|  |
| --- |
| **BOOK RECOMMENDATION SYSTEM** |

TEAM 1:

VISHNU SUDHARSAN .A

DANVANTH .S

ARUN PRASAD .I

HARISH .S

DHARANISH .S

CODE:

import numpy as np

import pandas as pd

import os

import seaborn as sns

import isbnlib

from newspaper import Article

import matplotlib.pyplot as plt

plt.style.use('ggplot')

from tqdm import tqdm

from progressbar import ProgressBar

import re

from scipy.cluster.vq import kmeans, vq

from pylab import plot, show

from matplotlib.lines import Line2D

import matplotlib.colors as mcolors

import goodreads\_api\_client as gr

from sklearn.cluster import KMeans

from sklearn import neighbors

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv('C:/Users/ompra/Desktop/books.csv', error\_bad\_lines = False)

df.index = df['bookID']

#Finding Number of rows and columns

print("Dataset contains {} rows and {} columns".format(df.shape[0], df.shape[1]))

df.head()

df.replace(to\_replace='J.K. Rowling-Mary GrandPré', value = 'J.K. Rowling', inplace=True)

df.head()

#Taking the first 20:

sns.set\_context('poster')

plt.figure(figsize=(20,15))

books = df['title'].value\_counts()[:20]

rating = df.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()

sns.set\_context('paper')

plt.figure(figsize=(15,10))

ax = df.groupby('language\_code')['title'].count().plot.bar()

plt.title('Language Code')

plt.xticks(fontsize = 15)

for p in ax.patches:

ax.annotate(str(p.get\_height()), (p.get\_x()-0.3, p.get\_height()+100))

most\_rated = df.sort\_values('ratings\_count', ascending = False).head(10).set\_index('title')

plt.figure(figsize=(15,10))

sns.barplot(most\_rated['ratings\_count'], most\_rated.index, palette='rocket')

sns.set\_context('talk')

most\_books = df.groupby('authors')['title'].count().reset\_index().sort\_values('title', ascending=False).head(10).set\_index('authors')

plt.figure(figsize=(15,10))

ax = sns.barplot(most\_books['title'], most\_books.index, palette='icefire\_r')

ax.set\_title("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())), fontsize = 10, color = 'k')

client = gr.Client(developer\_key= 'fgwnppR6Q1wpFt0n6umUQ')

#Finding the top 15 authors with the most number of books

df['authors'].value\_counts().head(10)

def segregation(data):

values = []

for val in data.average\_rating:

if val>=0 and val<=1:

values.append("Between 0 and 1")

elif val>1 and val<=2:

values.append("Between 1 and 2")

elif val>2 and val<=3:

values.append("Between 2 and 3")

elif val>3 and val<=4:

values.append("Between 3 and 4")

elif val>4 and val<=5:

values.append("Between 4 and 5")

else:

values.append("NaN")

print(len(values))

return values

df.average\_rating.isnull().value\_counts()

df.dropna(0, inplace=True)

#Removing Any null values

plt.figure(figsize=(10,10))

rating= df.average\_rating.astype(float)

sns.distplot(rating, bins=20)

df['Ratings\_Dist'] = segregation(df)

ratings\_pie = df['Ratings\_Dist'].value\_counts().reset\_index()

labels = ratings\_pie['index']

colors = ['lightblue','darkmagenta','coral','bisque', 'black']

percent = 100.\*ratings\_pie['Ratings\_Dist']/ratings\_pie['Ratings\_Dist'].sum()

fig, ax1 = plt.subplots()

ax1.pie(ratings\_pie['Ratings\_Dist'],colors = colors,

pctdistance=0.85, startangle=90, explode=(0.05, 0.05, 0.05, 0.05, 0.05))

#Draw a circle now:

centre\_circle = plt.Circle((0,0), 0.70, fc ='white')

fig1 = plt.gcf()

fig1.gca().add\_artist(centre\_circle)

#Equal Aspect ratio ensures that pie is drawn as a circle

plt.axis('equal')

plt.tight\_layout()

labels = ['{0} - {1:1.2f} %'.format(i,j) for i,j in zip(labels, percent)]

plt.legend( labels, loc = 'best',bbox\_to\_anchor=(-0.1, 1.),)

#Checking for any relation between them.

