```
goodreads_books_tags = goodreads_books.merge(
    books_tagnames, how="left", on="goodreads_book_id")
goodreads_books_tags = goodreads_books_tags.loc[goodreads_books_tags['title
show_missing_info(columns)
```

```
goodreads_book_id has no missing values
title has no missing values
series  has:  28880  missing values
cover_link  has:  607  missing values
author has no missing values
rating_count has no missing values
review_count has no missing values
average_rating has no missing values
five_star_ratings has no missing values
four_star_ratings has no missing values
three_star_ratings has no missing values
two_star_ratings has no missing values
one_star_ratings has no missing values
tag_id  has:  47657  missing values
count  has:  47657  missing values
tag_name  has:  47657  missing values
```

For books without tag_id, count, tag_name giving the tag name "to-read" as default

In [ ]:
```
to_read_tag_row = tags[tags['tag_name'] == 'to-read'].to_dict()

goodreads_books_tags.loc[goodreads_books_tags['tag_id'].isna(
), "count"] = goodreads_books_tags['count'].isna().count()
goodreads_books_tags.loc[goodreads_books_tags['tag_id'].isna(
), "tag_name"] = to_read_tag_row['tag_name'][72]
goodreads_books_tags.loc[goodreads_books_tags['tag_id'].isna(
), 'tag_id'] = to_read_tag_row['tag_id'][72]

show_missing_info(columns)
```

```
goodreads_book_id has no missing values
title has no missing values
series  has:  28880  missing values
cover_link  has:  607  missing values
author has no missing values
rating_count has no missing values
review_count has no missing values
average_rating has no missing values
five_star_ratings has no missing values
four_star_ratings has no missing values
three_star_ratings has no missing values
two_star_ratings has no missing values
one_star_ratings has no missing values
tag_id has no missing values
count has no missing values
tag_name has no missing values
```

Initializing the Series, Coverlink columns with Defalt values

In [ ]:
```
goodreads_books_tags.loc[goodreads_books_tags['series'].isna(),'series'] =

goodreads_books_tags.loc[goodreads_books_tags['cover_link'].isna(
), 'cover_link'] = "https://upload.wikimedia.org/wikipedia/commons/6/65/No-

show_missing_info(columns)
```

```
goodreads_book_id has no missing values
title has no missing values
series has no missing values
cover_link has no missing values
author has no missing values
rating_count has no missing values
review_count has no missing values
average_rating has no missing values
five_star_ratings has no missing values
four_star_ratings has no missing values
three_star_ratings has no missing values
two_star_ratings has no missing values
one_star_ratings has no missing values
tag_id has no missing values
count has no missing values
tag_name has no missing values
```

Combining series,author name with title name to form a single column to recognise the each book differently

In [ ]:
```
goodreads_books_tags["title_with_series_name"] = goodreads_books_tags["titl
```

Exploratory Data Analysis

Statistics of Overall Data

In [ ]:
```
# Understanding the statistics of the object type attributes
goodreads_books_tags.describe(include='object',percentiles=[]).loc[['count'
```

Out[ ]:

	goodreads_book_id	title	series	cover_link	authc
count	52199	52199	52199	52199	5219
unique	52199	49685	22655	51593	2773
top	630104	Legacy		https://upload.wikimedia.org/wikipedia/commons...	No Rober
freq	1	14	28880	607	8

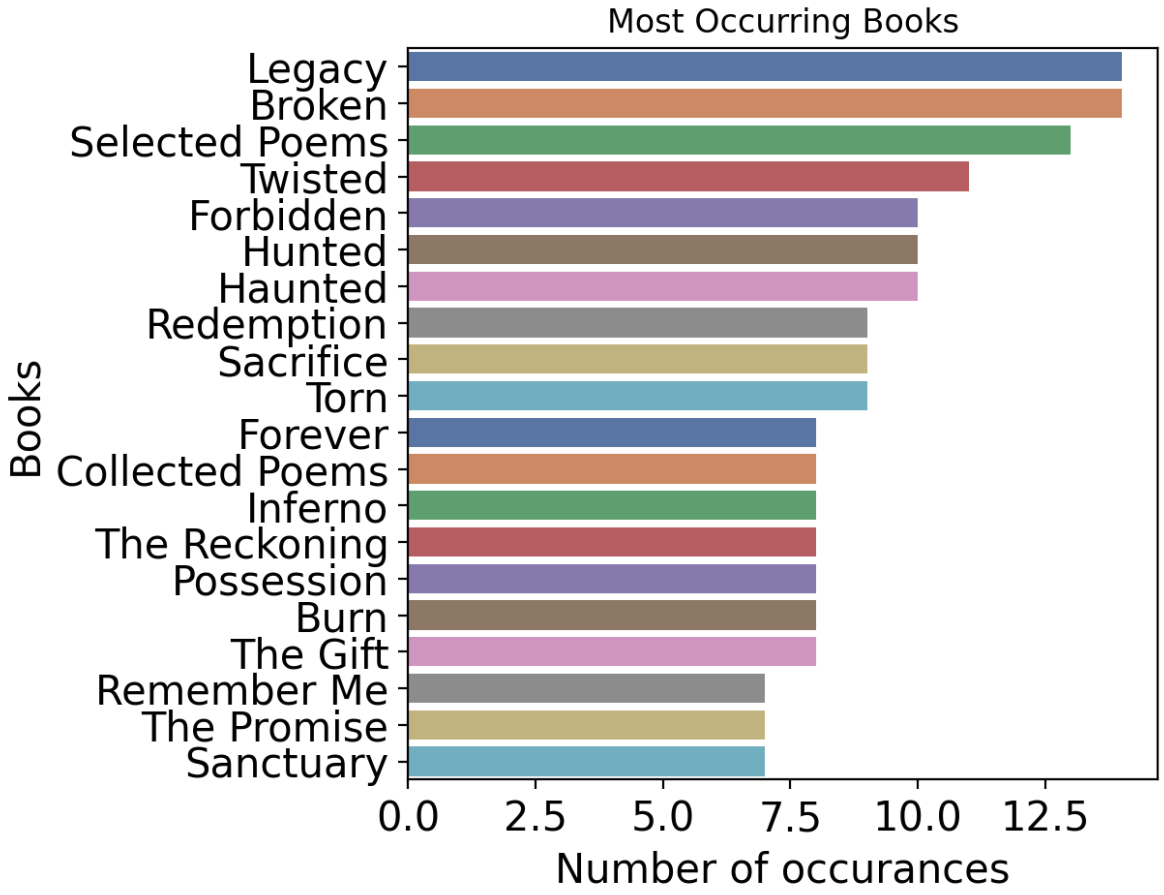
In [ ]:
```
# Understanding the statistics of the float,int type attributes
goodreads_books_tags.describe(include=['float64','int64'])
```

Out[ ]:

	rating_count	review_count	average_rating	five_star_ratings	four_star_ratings	three_s
count	5.219900e+04	52199.000000	52199.000000	5.219900e+04	5.219900e+04	52
mean	1.887361e+04	1012.980881	4.020610	7.817176e+03	6.250785e+03	3
std	1.163978e+05	4054.802421	0.367161	5.876373e+04	3.473533e+04	18
min	0.000000e+00	0.000000	0.000000	0.000000e+00	0.000000e+00	
25%	3.400000e+02	31.000000	3.820000	1.170000e+02	1.090000e+02	
50%	2.295000e+03	163.000000	4.030000	8.100000e+02	7.650000e+02	
75%	9.297500e+03	622.000000	4.230000	3.375500e+03	3.190500e+03	1
max	6.801077e+06	169511.000000	5.000000	4.414877e+06	1.868421e+06	980

