**Book Recommendation system**

[**https://thecleverprogrammer.com/2021/01/17/book-recommendation-system/#google\_vignette**](https://thecleverprogrammer.com/2021/01/17/book-recommendation-system/#google_vignette)

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn import neighbors

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

from sklearn.preprocessing import MinMaxScaler

**Read the dataset**

from google.colab import files

uploaded = files.upload()

df = pd.read\_csv('books.csv',error\_bad\_lines = False)

df.head()

**Data Exploration**

df.isnull().sum()

df.describe()

**Top 10 books based on average rating**

top\_ten = df[df['ratings\_count'] > 1000000]

top\_ten.sort\_values(by='average\_rating', ascending=False)

plt.style.use('seaborn-whitegrid')

plt.figure(figsize=(10, 10))

data = top\_ten.sort\_values(by='average\_rating', ascending=False).head(10)

sns.barplot(x="average\_rating", y="title", data=data, palette='inferno')

**Top 10 authors based on no of books**

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='inferno')

ax.set\_title("Top 10 authors with most books")

ax.set\_xlabel("Total number of books")

totals = []

for i in ax.patches:

    totals.append(i.get\_width())

total = sum(totals)

for i in ax.patches:

    ax.text(i.get\_width()+.2, i.get\_y()+.2,str(round(i.get\_width())), fontsize=15,color='black')

plt.show()

**Books that have reviewd the most**

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

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

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

totals = []

for i in ax.patches:

    totals.append(i.get\_width())

total = sum(totals)

for i in ax.patches:

    ax.text(i.get\_width()+.2, i.get\_y()+.2,str(round(i.get\_width())), fontsize=15,color='black')

plt.show()

**Relation between average scores and number of scores**

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

fig, ax = plt.subplots(figsize=[15,10])

sns.distplot(df['average\_rating'],ax=ax)

ax.set\_title('Average rating distribution for all books',fontsize=20)

ax.set\_xlabel('Average rating',fontsize=13)

**Relation between ratings count and average ratings**

ax = sns.relplot(data=df, x="average\_rating", y="ratings\_count", color = 'red', sizes=(100, 200), height=7, marker='o')

plt.title("Relation between Rating counts and Average Ratings",fontsize = 15)

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

**Relation between no of pages and average ratings**

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

ax = sns.relplot(x="average\_rating", y="  num\_pages", data = df, color = 'red',sizes=(100, 200), height=7, marker='o')

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

**Data Preparation**

df2 = df.copy()

df2.loc[ (df2['average\_rating'] >= 0) & (df2['average\_rating'] <= 1), 'rating\_between'] = "between 0 and 1"

df2.loc[ (df2['average\_rating'] > 1) & (df2['average\_rating'] <= 2), 'rating\_between'] = "between 1 and 2"

df2.loc[ (df2['average\_rating'] > 2) & (df2['average\_rating'] <= 3), 'rating\_between'] = "between 2 and 3"

df2.loc[ (df2['average\_rating'] > 3) & (df2['average\_rating'] <= 4), 'rating\_between'] = "between 3 and 4"

df2.loc[ (df2['average\_rating'] > 4) & (df2['average\_rating'] <= 5), 'rating\_between'] = "between 4 and 5"

**Creating Dataframes**

rating\_df = pd.get\_dummies(df2['rating\_between'])

language\_df = pd.get\_dummies(df2['language\_code'])

**Concatenating dataframes**

features = pd.concat([rating\_df,

language\_df,

df2['average\_rating'],

df2['ratings\_count']], axis=1)

**Book Recommendation**

from sklearn.preprocessing import MinMaxScaler

min\_max\_scaler = MinMaxScaler()

features = min\_max\_scaler.fit\_transform(features)

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

model.fit(features)

dist, idlist = model.kneighbors(features)

def BookRecommender(book\_name):

    book\_list\_name = []

    book\_id = df2[df2['title'] == book\_name].index

    book\_id = book\_id[0]

    for newid in idlist[book\_id]:

        book\_list\_name.append(df2.loc[newid].title)

    return book\_list\_name

BookNames = BookRecommender('Harry Potter and the Half-Blood Prince (Harry Potter  #6)')

BookNames