plt.figure(figsize=(15,10))

df.dropna(0, inplace=True)

sns.set\_context('paper')

ax =sns.jointplot(x="average\_rating",y='text\_reviews\_count', kind='scatter', data= df[['text\_reviews\_count', 'average\_rating']])

ax.set\_axis\_labels("Average Rating", "Text Review Count")

plt.show()

trial = df[~(df['text\_reviews\_count']>5000)]

#Checking for any relation between them.

plt.figure(figsize=(15,10))

df.dropna(0, inplace=True)

sns.set\_context('paper')

ax =sns.jointplot(x="average\_rating",y='text\_reviews\_count', kind='scatter', data= trial, color = 'green')

ax.set\_axis\_labels("Average Rating", "Text Review Count")

plt.show()

plt.figure(figsize=(15,10))

sns.set\_context('paper')

ax = sns.jointplot(x="average\_rating", y="# num\_pages", data = df, color = 'crimson')

ax.set\_axis\_labels("Average Rating", "Number of Pages")

trial = df[~(df['# num\_pages']>1000)]

ax = sns.jointplot(x="average\_rating", y="# num\_pages", data = trial, color = 'darkcyan')

ax.set\_axis\_labels("Average Rating", "Number of Pages")

sns.set\_context('paper')

ax = sns.jointplot(x="average\_rating", y="ratings\_count", data = df, color = 'blueviolet')

ax.set\_axis\_labels("Average Rating", "Ratings Count")

trial = df[~(df.ratings\_count>2000000)]

sns.set\_context('paper')

ax = sns.jointplot(x="average\_rating", y="ratings\_count", data = trial, color = 'brown')

ax.set\_axis\_labels("Average Rating", "Ratings Count")

most\_text = df.sort\_values('text\_reviews\_count', ascending = False).head(10).set\_index('title')

plt.figure(figsize=(15,10))

sns.set\_context('poster')

ax = sns.barplot(most\_text['text\_reviews\_count'], most\_text.index, palette='magma')

for i in ax.patches:

ax.text(i.get\_width()+2, i.get\_y()+0.5,str(round(i.get\_width())), fontsize=10,color='black')

plt.show()

trial = df[['average\_rating', 'ratings\_count']]

data = np.asarray([np.asarray(trial['average\_rating']), np.asarray(trial['ratings\_count'])]).T

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")

#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)

# 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= 'o', markerfacecolor='red')

circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')

circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0, marker= 'o', markerfacecolor='yellow')

circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')

circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')

plt.legend((circle1, circle2, circle3, circle4, circle5)

, ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )

plt.show()

trial.idxmax()

trial.drop(3, inplace = True)

trial.drop(41865, inplace = True)

data = np.asarray([np.asarray(trial['average\_rating']), np.asarray(trial['ratings\_count'])]).T

#Computing K means with K = 8, thus, taking it as 8 clusters

centroids, \_ = kmeans(data, 5)

#assigning each sample to a cluster

#Vector Quantisation:

idx, \_ = vq(data, centroids)

# 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= 'o', markerfacecolor='red')

circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')

circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0, marker= 'o', markerfacecolor='yellow')

circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')

circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')

plt.legend((circle1, circle2, circle3, circle4, circle5)

, ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )

plt.show()

books\_features = pd.concat([df['Ratings\_Dist'].str.get\_dummies(sep=","), df['average\_rating'], df['ratings\_count']], axis=1)

books\_features.head()

min\_max\_scaler = MinMaxScaler()

books\_features = min\_max\_scaler.fit\_transform(books\_features)

np.round(books\_features, 2)

model = neighbors.NearestNeighbors(n\_neighbors=6, algorithm='ball\_tree')

model.fit(books\_features)

distance, indices = model.kneighbors(books\_features)

def get\_index\_from\_name(name):

return df[df["title"]==name].index.tolist()[0]

all\_books\_names = list(df.title.values)

def get\_id\_from\_partial\_name(partial):

for name in all\_books\_names:

if partial in name:

print(name,all\_books\_names.index(name))

def print\_similar\_books(query=None,id=None):

if id:

for id in indices[id][1:]:

print(df.iloc[id]["title"])

if query:

found\_id = get\_index\_from\_name(query)

for id in indices[found\_id][1:]:

print(df.iloc[id]["title"])

print\_similar\_books("The Catcher in the Rye")

print\_similar\_books("The Hobbit or There and Back Again")

get\_id\_from\_partial\_name("Harry Potter and the ")

print\_similar\_books(id = 1) #ID for the Book 5

OUTPUT:







 