Finding the most common books in the list

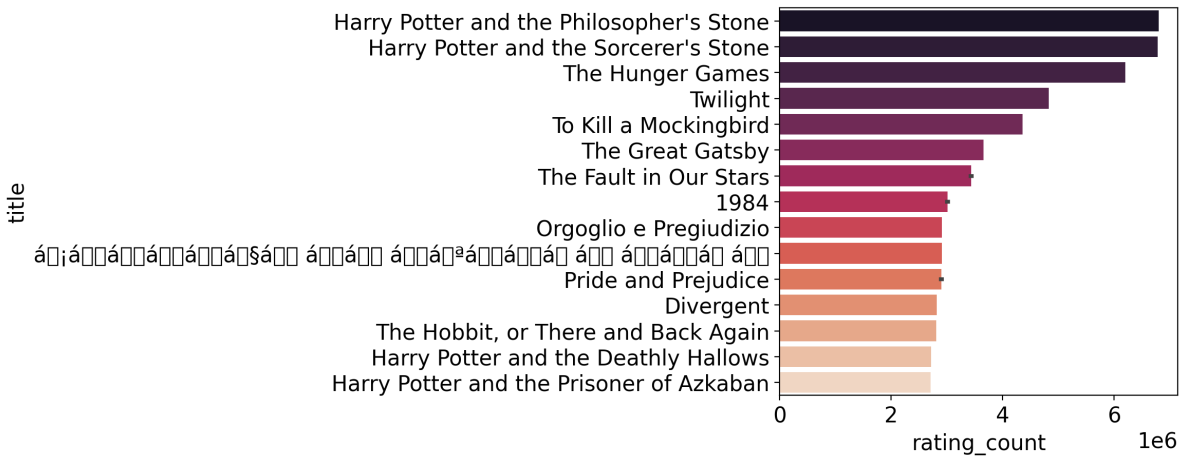
In [ ]:
```
#Taking the first 20:

# sns.set_context('poster')
books = goodreads_books_tags['title'].value_counts()[:20]
rating = goodreads_books_tags.average_rating[:20]
sns.barplot(x=books, y=books.index, palette='deep')
plt.title("Most Occurring Books")
plt.xlabel("Number of occurances")
plt.ylabel("Books")
plt.show()
```



Finding the top rated books in list

In [ ]:
```
most_rated_books = goodreads_books_tags.sort_values(
    "rating_count", ascending=False).head(20).set_index('title_with_series_
# print(most_rated_books['rating_count'])
sns.barplot(most_rated_books['rating_count'], most_rated_books.title, palet
```

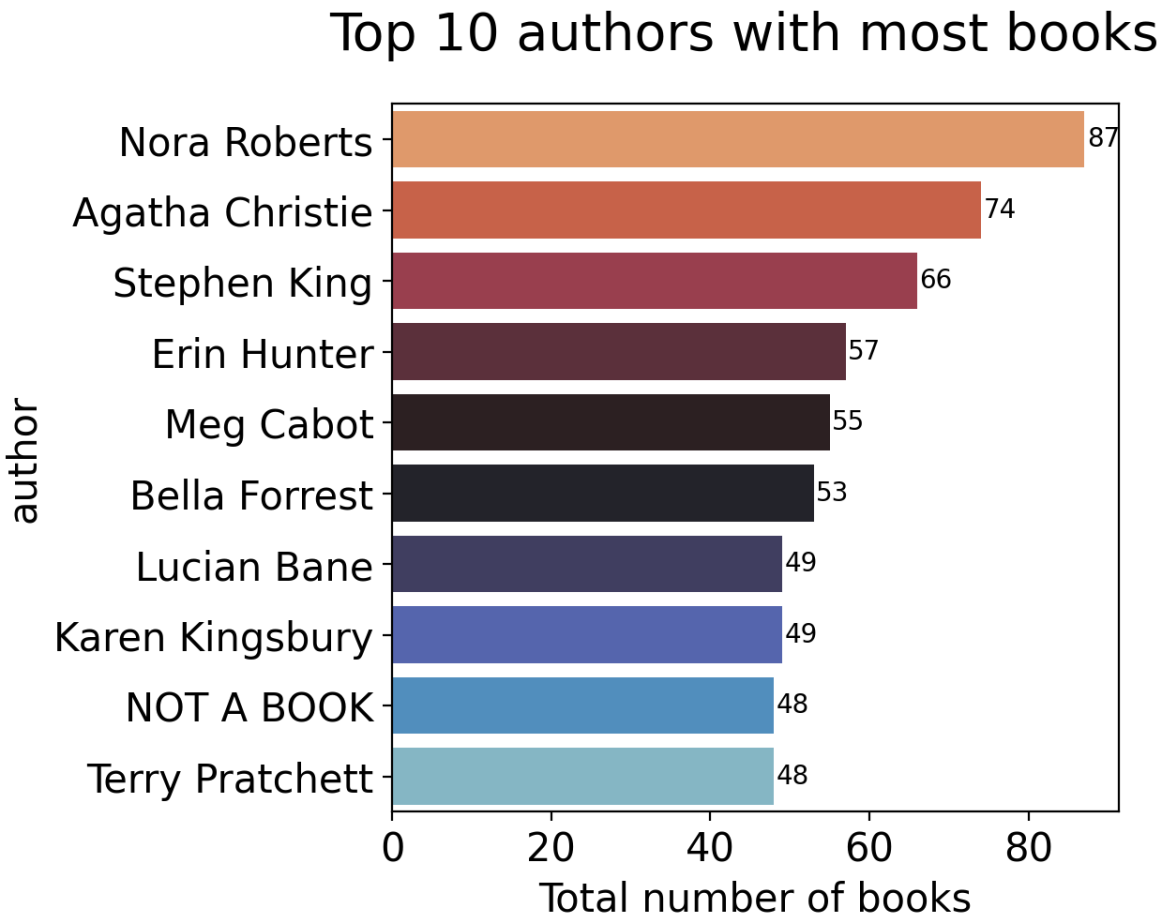
Out[ ]:
```
<AxesSubplot:xlabel='rating_count', ylabel='title'>
```



Finding the authors publishing the most books

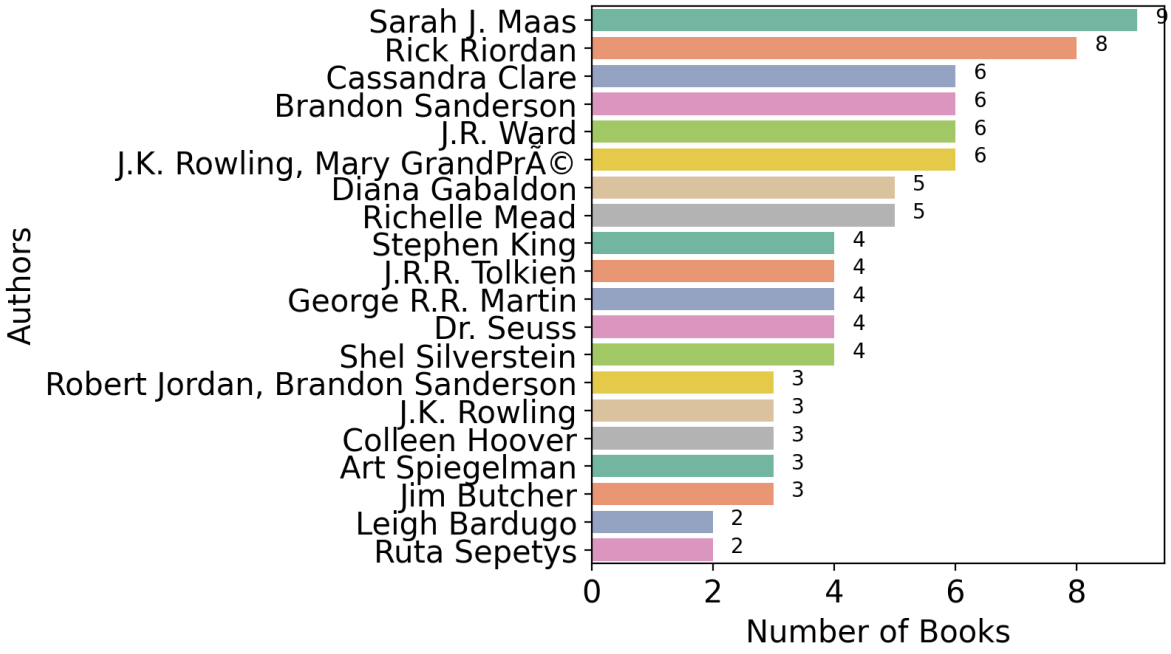
In [ ]:
```
# sns.set_context('talk')

most_books = goodreads_books_tags \
                .groupby('author')['title']\
                .count()\
                .reset_index()\
                .sort_values('title',ascending=False).head(10).set_index('a
```

```
ax = sns.barplot(most_books['title'],most_books.index, palette='icefire_r')
plt.suptitle("Top 10 authors with most books")
ax.set_xlabel("Total number of books")

for i in ax.patches:
    ax.text(i.get_width()+.3, i.get_y()+0.5, str(round(i.get_width())), col
```



