CSE4077- Recommender Systems

J Component – Review 2 Project Report

RECOMMENDATION BASED ON AMAZON FOOD REVIEW

By

19MIA1065	B. Phanindra Sai
19MIA1071	O. Naga Sai Kumar
19MIA1086	T. Siva Nikhil
19MIA1032	M. Jay Kumar Patel

M.Tech CSE with Specialization Business Analytics

Submitted to

Dr. A. Bhuvaneswari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

August 2022

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 22-23

Worklet details

Programme	M.Tech with Specialization Business Analytics	
Course Name / Code	Recommender system/CSE4077	
Slot	E1+TE1	
Faculty Name	Dr. A. BHUVANESWARI	
Component	J – Component	
J Component Title	Recommendation based on Amazon Food review	
Team Members Name | Reg. No	19MIA1065	B. Phanindra Sai
	19MIA1071	O. Naga Sai Kumar
	19MIA1086	T. Siva Nikhil
	19MIA1032	M. Jay Kumar Patel

Team Members(s) Contributions – Tentatively planned for implementation:

Worklet Tasks	*Contributor's Names*
Data collection & literature survey	Sai Kumar, Jay Kumar Patel, Siva Nikhil, Phanindra Sai
Preprocessing	Jay Kumar Patel, Phanindra Sai
Model building	Phanindra Sai, jay Kumar Patel, Siva Nikhil, Sai Kumar
Visualization	Siva Nikhil, Sai Kumar
Technical Report writing	Sai Kumar, Siva Nikhil
Presentation preparation	Phanindra Sai, Jay Kumar Patel

ABSTRACT

Amazon sells lots of products worldwide and it plays a vital role in our life. now we are analyzing more on their food products. Considering that everyone has different purchase profile, a recommendation system is required to help and give a personalized suggestion products based on the user's preferences. In recent years, consumer interest in shopping online is increased globally with a focus on home delivery. We have data filled with reviews and the ingredients of food. We are trying out content based, popularity based, collaborative based filtering and SVM methods and we are finding out the best and their performances. We will be making the model using the reviews of the people who purchased past and their reviews.

1.Introduction

Almost all e-commerce websites allow users to rate the products or services which they received when shopping. These feedbacks serve as suggestions to other users and are influential to people's decisions on whether to buy the product. Therefore, exploring and understanding the ratings has become an important way to understand the need of users. Moreover, applying the data to build a better recommendation system is an integral part of the success of a company. Recommendation systems of Amazon brings more than 30% of revenues, and Netflix, where 75% of what people watch is from some sort of recommendation. Based on the Amazon Data, we built a recommendation system for Amazon users. We implemented Matrix Factorization, SVD, Deep Learning, content based, popularity based, collaborative based filtering. We compared different methods and made a combination of some methods to provide a better recommendation. The problem we are going to solve is how to help users select products which they may like and to make recommendation to stimulate sales and increase profits. Firstly, we decided to choose the Amazon Fine Food Reviews dataset which consists of 568,454 food reviews Amazon users left up to October 2012 as our dataset. Secondly, our recommendation system is based on users rating prediction. We assume that users tend to like the products that have a score of greater than 4 and we will consider the highest 5 scores product as our recommendation candidates. Thirdly, we implemented several algorithms to predict the scores of each product for each user. 2.2 Distance Based Model Here we use the cosine-distance to give the similarity between vectors. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The similarity ranges from -1 to 1 where -1 means exactly opposite, 1 means exactly the same and in-between values indicating intermediate similarity or dissimilarity.

2. Literature Survey

Sl no	Title	Author / Journal name / Year	Technique	Result
1	Diet-Right: A Smart Food Recommendation System	Faisal Rehman Journal of researchgate 2017	ACO, Cloud Computing	The result shows that the highest accuracy is achieved with 110 ants. It is quite evident that when we increase the number of ants, the accuracy is also increased. Moreover, it is observed that the accuracy remains constant between 80 to 100 ants.
2	Food recommender systems for diabetic patients: a narrative review	Somaye Norouzi, Mohsen Nematy. Journal of researchgate 2017	CFRS, KBRS and CARS	Rule- based reasoning and semantic web such as food ontology and the combination of both were the most popular techniques applied to develop food recommender systems
3	A Personalized Food Recommender System for Zomato	Mansi Goel, Ayush Agarwal. Journal of arvix 2019		Best performance (0.90 F-score) is obtained on manually-annotated ground-truth dataset.
4	Recommendation System for Grocery Store Considering Data Sparsity	NatsukiSanoa, NatsumiMachino Journal of sciencedirect. 2015	SVD-type recommendation based on real POS data	The F-value of the best recommendation method for product category recommendation

