**SOURCE CODE**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from nltk.corpus import stopwords

from nltk.tokenize import WordPunctTokenizer

import warnings

warnings.filterwarnings('ignore')

df = pd.read\_csv(r'yelp\_review\_arizona.csv')

df\_business = pd.read\_csv(r'yelp\_business.csv')

df.head()

#Select only stars and text

yelp\_data = df[['business\_id', 'user\_id', 'stars', 'text']]

import nltk

nltk.download('stopwords')

import string

from nltk.corpus import stopwords

stop = []

for word in stopwords.words('english'):

    s = [char for char in word if char not in string.punctuation]

    stop.append(''.join(s))

def text\_process(mess):

    """

    Takes in a string of text, then performs the following:

    1. Remove all punctuation

    2. Remove all stopwords

    3. Returns a list of the cleaned text

    """

    # Check characters to see if they are in punctuation

    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.

    nopunc = ''.join(nopunc)

    # Now just remove any stopwords

    return " ".join([word for word in nopunc.split() if word.lower() not in stop])

yelp\_data['text'] = yelp\_data['text'].apply(text\_process)

#Split train test for testing the model later

vld\_size=0.15

X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(yelp\_data['text'], df['business\_id'], test\_size = vld\_size)

userid\_df = yelp\_data[['user\_id','text']]

business\_df = yelp\_data[['business\_id', 'text']]

userid\_df.head()

userid\_df[userid\_df['user\_id']=='ZwVz20be-hOZnyAbevyMyQ']['text']

business\_df.head()

userid\_df = userid\_df.groupby('user\_id').agg({'text': ' '.join})

business\_df = business\_df.groupby('business\_id').agg({'text': ' '.join})

userid\_df.head()

userid\_df.loc['ZwVz20be-hOZnyAbevyMyQ']['text']

from sklearn.feature\_extraction.text import TfidfVectorizer

#userid vectorizer

userid\_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max\_features=5000)

userid\_vectors = userid\_vectorizer.fit\_transform(userid\_df['text'])

userid\_vectors.shape

userid\_vectors

#Business id vectorizer

businessid\_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max\_features=5000)

businessid\_vectors = businessid\_vectorizer.fit\_transform(business\_df['text'])

businessid\_vectors.shape

userid\_rating\_matrix = pd.pivot\_table(yelp\_data, values='stars', index=['user\_id'], columns=['business\_id'])

userid\_rating\_matrix.shape

userid\_rating\_matrix.head()

P = pd.DataFrame(userid\_vectors.toarray(), index=userid\_df.index, columns=userid\_vectorizer.get\_feature\_names())

Q = pd.DataFrame(businessid\_vectors.toarray(), index=business\_df.index, columns=businessid\_vectorizer.get\_feature\_names())

Q.head()

def matrix\_factorization(R, P, Q, steps=25, gamma=0.001,lamda=0.02):

    for step in range(steps):

        for i in R.index:

            for j in R.columns:

                if R.loc[i,j]>0:

                    eij=R.loc[i,j]-np.dot(P.loc[i],Q.loc[j])

                    P.loc[i]=P.loc[i]+gamma\*(eij\*Q.loc[j]-lamda\*P.loc[i])

                    Q.loc[j]=Q.loc[j]+gamma\*(eij\*P.loc[i]-lamda\*Q.loc[j])

        e=0

        for i in R.index:

            for j in R.columns:

                if R.loc[i,j]>0:

                    e= e + pow(R.loc[i,j]-np.dot(P.loc[i],Q.loc[j]),2)+lamda\*(pow(np.linalg.norm(P.loc[i]),2)+pow(np.linalg.norm(Q.loc[j]),2))

        if e<0.001:

            break

    return P,Q

%%time

P, Q = matrix\_factorization(userid\_rating\_matrix, P, Q, steps=25, gamma=0.001,lamda=0.02)

Q.head()

Q.iloc[0].sort\_values(ascending=False).head(10)

# Store P, Q and vectorizer in pickle file

import pickle

output = open('yelp\_recommendation\_model\_8.pkl', 'wb')

pickle.dump(P,output)

pickle.dump(Q,output)

pickle.dump(userid\_vectorizer,output)

output.close()

#static input

words = "i want to have dinner with excellent dessert"

test\_df= pd.DataFrame([words], columns=['text'])

test\_df['text'] = test\_df['text'].apply(text\_process)

test\_vectors = userid\_vectorizer.transform(test\_df['text'])

test\_v\_df = pd.DataFrame(test\_vectors.toarray(), index=test\_df.index, columns=userid\_vectorizer.get\_feature\_names())

predictItemRating=pd.DataFrame(np.dot(test\_v\_df.loc[0],Q.T),index=Q.index,columns=['Rating'])

topRecommendations=pd.DataFrame.sort\_values(predictItemRating,['Rating'],ascending=[0])[:7]

for i in topRecommendations.index:

    print(df\_business[df\_business['business\_id']==i]['name'].iloc[0])

    print(df\_business[df\_business['business\_id']==i]['categories'].iloc[0])

    print(str(df\_business[df\_business['business\_id']==i]['stars'].iloc[0])+ ' '+str(df\_business[df\_business['business\_id']==i]['review\_count'].iloc[0]))

    print('')

#dynamic input

words = input()

test\_df= pd.DataFrame([words], columns=['text'])

test\_df['text'] = test\_df['text'].apply(text\_process)

test\_vectors = userid\_vectorizer.transform(test\_df['text'])

test\_v\_df = pd.DataFrame(test\_vectors.toarray(), index=test\_df.index, columns=userid\_vectorizer.get\_feature\_names())

predictItemRating=pd.DataFrame(np.dot(test\_v\_df.loc[0],Q.T),index=Q.index,columns=['Rating'])

topRecommendations=pd.DataFrame.sort\_values(predictItemRating,['Rating'],ascending=[0])[:7]

for i in topRecommendations.index:

    print(df\_business[df\_business['business\_id']==i]['name'].iloc[0])

    print(df\_business[df\_business['business\_id']==i]['categories'].iloc[0])

    print(str(df\_business[df\_business['business\_id']==i]['stars'].iloc[0])+ ' '+str(df\_business[df\_business['business\_id']==i]['review\_count'].iloc[0]))

    print('')