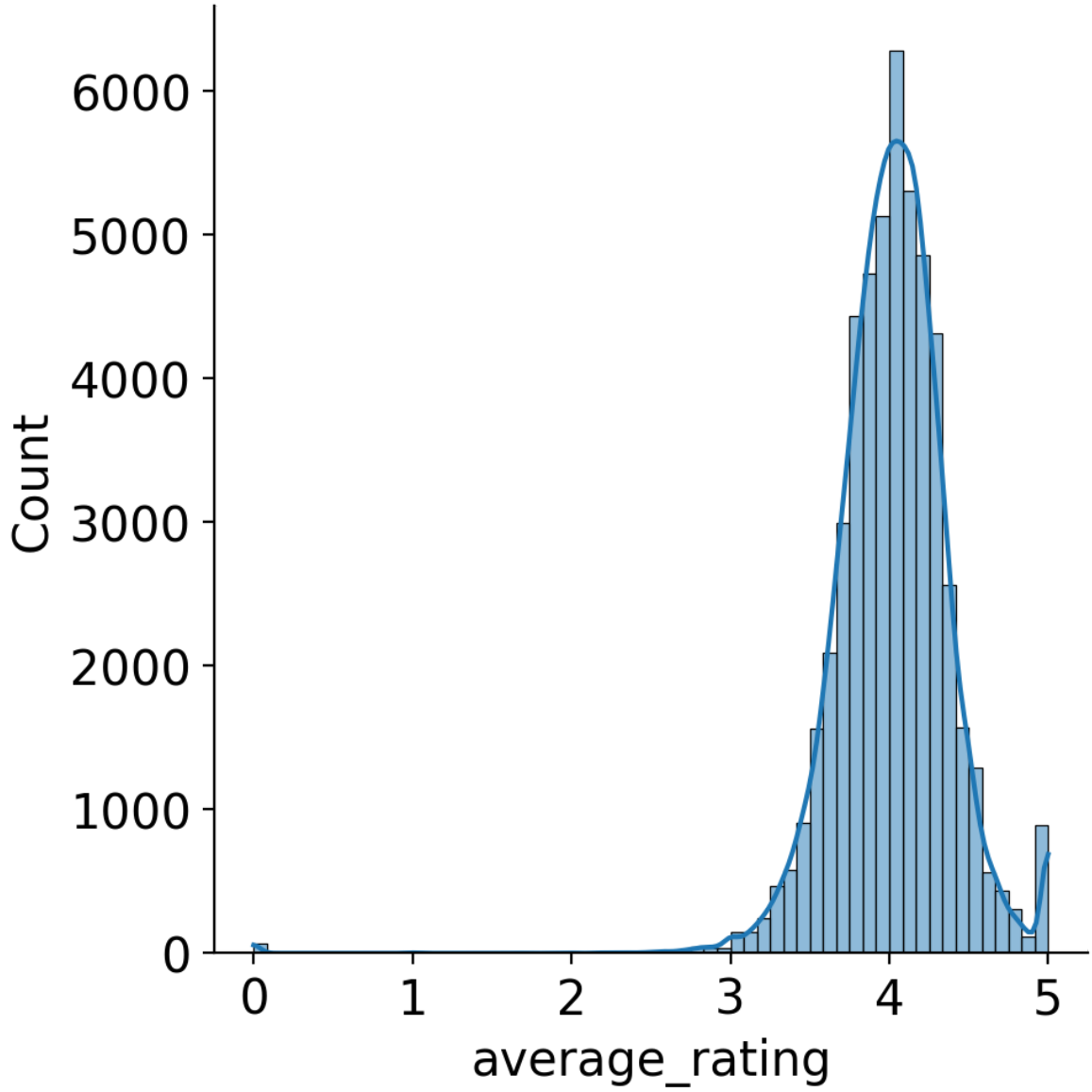
Finding Best Author who has written books having best average rating >=4.3 and rating count > 50000

In [ ]:
```
high_rated_author = goodreads_books_tags.loc[(goodreads_books_tags['rating_

high_rated_author = high_rated_author.groupby('author')['title'].count(
).reset_index().sort_values('title', ascending=False).head(20).set_index('a
ax = sns.barplot(high_rated_author['title'],
                 high_rated_author.index, palette='Set2')
ax.set_xlabel("Number of Books")
ax.set_ylabel("Authors")
for i in ax.patches:
    ax.text(i.get_width()+.3, i.get_y()+0.5,
            str(round(i.get_width())), color='k')
```



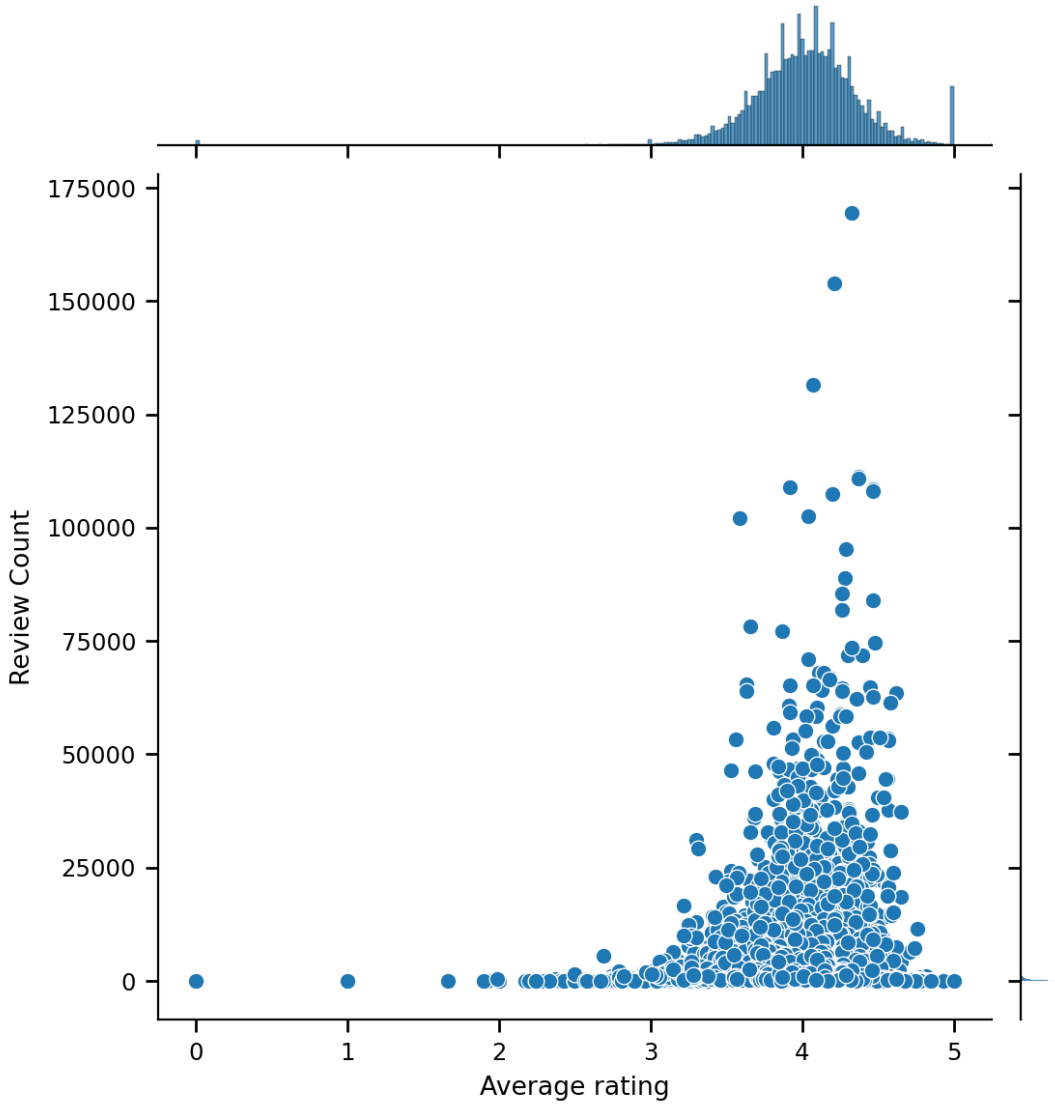
Finding the rating distribution for the books

In [ ]:
```
rating = goodreads_books_tags.average_rating.astype(float)
ax = sns.displot(data=goodreads_books_tags, x="average_rating",kde=True, bi
```



