import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

books = pd.read\_csv('BX-Books.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

books.columns = ['ISBN', 'bookTitle', 'bookAuthor', 'yearOfPublication', 'publisher', 'imageUrlS', 'imageUrlM', 'imageUrlL']

users = pd.read\_csv('BX-Users.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

users.columns = ['userID', 'Location', 'Age']

ratings = pd.read\_csv('BX-Book-Ratings.csv', sep=';', error\_bad\_lines=False, encoding="latin-1")

ratings.columns = ['userID', 'ISBN', 'bookRating']

print(ratings.shape)

print(list(ratings.columns))

plt.rc("font", size=15)

ratings.bookRating.value\_counts(sort=False).plot(kind='bar')

plt.title('Rating Distribution\n')

plt.xlabel('Rating')

plt.ylabel('Count')

plt.savefig('system1.png', bbox\_inches='tight')

plt.show()

print(books.shape)

print(list(books.columns))

print(users.shape)

print(list(users.columns))

users.Age.hist(bins=[0, 10, 20, 30, 40, 50, 100])

plt.title('Age Distribution\n')

plt.xlabel('Age')

plt.ylabel('Count')

plt.savefig('system2.png', bbox\_inches='tight')

plt.show()

rating\_count = pd.DataFrame(ratings.groupby('ISBN')['bookRating'].count())

rating\_count.sort\_values('bookRating', ascending=False).head()

most\_rated\_books = pd.DataFrame(['0971880107', '0316666343', '0385504209', '0060928336', '0312195516'], index=np.arange(5), columns = ['ISBN'])

most\_rated\_books\_summary = pd.merge(most\_rated\_books, books, on='ISBN')

most\_rated\_books\_summary

average\_rating = pd.DataFrame(ratings.groupby('ISBN')['bookRating'].mean())

average\_rating['ratingCount'] = pd.DataFrame(ratings.groupby('ISBN')['bookRating'].count())

average\_rating.sort\_values('ratingCount', ascending=False).head()

counts1 = ratings['userID'].value\_counts()

ratings = ratings[ratings['userID'].isin(counts1[counts1 >= 200].index)]

counts = ratings['bookRating'].value\_counts()

ratings = ratings[ratings['bookRating'].isin(counts[counts >= 100].index)]

ratings\_pivot = ratings.pivot(index='userID', columns='ISBN').bookRating

userID = ratings\_pivot.index

ISBN = ratings\_pivot.columns

print(ratings\_pivot.shape)

ratings\_pivot.head()

bones\_ratings = ratings\_pivot['0316666343']

similar\_to\_bones = ratings\_pivot.corrwith(bones\_ratings)

corr\_bones = pd.DataFrame(similar\_to\_bones, columns=['pearsonR'])

corr\_bones.dropna(inplace=True)

corr\_summary = corr\_bones.join(average\_rating['ratingCount'])

corr\_summary[corr\_summary['ratingCount']>=300].sort\_values('pearsonR', ascending=False).head(10)

bones\_ratings = ratings\_pivot['0316666343']

similar\_to\_bones = ratings\_pivot.corrwith(bones\_ratings)

corr\_bones = pd.DataFrame(similar\_to\_bones, columns=['pearsonR'])

corr\_bones.dropna(inplace=True)

corr\_summary = corr\_bones.join(average\_rating['ratingCount'])

corr\_summary[corr\_summary['ratingCount']>=300].sort\_values('pearsonR', ascending=False).head(10)

books\_corr\_to\_bones = pd.DataFrame(['0312291639', '0316601950', '0446610038', '0446672211', '0385265700', '0345342968', '0060930535', '0375707972', '0684872153'],

index=np.arange(9), columns=['ISBN'])

corr\_books = pd.merge(books\_corr\_to\_bones, books, on='ISBN')

corr\_books

combine\_book\_rating = pd.merge(ratings, books, on='ISBN')

columns = ['yearOfPublication', 'publisher', 'bookAuthor', 'imageUrlS', 'imageUrlM', 'imageUrlL']

combine\_book\_rating = combine\_book\_rating.drop(columns, axis=1)

combine\_book\_rating.head()

combine\_book\_rating = pd.merge(ratings, books, on='ISBN')

columns = ['yearOfPublication', 'publisher', 'bookAuthor', 'imageUrlS', 'imageUrlM', 'imageUrlL']

combine\_book\_rating = combine\_book\_rating.drop(columns, axis=1)

combine\_book\_rating.head()

combine\_book\_rating = combine\_book\_rating.dropna(axis = 0, subset = ['bookTitle'])

book\_ratingCount = (combine\_book\_rating.