				is increased 5.24 times compared to the product item method.
5	Online Grocery Recommendation System	Suja Panicker Journal of researchgate 2016	slope-one and min hash algorithms	Total number of common elements=7 Total number of elements=12 So, 7/12=0.58 That means, similarity between User1 and User2 is 58%.
6	Amazon.com Recommendations Item-to-Item Collaborative Filtering	Greg Linden, Brent Smith, and Jeremy York Journal of UMD 2003	Item-to-Item Collaborative Filtering, search based model	a good recommendation algorithm is scalable over very large customer bases and product catalogs, requires only subsecond processing time to generate online recommendations, is able to react immediately to changes in a user's data, and makes compelling recommendations for all users regardless of the number of purchases and ratings. Unlike other algorithms, item-to-item collaborative filtering is able to meet this challenge.

7	Amazon Food Review Classification Using Deep Learning and Recommender System	Z Zhou, L Xu Journal of stanford Systems, 2009	Feed-forward Neural Network, LSTM.	Model RMSE Popular(baseline) 1.7372 Collaborative Filtering 1.4538 Matrix Factorization 1.1198

3. Dataset and Tool to be used (Details)

https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

4. Algorithms/Techniques Description

i. Popularity Recommender model. (Non-personalized)

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

For example, if a product is often purchased by most people, then the system will get to know that that product is most popular so for every new user who just signed it, the system will recommend that product to that user also and chances becomes high that the new user will also purchase that.

ii. Build Collaborative Filtering model

It is considered to be one of the very smart recommender systems that work on the similarity between different users and also items that are widely used as an e-commerce website and also online movie websites. It checks about the taste of similar users and does recommendations.

The similarity is not restricted to the taste of the user moreover there can be consideration of similarity between different items also. The system will give more efficient recommendations if we have a large volume of information about users and items.

5. Implementation Details

Import Necessary Libraries

```
import numpy as np
import pandas as pd
import math
import json
import time
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.model_selection import train_test_split
from sklearn.neighbors import NearestNeighbors
#from sklearn.externals import joblib
import scipy.sparse
from scipy.sparse import csr_matrix
import warnings; warnings.simplefilter('ignore')
%matplotlib inline
```

Read and Explore the Data

```
#Import the data set
df = pd.read_csv('Reviews.csv')
```

```
df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```python
# Dropping the columns
df = df.drop(['Id', 'ProfileName','Time','HelpfulnessNumerator','HelpfulnessDenominator','Text','Summary'], axis = 1)
```

```python
# see few rows of the imported dataset
df.tail()
```

	ProductId	UserId	Score
568449	B001EO7N10	A28KG5XORO54AY	5
568450	B003S1WTCU	A3I8AFVPEE8KI5	2
568451	B004I613EE	A121AA1GQV751Z	5
568452	B004I613EE	A3IBEVCTXKNOH	5
568453	B001LR2CU2	A3LGQPJCZVL9UC	5

```python
# Check the number of rows and columns
rows, columns = df.shape
print("No of rows: ", rows)
print("No of columns: ", columns)
```

```
No of rows:  568454
No of columns:  3
```

```python
#Check Data types
df.dtypes
```

```
ProductId    object
UserId       object
Score         int64
dtype: object
```

```python
# Check for missing values present
print('Number of missing values across columns-\n', df.isnull().sum())
```

```
Number of missing values across columns-
 ProductId    0
UserId       0
Score        0
dtype: int64
```

There are no missing values with total records 568454

```
# Summary statistics of 'rating' variable
df[['Score']].describe().transpose()
```