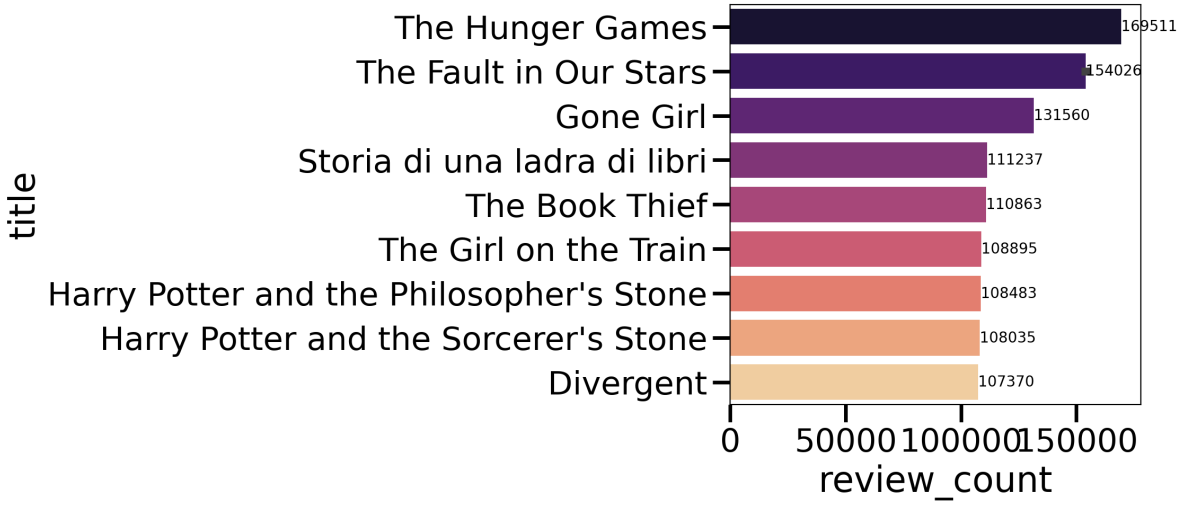
Checking for relation between average_rating and review counts

In [ ]:
```
sns.set_context('paper')
ax = sns.jointplot(x="average_rating", y="review_count",kind="scatter", dat
ax.set_axis_labels("Average rating","Review Count")
plt.show()
```



Books with most reviews

In [ ]:
```
most_reviews = goodreads_books_tags.sort_values("review_count",ascending=Fa

sns.set_context("poster")
ax = sns.barplot(most_reviews['review_count'],most_reviews.index, palette='
for i in ax.patches:
    ax.text(i.get_width()+2, i.get_y()+0.5,str(round(i.get_width())), color
plt.show()
```



Common Funtions Useful for Recommendation Systems

In [ ]:

```
from functools import reduce
from io import BytesIO
from PIL import Image
import requests


def get_index_from_name(dataframe,name):
    return dataframe[dataframe["title"] == name].index.tolist()[0]


def get_id_from_partial_name(dataframe, partial):
    all_books_names = list(dataframe.title.values)
    for name in all_books_names:
        if partial in name:
            print(name, all_books_names.index(name))


def print_format(dataframe, id):
    return ", ".join(reduce(
        lambda acc, prop:
        acc if dataframe.iloc[id][prop] == "" else acc + [str(f'{prop}: {da

def print_cover_image(url, title="no title"):
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    plt.figure(figsize=(20, 2))
    plt.grid(False)
    plt.axis('off')
    plt.title(title)
    print(plt.imshow(img))
```

Train and Test Split (Not Applicable)

Recommendation System With K Nearest Neighbours

Discritizing average_rating column

In [ ]:
```
goodreads_books_tags_2 = goodreads_books_tags.copy()

goodreads_books_tags_2.loc[(goodreads_books_tags_2['average_rating'] >= 0) &
        'rating_between'] = "between 0 and 1"
goodreads_books_tags_2.loc[(goodreads_books_tags_2['average_rating'] > 1) &
        'rating_between'] = "between 1 and 2"
goodreads_books_tags_2.loc[(goodreads_books_tags_2['average_rating'] > 2) &
        'rating_between'] = "between 2 and 3"
goodreads_books_tags_2.loc[(goodreads_books_tags_2['average_rating'] > 3) &
        'rating_between'] = "between 3 and 4"
goodreads_books_tags_2.loc[(goodreads_books_tags_2['average_rating'] > 4) &
        'rating_between'] = "between 4 and 5"


goodreads_books_tags_2.head()
```

Out[ ]:

	goodreads_book_id	title	series	cover_link	author	ra
0	630104	Inner Circle	(Private #5)	https://i.gr-assets.com/images/S/compressed.ph...	Kate Brian, Julian Peploe	
1	9487	A Time to Embrace	(Timeless Love #2)	https://i.gr-assets.com/images/S/compressed.ph...	Karen Kingsbury	
2	6050894	Take Two	(Above the Line #2)	https://i.gr-assets.com/images/S/compressed.ph...	Karen Kingsbury	
3	39030	Reliquary	(Pendergast #2)	https://i.gr-assets.com/images/S/compressed.ph...	Douglas Preston, Lincoln Child	
4	998	The Millionaire Next Door: The Surprising Secr...		https://i.gr-assets.com/images/S/compressed.ph...	Thomas J. Stanley, William D. Danko	

Converting the discretized average rating columns into features

In [ ]:
```
books_features = pd.concat([goodreads_books_tags_2['rating_between'].str.ge
    sep=","), goodreads_books_tags_2['average_rating'], goodreads_books_tag

books_features.head()
```

Out[ ]:

	between 0 and 1	between 1 and 2	between 2 and 3	between 3 and 4	between 4 and 5	average_rating	rating_count
0	0	0	0	0	1	4.03	7597.0
1	0	0	0	0	1	4.35	4179.0
2	0	0	0	0	1	4.23	6288.0
3	0	0	0	0	1	4.01	38382.0
4	0	0	0	0	1	4.04	72168.0

Scaling down the average rating columns features to a fixed range

In [ ]:
```
min_max_scaler = MinMaxScaler()
books_features = min_max_scaler.fit_transform(books_features)
np.round(books_features, 2)
```

Out[ ]:
```
array([[0.  , 0.  , 0.  , ..., 1.  , 0.81, 0.  ],
       [0.  , 0.  , 0.  , ..., 1.  , 0.87, 0.  ],
       [0.  , 0.  , 0.  , ..., 1.  , 0.85, 0.  ],
       ...,
       [0.  , 0.  , 0.  , ..., 1.  , 0.83, 0.  ],
       [0.  , 0.  , 0.  , ..., 1.  , 0.82, 0.  ],
       [0.  , 0.  , 0.  , ..., 0.  , 0.66, 0.  ]])
```

Applying K nearest neighbours algorithm where K = 6

In [ ]:
```
model = neighbors.NearestNeighbors(n_neighbors=6, algorithm='ball_tree')
model.fit(books_features)
distance, indices = model.kneighbors(books_features)
```

In [ ]:
```
def print_similar_books(query=None, id=None):
    if id:
        for id in indices[id][1:]:
            print_cover_image(goodreads_books_tags_2.iloc[id]['cover_link']
                goodreads_books_tags_2, id))
    if query:
        found_id = get_index_from_name(goodreads_books_tags_2,query)
        for id in indices[found_id][1:]:
            print_cover_image(goodreads_books_tags_2.iloc[id]['cover_link']
                goodreads_books_tags_2, id))
```

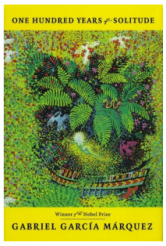
Evaluating Books Recommended for Stardust by Neil Gailman

In [ ]:
```
print_similar_books("Stardust")
```

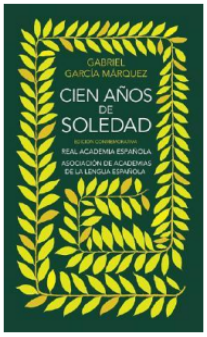
```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: Room, author: Emma Donoghue

title: One Hundred Years of Solitude, author: Gabriel GarcÃ-a MÃ¡rquez, Gregory Rabassa



title: Cien aÃ±os de soledad, author: Gabriel GarcÃ-a MÃ¡rquez



title: Cem Anos de SolidÃ£o, author: Gabriel GarcÃ-a MÃ¡rquez, Margarida Santiago

title: YÃ¼zyÄ±llÄ±k YalnÄ±zlÄ±k, author: Gabriel GarcÃ-a MÃ¡rquez, SeÃ§kin Selvi

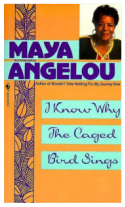
Evaluating Books Recommended for Book titled *Room*

In [ ]:
```
print_similar_books("Room")
```

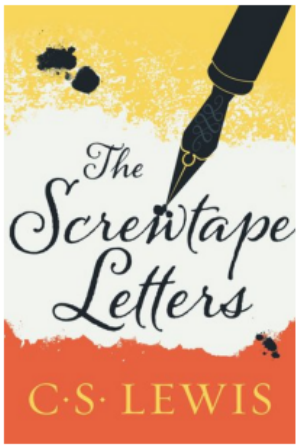
```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: Twilight: The Complete Illustrated Movie Companion, series: (The Twilight Saga: The Official Illustrated Movie Companion #1), author: Mark Cotta Vaz