groupby(by = ['bookTitle'])['bookRating'].

count().

reset\_index().

rename(columns = {'bookRating': 'totalRatingCount'})

[['bookTitle', 'totalRatingCount']]

)

book\_ratingCount.head()

rating\_with\_totalRatingCount = combine\_book\_rating.merge(book\_ratingCount, left\_on = 'bookTitle', right\_on = 'bookTitle', how = 'left')

rating\_with\_totalRatingCount.head()

pd.set\_option('display.float\_format', lambda x: '%.3f' % x)

print(book\_ratingCount['totalRatingCount'].describe())

print(book\_ratingCount['totalRatingCount'].quantile(np.arange(.9, 1, .01)))

popularity\_threshold = 50

rating\_popular\_book = rating\_with\_totalRatingCount.query('totalRatingCount >= @popularity\_threshold')

rating\_popular\_book.head()

combined = rating\_popular\_book.merge(users, left\_on = 'userID', right\_on = 'userID', how = 'left')

us\_canada\_user\_rating = combined[combined['Location'].str.contains("usa|canada")]

us\_canada\_user\_rating=us\_canada\_user\_rating.drop('Age', axis=1)

us\_canada\_user\_rating.head()

[{"metadata":{"trusted":true},"cell\_type":"code","source":"combined = rating\_popular\_book.merge(users, left\_on = 'userID', right\_on = 'userID', how = 'left')\n\nus\_canada\_user\_rating = combined[combined['Location'].str.contains(\"usa|canada\")]\nus\_canada\_user\_rating=us\_canada\_user\_rating.drop('Age', axis=1)\nus\_canada\_user\_rating.head()\n\nus\_canada\_user\_rating = us\_canada\_user\_rating.drop\_duplicates(['userID', 'bookTitle'])\nus\_canada\_user\_rating\_pivot = us\_canada\_user\_rating.pivot(index = 'bookTitle', columns = 'userID', values = 'bookRating').fillna(0)\nus\_canada\_user\_rating\_matrix = csr\_matrix(us\_canada\_user\_rating\_pivot.values)\n\nfrom sklearn.neighbors import NearestNeighbors\n\nmodel\_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')\nmodel\_knn.fit(us\_canada\_user\_rating\_matrix)\n\nquery\_index = np.random.choice(us\_canada\_user\_rating\_pivot.shape[0])\ndistances, indices = model\_knn.kneighbors(us\_canada\_user\_rating\_pivot.iloc[query\_index, :].reshape(1, -1), n\_neighbors = 6)\n\nfor i in range(0, len(distances.flatten())):\n if i == 0:\n print('Recommendations for {0}:\\n'.format(us\_canada\_user\_rating\_pivot.index[query\_index]))\n else:\n print('{0}: {1}, with distance of {2}:'.format(i, us\_canada\_user\_rating\_pivot.index[indices.flatten()[i]], distances.flatten()[i]))\n \nus\_canada\_user\_rating\_pivot2 = us\_canada\_user\_rating.pivot(index = 'userID', columns = 'bookTitle', values = 'bookRating').fillna(0)\nus\_canada\_user\_rating\_pivot2.head() \nus\_canada\_user\_rating\_pivot2.shape\n\nX = us\_canada\_user\_rating\_pivot2.values.T\nX.shape\n\nimport sklearn\nfrom sklearn.decomposition import TruncatedSVD\n\nSVD = TruncatedSVD(n\_components=12, random\_state=17)\nmatrix = SVD.fit\_transform(X)\nmatrix.shape\n\nimport warnings\nwarnings.filterwarnings(\"ignore\",category =RuntimeWarning)\ncorr = np.corrcoef(matrix)\ncorr.shape\n\nus\_canada\_book\_title = us\_canada\_user\_rating\_pivot2.columns\nus\_canada\_book\_list = list(us\_canada\_book\_title)\ncoffey\_hands = us\_canada\_book\_list.index(\"The Green Mile: Coffey's Hands (Green Mile Series)\")\nprint(coffey\_hands)\n\ncorr\_coffey\_hands = corr[coffey\_hands]\nlist(us\_canada\_book\_title[(corr\_coffey\_hands0.9)])","execution\_count":null,"outputs":[]}]