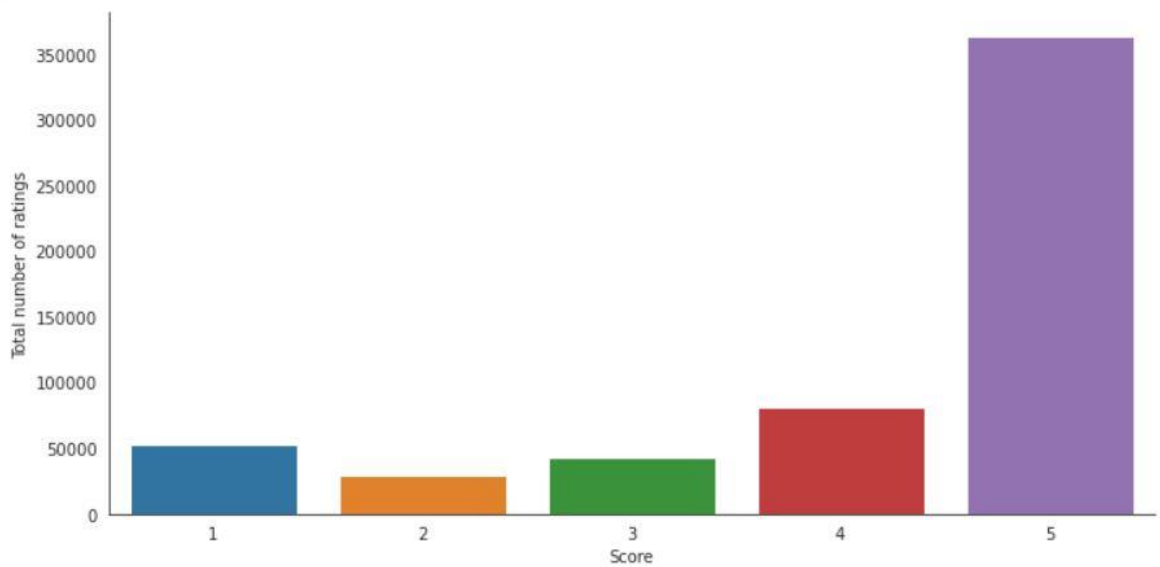
	count	mean	std	min	25%	50%	75%	max
Score	568454.0	4.183199	1.310436	1.0	4.0	5.0	5.0	5.0

```
# find minimum and maximum ratings

def find_min_max_rating():
    print('The minimum rating is: %d' %(df['Score'].min()))
    print('The maximum rating is: %d' %(df['Score'].max()))

find_min_max_rating()
```

```
The minimum rating is: 1
The maximum rating is: 5
```

Ratings are on scale of 1 - 5

```
# Check the distribution of ratings
with sns.axes_style('white'):
    g = sns.factorplot("Score", data=df, aspect=2.0,kind='count')
    g.set_ylabels("Total number of ratings")
```



```python
# Number of unique user id and product id in the data
print('Number of unique USERS in Raw data = ', df['UserId'].nunique())
print('Number of unique ITEMS in Raw data = ', df['ProductId'].nunique())
```

```
Number of unique USERS in Raw data =  256059
Number of unique ITEMS in Raw data =  74258
```

Take subset of dataset to make it less sparse/more dense. (For example, keep the users only who has given 50 or more number of ratings)

```python
# Top 10 users based on rating
most_rated = df.groupby('UserId').size().sort_values(ascending=False)[:10]
most_rated
```

```
UserId
A3OXHLG6DIBRW8    448
A1YUL9PCJR3JTY    421
AY12DBB0U420B     389
A281NPSIMI1C2R    365
A1Z54EM24Y40LL    256
A1TMAVN4CEM8U8    204
A2MUGFV2TDQ47K    201
A3TVZM3ZIXG8YW    199
A3PJZ8TU8FDQ1K    178
AQQLWCMRNDFGI     176
dtype: int64
```

Data model preparation as per requirement on number of minimum ratings

```python
counts = df['UserId'].value_counts()
df_final = df[df['UserId'].isin(counts[counts >= 50].index)]
```

```python
df_final.head()
```

	ProductId	UserId	Score
14	B001GVISJM	A2MUGFV2TDQ47K	5
44	B001EO5QW8	A2G7B7FKP2O2PU	5
46	B001EO5QW8	AQLL2R1PPR46X	5
109	B001REEG6C	AY12DBB0U420B	5
141	B001GVISJW	A2YIO225BTKVPU	4

```python
print('Number of users who have rated 50 or more items =', len(df_final))
print('Number of unique USERS in final data = ', df_final['UserId'].nunique())
print('Number of unique ITEMS in final data = ', df_final['ProductId'].nunique())
```

```
Number of users who have rated 50 or more items = 22941
Number of unique USERS in final data =  267
Number of unique ITEMS in final data =  11313
```