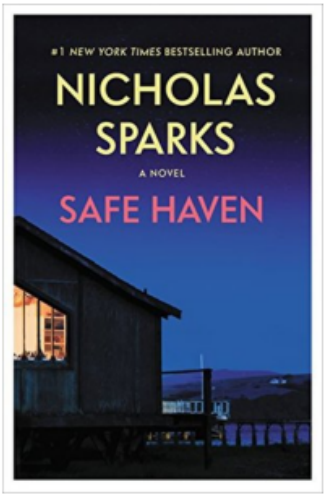
title: I Know Why the Caged Bird Sings, series: (Maya Angelou's Autobiography #1), author: Maya Angelou



title: The Screwtape Letters, author: C.S. Lewis



title: Safe Haven, author: Nicholas Sparks



title: The One, series: (The Selection #3), author: Kiera Cass

Recommendation System with Kmeans Classifier

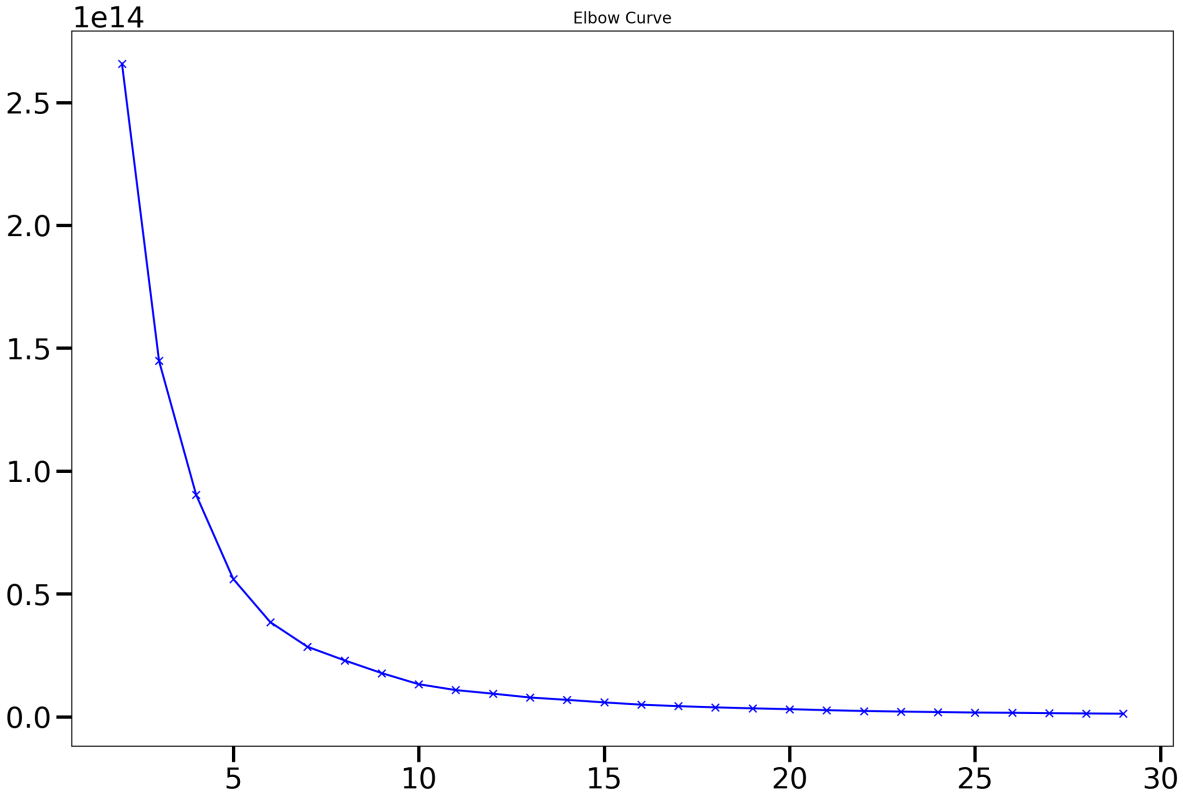
In [ ]:
```
trial = goodreads_books_tags_2[['average_rating', 'rating_count']]
```

```
data = np.asarray([np.asarray(trial['average_rating']),
                   np.asarray(trial['rating_count'])]).T
```

Finding the optimal cluster count

In [ ]:
```
from sklearn.cluster import KMeans


X = data
distortions = []
for k in range(2, 30):
    k_means = KMeans(n_clusters=k)
    k_means.fit(X)
    distortions.append(k_means.inertia_)

fig = plt.figure(figsize=(15, 10))
plt.plot(range(2, 30), distortions, 'bx-')
plt.title("Elbow Curve")
```

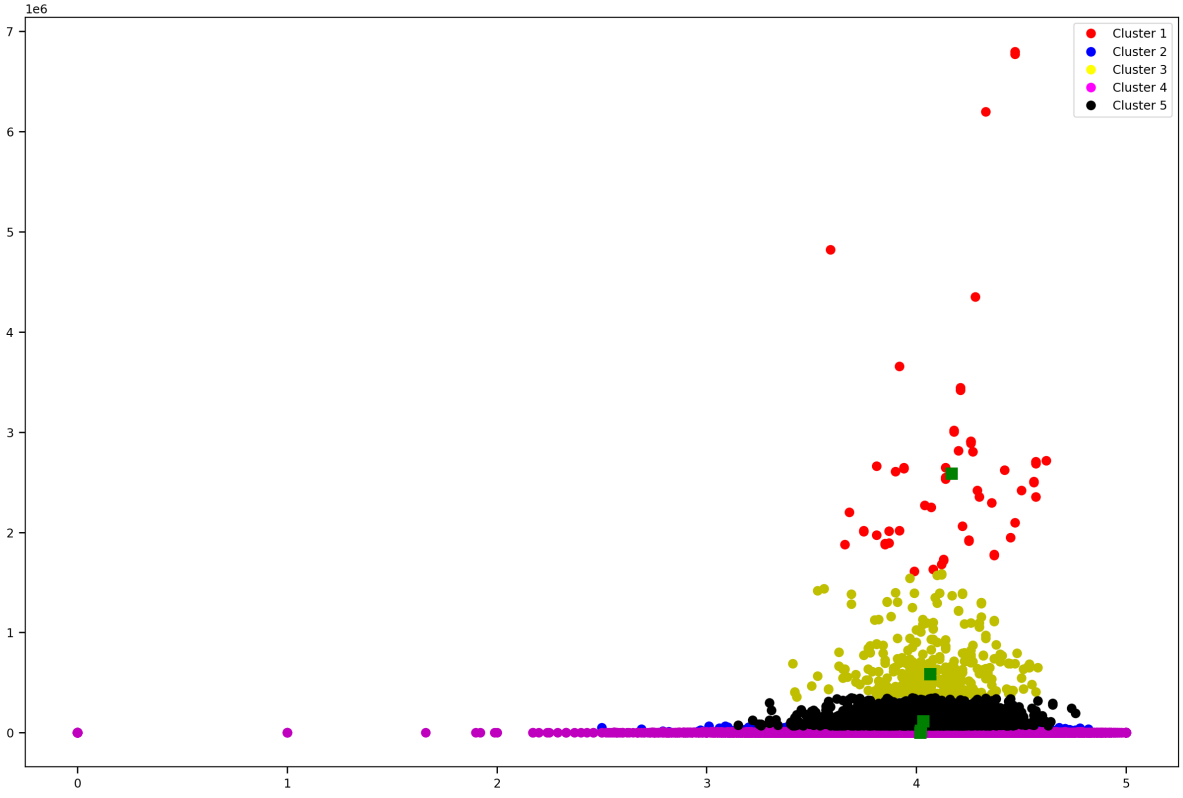
Out[ ]:
```
Text(0.5, 1.0, 'Elbow Curve')
```



In [ ]:
```
from scipy.cluster.vq import kmeans, vq

#Computing K means with K = 5, thus, taking it as 5 clusters
centroids, _ = kmeans(data, 5)

#assigning each sample to a cluster
#Vector Quantisation:

idx, _ = vq(data, centroids)
```

In [ ]:
```
from matplotlib.lines import Line2D
# some plotting using numpy's logical indexing
sns.set_context('paper')
plt.figure(figsize=(15,10))
plt.plot(
     data[idx==0,0],data[idx==0,1],'or',#red circles
```

```
    data[idx==1,0],data[idx==1,1],'ob',#blue circles
    data[idx==2,0],data[idx==2,1],'oy', #yellow circles
    data[idx==3,0],data[idx==3,1],'om', #magenta circles
    data[idx==4,0],data[idx==4,1],'ok',#black circles
)
plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8, )

circle1 = Line2D(range(1), range(1), color = 'red', linewidth = 0, marker=
circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker=
circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0,  marker=
circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker=
circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker=

plt.legend((circle1, circle2, circle3, circle4, circle5)
           , ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5


plt.show()
```



Finding the cluster of the Stardust book

In [ ]:
```
selected_book_1 = goodreads_books_tags[goodreads_books_tags["title"] == "St

# to print the table vertically
selected_book_1.T

selected_book_1_id = selected_book_1.index[0]

print(f"The Stardust book belongs to cluster {idx[selected_book_1.index][0]
```

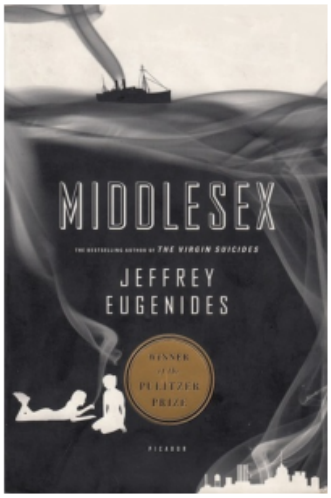
```
The Stardust book belongs to cluster 2 and its index in dataframe is 41860
```

In [ ]:
```
# idx[idx==idx[41860]]
indexes_to_keep = np.where(idx == idx[selected_book_1_id])
four_randomly_selected_recommendations = indexes_to_keep[0][np.random.choic
    np.size(indexes_to_keep[0]), size=5, replace=False)]

for id in four_randomly_selected_recommendations:
```

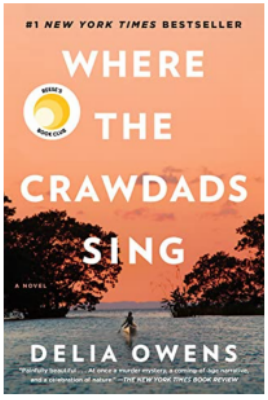
```
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: Middlesex, author: Jeffrey Eugenides

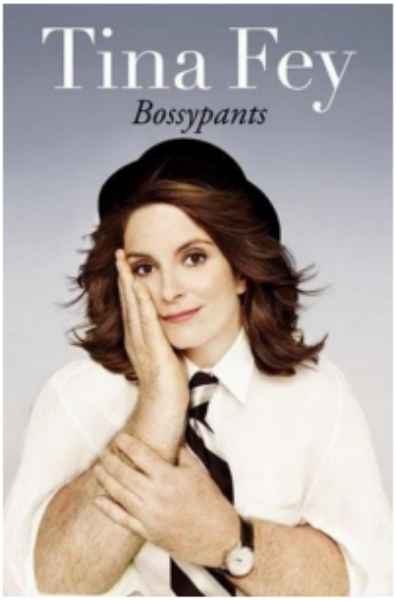


title: Where the Crawdads Sing, author: Delia Owens



title: The Little Prince, author: Antoine de Saint-ExupÃ©ry, Richard Howard, Ivan Minatti, Nguyá»n ThÃ nh VÅ©, Janez GradiÅ¡nik

title: Bossypants, author: Tina Fey



title: Into Thin Air: A Personal Account of the Mount Everest Disaster, author: Jon Krakauer

Finding the cluster of the Room book

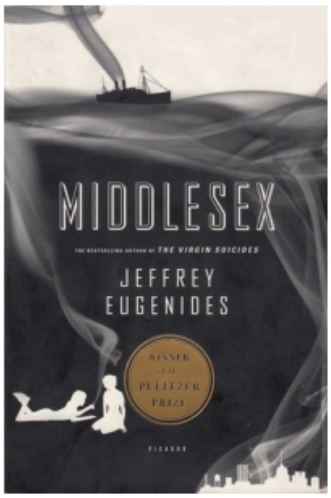
In [ ]:
```
selected_book_2 = goodreads_books_tags[goodreads_books_tags["title"] == "Ro

# to print the table vertically
selected_book_2.T

selected_book_2_id = selected_book_2.index[0]
print(
    f"The Room book belongs to cluster {idx[selected_book_2.index][0]} and
```

```
The Room book belongs to cluster 2 and its index in dataframe is 41948
```

In [ ]:
```
# idx[idx==idx[41860]]
indexes_to_keep = np.where(idx == idx[selected_book_2_id])
four_randomly_selected_recommendations = indexes_to_keep[0][np.random.choic
    np.size(indexes_to_keep[0]), size=5, replace=False)]

for id in four_randomly_selected_recommendations:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

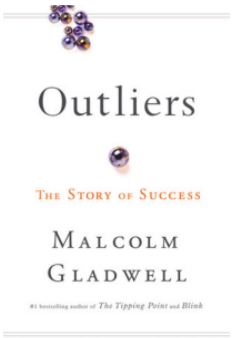
title: Middlesex, author: Jeffrey Eugenides



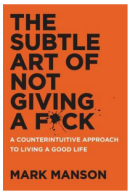
title: Lo strano caso del dottor Jekyll e del signor Hyde, author: Robert Louis Stevenson, Sandro Veronesi, Oreste del Buono

title: Paper Towns, author: John Green

title: Outliers: The Story of Success, author: Malcolm Gladwell



title: The Subtle Art of Not Giving a F*ck: A Counterintuitive Approach to Living a Good Life, author: Mark Manson



Recommendation System with 1R Classifier

Viewing the values of the Stardust Book

In [ ]:
```
selected_book = goodreads_books_tags[goodreads_books_tags["title"] == "Star

# to print the table vertically
selected_book.T
```

Out[ ]:

	41860
goodreads_book_id	16793
title	Stardust
series	
cover_link	https://i.gr-assets.com/images/S/compressed.ph...
author	Neil Gaiman
rating_count	346051.0
review_count	17981.0
average_rating	4.09
five_star_ratings	130605.0
four_star_ratings	135308.0
three_star_ratings	63731.0
two_star_ratings	12469.0
one_star_ratings	3938.0
tag_id	30574.0
count	177061.0
tag_name	to-read
title_with_series_name	Stardust, , Neil Gaiman

Recommending based on average rating greater than or equal to the book selected

In [ ]:
```
def get_books_average_rating_greater_than(selected_book_name):
    selected_book_row = goodreads_books_tags[goodreads_books_tags["title"]
                                             == selected_book_name ]
    return goodreads_books_tags[(goodreads_books_tags["average_rating"]
                                 >= selected_book_row.average_rating.values[
                                 != selected_book_row.title.values[0])]


recommended_books_stardust = get_books_average_rating_greater_than(
    "Stardust").sample(5)
```

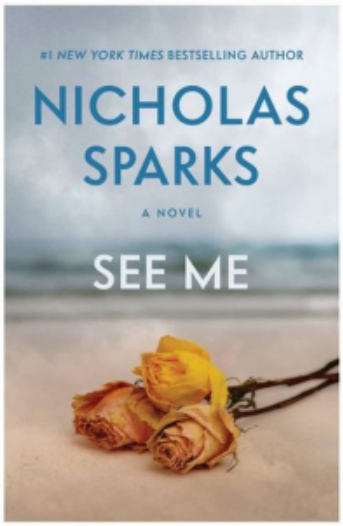
```
recommended_books_room = get_books_average_rating_greater_than("Room").samp
```

Stardust Recommendation

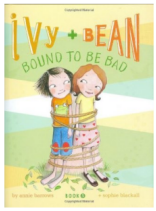
In [ ]:
```
for id in recommended_books_stardust.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: See Me, author: Nicholas Sparks

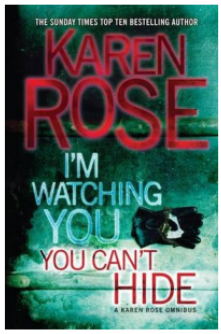


title: Ivy and Bean: Bound to be Bad, series: (Ivy & Bean #5), author: Annie Barrows, Sophie Blackall

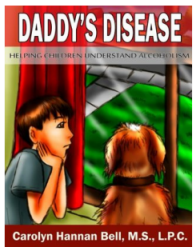


title: Sweet Danger, series: (A Mermaid Isle Romance #3), author: Cali MacKay

title: I'm Watching You / You Can't Hide, author: Karen Rose



title: Daddy's Disease: Helping Children Understand Alcoholism, author: Carolyn Bell

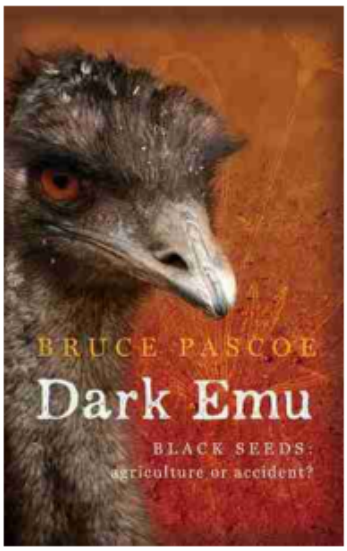


Room Book Recommendation

In [ ]:
```
for id in recommended_books_room.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