df_final has users who have rated 50 or more items

Calculate the density of the rating matrix

```
final_ratings_matrix = pd.pivot_table(df_final,index=['UserId'], columns = 'ProductId', values = "Score")
final_ratings_matrix.fillna(0,inplace=True)
print('Shape of final_ratings_matrix: ', final_ratings_matrix.shape)
given_num_of_ratings = np.count_nonzero(final_ratings_matrix)
print('given_num_of_ratings = ', given_num_of_ratings)
possible_num_of_ratings = final_ratings_matrix.shape[0] * final_ratings_matrix.shape[1]
print('possible_num_of_ratings = ', possible_num_of_ratings)
density = (given_num_of_ratings/possible_num_of_ratings)
density *= 100
print ('density: {:4.2f}%'.format(density))
```

```
Shape of final_ratings_matrix:  (267, 11313)
given_num_of_ratings =  20829
possible_num_of_ratings =  3020571
density: 0.69%
```

```
final_ratings_matrix.tail()
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9	B00004RBDU	B00004RBDZ
UserId							
AY1EF0GOH80EK	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AYB4ELCS5AM8P	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AYGJ96W5KQMUJ	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AYOMAHLWRQHUG	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AZV26LP92E6WU	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 11313 columns

```
# Matrix with one row per 'Product' and one column per 'user' for Item-bas
final_ratings_matrix_T = final_ratings_matrix.transpose()
final_ratings_matrix_T.head()
```

UserId	A100WO06OQR8BQ	A106ZCP7RSXMRU	A1080SE9X3ECK0	A10G136JEISLVR
ProductId				
7310172001	0.0	0.0	0.0	0.0
7310172101	0.0	0.0	0.0	0.0
7800648702	0.0	0.0	0.0	0.0
B00004CI84	0.0	0.0	0.0	0.0
B00004CXX9	0.0	0.0	0.0	0.0

5 rows × 267 columns

Split the data randomly into train and test dataset. (For example split it in 70/30 ratio)

```
#Split the training and test data in the ratio 70:30
train_data, test_data = train_test_split(df_final, test_size = 0.3, random_state=0)

print(train_data.head(5))
```

```
         ProductId          UserId  Score
399863  B002IEVJRY  A1N5FSCYN4796F      3
20262   B001BDDTB2  A1Q7A78VSQ5GQ4      5
139611  B001BCXTGS  A2PNOU7NXB1JE4      3
455504  B005HG9ERW  A2SZLNSI5KOQJT      3
512008  B0028PDER6   ALSAOZ1V546VT      5
```

```
def shape():
    print("Test data shape: ", test_data.shape)
    print("Train data shape: ", train_data.shape)
shape()
```

```
Test data shape:  (6883, 3)
Train data shape:  (16058, 3)
```

Build Popularity Recommender model. (Non-personalised)

```
#Count of user_id for each unique product as recommendation score
train_data_grouped = train_data.groupby('ProductId').agg({'UserId': 'count'}).reset_index()
train_data_grouped.rename(columns = {'UserId': 'score'},inplace=True)
train_data_grouped.head()
```

	ProductId	score
0	7310172001	5
1	7310172101	5
2	7800648702	1
3	B00004CI84	2
4	B00004CXX9	3

```python
#Sort the products on recommendation score
train_data_sort = train_data_grouped.sort_values(['score', 'ProductId'], ascending = [0,1])

#Generate a recommendation rank based upon score
train_data_sort['Rank'] = train_data_sort['score'].rank(ascending=0, method='first')

#Get the top 5 recommendations
popularity_recommendations = train_data_sort.head(5)
popularity_recommendations
```

	ProductId	score	Rank
5621	B002IEZJMA	48	1.0
8130	B006MONQMC	42	2.0
5620	B002IEVJRY	41	3.0
6779	B0041NYV8E	39	4.0
7876	B005HG9ET0	39	5.0

```python
# Use popularity based recommender model to make predictions
def recommend(user_id):
    user_recommendations = popularity_recommendations

    #Add user_id column for which the recommendations are being generated
    user_recommendations['UserId'] = user_id

    #Bring user_id column to the front
    cols = user_recommendations.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    user_recommendations = user_recommendations[cols]

    return user_recommendations
```

```python
find_recom = [15,121,200]   # This list is user choice.
for i in find_recom:
    print("Here is the recommendation for the userId: %d\n" %(i))
    print(recommend(i))
    print("\n")
```