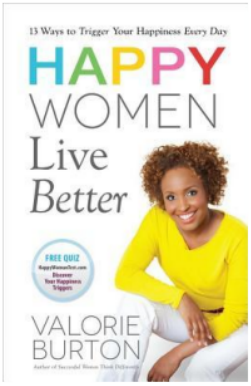
```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: Dark Emu, author: Bruce Pascoe



title: A Court of Mist and Fury, series: (A Court of Thorns and Roses #2), author: Sarah J. Maas

title: Happy Women Live Better, author: Valorie Burton



title: Sweet Summer Love, series: (Courting Love (The Sweetest Thing) #3), author: Sierra Hill

title: The Changeling Sea, author: Patricia A. McKillip

Recommending Books based on author of the book selected

In [ ]:
```
def get_books_with_same_author(selected_book_name):
    selected_book_row = goodreads_books_tags[goodreads_books_tags["title"]
                                             == selected_book_name]
    return goodreads_books_tags[(goodreads_books_tags["author"]
                                 == selected_book_row.author.values[0]) & (g
                                 != selected_book_row.title.values[0])]


recommended_books_stardust = get_books_with_same_author("Stardust").sample(
recommended_books_room = get_books_with_same_author("Room").sample()
```

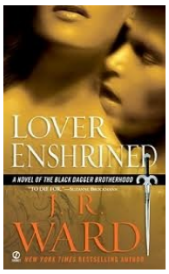
```
# print("Star dust recommendations: ")
# for id in recommended_books_stardust.index:
#     print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_
#         goodreads_books_tags, id))
```

Stardust Recommendation

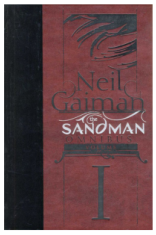
In [ ]:
```
for id in recommended_books_stardust.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

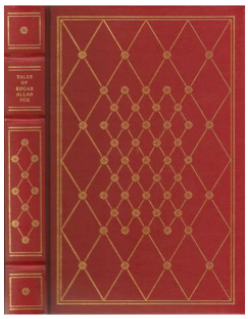
title: Lover Enshrined, series: (Black Dagger Brotherhood #6), author: J.R. Ward



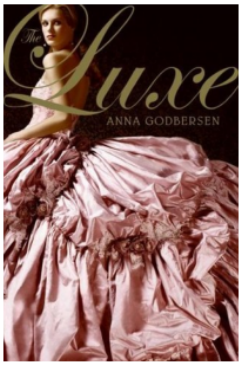
title: The Sandman Omnibus, Vol. 1, series: (The Sandman Omnibus #1), author: Neil Gaiman



title: Tales of Edgar Allan Poe, author: Edgar Allan Poe, Harry Clarke



title: The Luxe, series: (Luxe #1), author: Anna Godbersen



title: In the Hand of the Goddess, series: (Song of the Lioness #2), author: Tamora Pierce

Room Book Recommendation

In [ ]:
```
for id in recommended_books_room.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
```

title: The New Puppy from the Black Lagoon, series: (Black Lagoon Adventures #33), author: Mike Thaler, Jared Lee

Recommending Books Based on tag name of the selected Books

In [ ]:
```
def get_books_same_tag_name(selected_book_name):
    selected_book_row = goodreads_books_tags[goodreads_books_tags["title"]
                                             == selected_book_name]
    return goodreads_books_tags[(goodreads_books_tags["tag_name"]
                                 == selected_book_row.tag_name.values[0]) &
                                 != selected_book_row.title.values[0])]


recommended_books_stardust = get_books_same_tag_name(
    "Stardust").sample(5)
recommended_books_room = get_books_same_tag_name("Room").sample(5)


# print("Star dust recommendations: ",recommended_books_stardust['title'].v
# print("Room recommendations: ",recommended_books_room['title'].values)
```

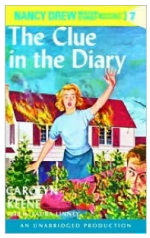
Stardust Recommendation

In [ ]:
```
for id in recommended_books_stardust.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: Beautifully Broken, series: (Infinite Love #3), author: Kira Adams

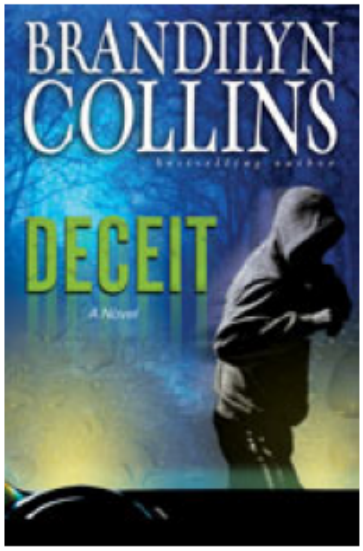
title: The Clue in the Diary, series: (Nancy Drew Mystery Stories #7), author: Carolyn Keene



title: Evil Librarian, series: (Evil Librarian #1), author: Michelle Knudsen

title: Naked Want, author: Emma Dask

title: Deceit, author: Brandilyn Collins

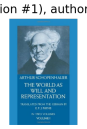


Room Book Recommendation

In [ ]:
```
for id in recommended_books_room.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

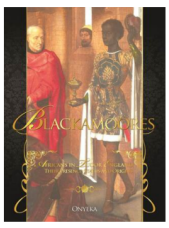
```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: The World as Will and Representation, Vol. 1, series: (The World as Will and Representation #1), author: Arthur Schopenhauer, Judith Norman, E.F.J. Payne, Alistair Welchman, Christopher Janaway



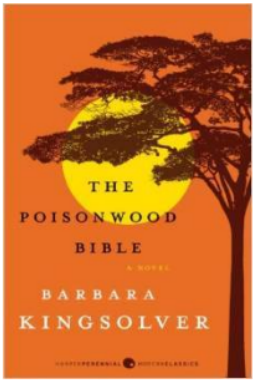
title: Curse's Claim, series: (Chaos Bleeds MC #3), author: Sam Crescent

title: Blackamoores: Africans in Tudor England, Their Presence, Status and Origins, author: Onyeka



title: Fallen, series: (Fallen #1), author: Lauren Kate

title: The Poisonwood Bible, author: Barbara Kingsolver



Recommending Books based on review count of the selected Book

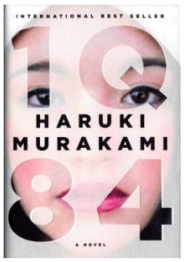
In [ ]:
```
def get_books_by_review_count(selected_book_name):
    selected_book_row = goodreads_books_tags[goodreads_books_tags["title"]
                                             == selected_book_name]
    return goodreads_books_tags[(goodreads_books_tags["review_count"]
                                 >= selected_book_row.review_count.values[0]
                                 != selected_book_row.title.values[0])]


recommended_books_stardust = get_books_by_review_count(
    "Stardust").sample(5)
recommended_books_room = get_books_by_review_count("Room").sample(5)


# print("Star dust recommendations: ",
#       recommended_books_stardust['title'].values)
# print("Room recommendations: ", recommended_books_room['title'].values)
```

Stardust Recommendation

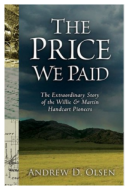
In [ ]:
```
for id in recommended_books_stardust.index:
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: 1Q84, series: (1Q84 #1-3), author: Haruki Murakami, Jay Rubin, Philip Gabriel



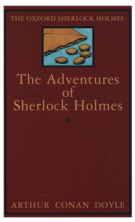
title: The Price We Paid: The Extraordinary Story of the Willie and Martin Handcart Pioneers, author: Andrew D. Olsen



title: Mystical Trash, author: Mary Ramsey

title: E l'eco rispose, author: Khaled Hosseini, Isabella Vaj

title: The Adventures of Sherlock Holmes, series: (Sherlock Holmes #3), author: Arthur Conan Doyle



Room Book Recommendation

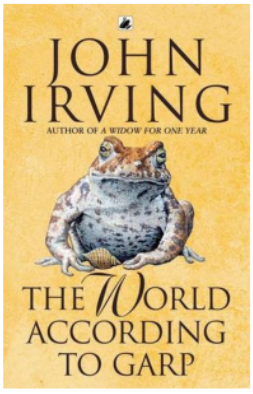
In [ ]:
```
for id in recommended_books_room.index:
```

```
    print_cover_image(goodreads_books_tags.iloc[id]['cover_link'], print_fo
        goodreads_books_tags, id))
```