```
Here is the recommendation for the userId: 15

      UserId   ProductId  score  Rank
5621      15  B002IEZJMA     48   1.0
8130      15  B006MONQMC     42   2.0
5620      15  B002IEVJRY     41   3.0
6779      15  B0041NYV8E     39   4.0
7876      15  B005HG9ET0     39   5.0


Here is the recommendation for the userId: 121

      UserId   ProductId  score  Rank
5621     121  B002IEZJMA     48   1.0
8130     121  B006MONQMC     42   2.0
5620     121  B002IEVJRY     41   3.0
6779     121  B0041NYV8E     39   4.0
7876     121  B005HG9ET0     39   5.0


Here is the recommendation for the userId: 200

      UserId   ProductId  score  Rank
5621     200  B002IEZJMA     48   1.0
8130     200  B006MONQMC     42   2.0
5620     200  B002IEVJRY     41   3.0
6779     200  B0041NYV8E     39   4.0
7876     200  B005HG9ET0     39   5.0
```

```python
print('Since this is a popularity-based recommender model, recommendations remain the same for all users')
print('\nWe predict the products based on the popularity. It is not personalized to particular user')
```

```
Since this is a popularity-based recommender model, recommendations remain the same for all users

We predict the products based on the popularity. It is not personalized to particular user
```

Build Collaborative Filtering model.

Model-based Collaborative Filtering: Singular Value Decomposition

```
df_CF = pd.concat([train_data, test_data]).reset_index()
df_CF.tail()
```

	index	ProductId	UserId	Score
22936	275741	B001M23WVY	AY1EF0GOH80EK	2
22937	281102	B002R8SLUY	A16AXQ11SZA8SQ	5
22938	205589	B00473PVVO	A281NPSIMI1C2R	5
22939	303238	B0002DGRZC	AJD41FBJD9010	5
22940	36703	B000EEWZD2	A2M9D9BDHONV3Y	3

```
#User-based Collaborative Filtering
# Matrix with row per 'user' and column per 'item'
pivot_df = pd.pivot_table(df_CF,index=['UserId'], columns = 'ProductId', values = "Score")
pivot_df.fillna(0,inplace=True)
print(pivot_df.shape)
pivot_df.head()
```

```
(267, 11313)
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9	B00004RBDU	B00004RBDZ	B00004RYGX
UserId								
A100WO06OQR8BQ	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
A106ZCP7RSXMRU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A1080SE9X3ECK0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A10G136JEISLVR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
A11ED8O95W2103	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 11313 columns

```
pivot_df['user_index'] = np.arange(0, pivot_df.shape[0], 1)
pivot_df.head()
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9
UserId					
A100WO06OQR8BQ	0.0	0.0	0.0	0.0	0.0
A106ZCP7RSXMRU	0.0	0.0	0.0	0.0	0.0
A1080SE9X3ECK0	0.0	0.0	0.0	0.0	0.0
A10G136JEISLVR	0.0	0.0	0.0	0.0	0.0
A11ED8O95W2103	0.0	0.0	0.0	0.0	0.0

5 rows × 11314 columns

SVD method

SVD is best to apply on a large sparse matrix

```
from scipy.sparse.linalg import svds
# Singular Value Decomposition
U, sigma, Vt = svds(pivot_df, k = 50)
# Construct diagonal array in SVD
sigma = np.diag(sigma)
```

Note that for sparse matrices, you can use the sparse.linalg.svds() function to perform the decomposition.

SVD is useful in many tasks, such as data compression, noise reduction similar to Principal Component Analysis and Latent Semantic Indexing (LSI), used in document retrieval and word similarity in Text mining

```
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt)

# Predicted ratings
preds_df = pd.DataFrame(all_user_predicted_ratings, columns = pivot_df.columns)
preds_df.head()
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9	B00004RBDU	B00004RBDZ	B00004RYGX
0	-0.023781	-0.023781	-0.002054	0.104898	0.104898	0.024303	0.107537	0.104898
1	-0.007905	-0.007905	-0.003851	-0.008111	-0.008111	-0.000537	-0.010274	-0.008111
2	0.002045	0.002045	0.021680	0.053874	0.053874	-0.005837	-0.008159	0.053874
3	0.000029	0.000029	-0.000028	0.000039	0.000039	-0.000002	-0.000218	0.000039
4	0.006935	0.006935	-0.000392	0.008952	0.008952	-0.000043	0.012956	0.008952