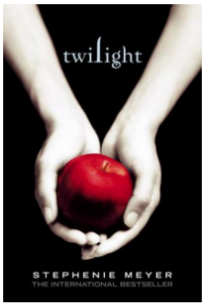
```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

title: The Lovely Bones, author: Alice Sebold

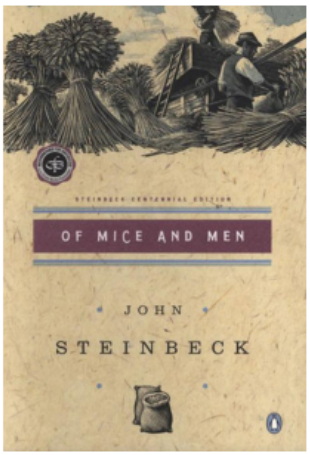
title: The World According to Garp, author: John Irving



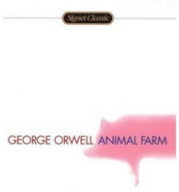
title: Twilight, series: (The Twilight Saga #1), author: Stephenie Meyer



title: Of Mice and Men, author: John Steinbeck



title: Animal Farm, author: George Orwell, Russell Baker, C.M. Woodhouse



Recommendation System Using Cosine Similarity

Cosine Similarity on average rating, tags, review_count, rating count

In [ ]:
```
goodreads_books_tags_3 = goodreads_books_tags.copy()

goodreads_books_tags_3 = goodreads_books_tags_3

goodreads_books_tags_3.loc[(goodreads_books_tags_3['average_rating'] >= 0) 
                           'rating_between'] = "between 0 and 1"
goodreads_books_tags_3.loc[(goodreads_books_tags_3['average_rating'] > 1) &
                           'rating_between'] = "between 1 and 2"
goodreads_books_tags_3.loc[(goodreads_books_tags_3['average_rating'] > 2) &
                           'rating_between'] = "between 2 and 3"
goodreads_books_tags_3.loc[(goodreads_books_tags_3['average_rating'] > 3) &
                           'rating_between'] = "between 3 and 4"
goodreads_books_tags_3.loc[(goodreads_books_tags_3['average_rating'] > 4) &
                           'rating_between'] = "between 4 and 5"

# rating_counts
# goodreads_books_tags_3.head()
```

In [ ]:
```
books_features = pd.concat([goodreads_books_tags_3['rating_between'].str.ge
    sep=","), goodreads_books_tags_3['tag_name'].str.get_dummies(
    sep=","), goodreads_books_tags_3['average_rating'], goodreads_books_tag

books_features.head()

min_max_scaler = MinMaxScaler()
books_features = min_max_scaler.fit_transform(books_features)
np.round(books_features, 2)
```

```
selected_book_features = books_features[selected_book_1_id]
room_book_features = books_features[selected_book_2_id]
```

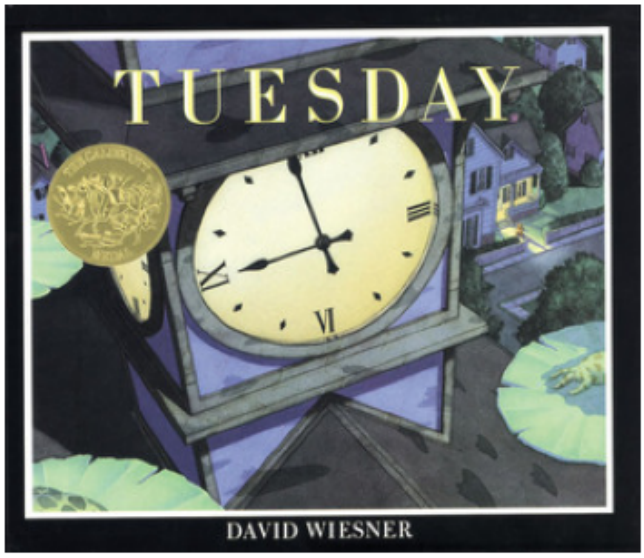
In [ ]:
```
cosine_list_stardust = np.multiply(np.ones([len(books_features), 1]),selecte
cosine_list_room = np.multiply(np.ones([len(books_features), 1]),selected_b
```

In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity
from numpy.linalg import norm
cos_sim_stardust = cosine_similarity(
    books_features[1:6000], cosine_list_stardust[1:6000])
cos_sim_room = cosine_similarity(
    books_features[1:6000], cosine_list_room[1:6000])
stardust_top5 = np.argsort(cos_sim_stardust, axis=0)[-5:][::-1]
room_top5 = np.argsort(cos_sim_room, axis=0)[-5:][::-1]
```

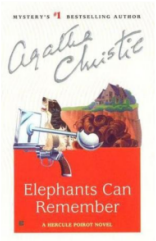
In [ ]:
```
for i in stardust_top5:
    print_cover_image(goodreads_books_tags.iloc[i[0]]['cover_link'], print_
        goodreads_books_tags, i[0]))
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

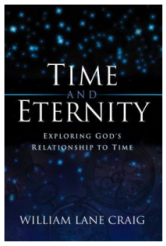
title: Tuesday, author: David Wiesner



title: Elephants Can Remember, series: (Hercule Poirot #40), author: Agatha Christie

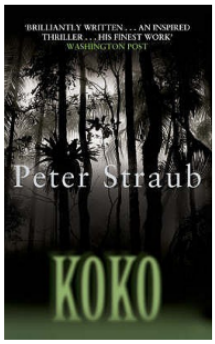


title: Time And Eternity: Exploring God's Relationship To Time, author: William Lane Craig



title: Sashenka, series: (Moscow Trilogy #1), author: Simon Sebag Montefiore

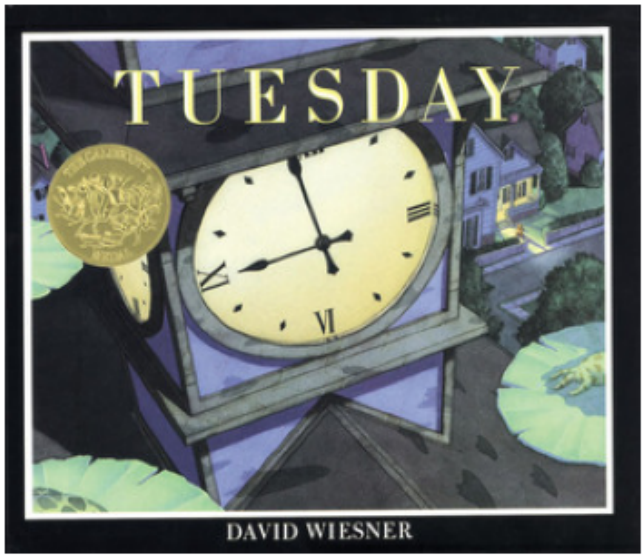
title: Koko, series: (Blue Rose Trilogy #1), author: Peter Straub



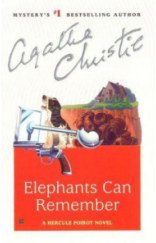
In [ ]:
```
for i in room_top5:
    print_cover_image(goodreads_books_tags.iloc[i[0]]['cover_link'], print_
```

```
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
AxesImage(500,50;3100x302)
```

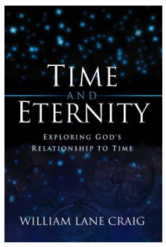
title: Tuesday, author: David Wiesner



title: Elephants Can Remember, series: (Hercule Poirot #40), author: Agatha Christie

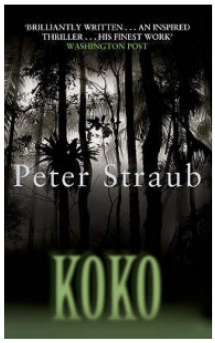


title: Time And Eternity: Exploring God's Relationship To Time, author: William Lane Craig



title: Sashenka, series: (Moscow Trilogy #1), author: Simon Sebag Montefiore

title: Koko, series: (Blue Rose Trilogy #1), author: Peter Straub



Evaluation

- Not applicable

Results

- The K means, K nearest neighbor and cosine similarity are more reliable for book recommendations based on similar books when compared to 1R Classifier.
- K means and K nearest neighbor are more scalable in performance when data is large.
- When dealing with cosine similarity as the data set size increases it requires a lot of memory for saving the similarity vectors.
- Cosine similarity needs one time execution to provide better recommendations.

References:

NumPy.Retrieved (2022, Jul 27)

Pandas Package. Retrieved (2022, Jul 27)

Matplotlib.Retrieved (2022, Jul 27)

Seaborn.Retrieved (2022, Jul 27)

Sk learn. Retrieved (2022, Jul 27)

Syntax for filter function (2022, July 27)

To read specific columns from CSV (2022, July 27)

Selecting Columns from group_by object in pandas

How to change the figure size for displot

Recommendation System Algorithms: An Overview