5 rows × 11313 columns

```python
# Recommend the items with the highest predicted ratings

def recommend_items(userID, pivot_df, preds_df, num_recommendations):

    user_idx = userID-1 # index starts at 0

    # Get and sort the user's ratings
    sorted_user_ratings = pivot_df.iloc[user_idx].sort_values(ascending=False)
    #sorted_user_ratings
    sorted_user_predictions = preds_df.iloc[user_idx].sort_values(ascending=False)
    #sorted_user_predictions

    temp = pd.concat([sorted_user_ratings, sorted_user_predictions], axis=1)
    temp.index.name = 'Recommended Items'
    temp.columns = ['user_ratings', 'user_predictions']

    temp = temp.loc[temp.user_ratings == 0]
    temp = temp.sort_values('user_predictions', ascending=False)
    print('\nBelow are the recommended items for user(user_id = {}):\n'.format(userID))
    print(temp.head(num_recommendations))
```

```python
#Enter 'userID' and 'num_recommendations' for the user #
userID = 121
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 121):

                   user_ratings  user_predictions
Recommended Items
B004E4EBMG                  0.0          1.553272
B004JGQ15E                  0.0          0.972833
B0061IUIDY                  0.0          0.923977
B0041NYV8E                  0.0          0.901132
B001LG940E                  0.0          0.893659
```

Evaluate both the models. (Once the model is trained on the training data, it can be used to compute the error (RMSE) on predictions made on the test data.)

Evaluation of Model-based Collaborative Filtering (SVD)

```
# Actual ratings given by the users
final_ratings_matrix.head()
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9	B00004RBDU	B00004RBDZ	B00004RYGX	B00004S1C6	B000052Y74	...
UserId											
A100WO06OQR8BQ	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...
A106ZCP7RSXMRU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
A1080SE9X3ECK0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
A10G136JEISLVR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
A11ED8O95W2103	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

5 rows × 11313 columns

```
# Average ACTUAL rating for each item
final_ratings_matrix.mean().head()
```

```
ProductId
7310172001    0.037453
7310172101    0.037453
7800648702    0.018727
B00004CI84    0.044944
B00004CXX9    0.044944
dtype: float64
```

```
# Predicted ratings
preds_df.head()
```

ProductId	7310172001	7310172101	7800648702	B00004CI84	B00004CXX9	B00004RBDU
0	-0.023781	-0.023781	-0.002054	0.104898	0.104898	0.024303
1	-0.007905	-0.007905	-0.003851	-0.008111	-0.008111	-0.000537
2	0.002045	0.002045	0.021680	0.053874	0.053874	-0.005837
3	0.000029	0.000029	-0.000028	0.000039	0.000039	-0.000002
4	0.006935	0.006935	-0.000392	0.008952	0.008952	-0.000043

5 rows × 11313 columns

```python
# Average PREDICTED rating for each item
preds_df.mean().head()
```

```
ProductId
7310172001    0.001174
7310172101    0.001174
7800648702    0.004557
B00004CI84    0.039487
B00004CXX9    0.039487
dtype: float64
```

```python
rmse_df = pd.concat([final_ratings_matrix.mean(), preds_df.mean()], axis=1)
rmse_df.columns = ['Avg_actual_ratings', 'Avg_predicted_ratings']
print(rmse_df.shape)
rmse_df['item_index'] = np.arange(0, rmse_df.shape[0], 1)
rmse_df.head()
```

```
(11313, 2)
```

ProductId	Avg_actual_ratings	Avg_predicted_ratings	item_index
7310172001	0.037453	0.001174	0
7310172101	0.037453	0.001174	1
7800648702	0.018727	0.004557	2
B00004CI84	0.044944	0.039487	3
B00004CXX9	0.044944	0.039487	4

```python
RMSE = round((((rmse_df.Avg_actual_ratings - rmse_df.Avg_predicted_ratings) ** 2).mean() ** 0.5), 5)
print('\nRMSE SVD Model = {} \n'.format(RMSE))
```

```
RMSE SVD Model = 0.00995
```

Get top - K (K = 5) recommendations. Since our goal is to recommend new products to each user based on his/her habits, we will recommend 5 new products.

```python
# Enter 'userID' and 'num_recommendations' for the user #
userID = 200
num_recommendations = 5
recommend_items(userID, pivot_df, preds_df, num_recommendations)
```

```
Below are the recommended items for user(user_id = 200):

                   user_ratings  user_predictions
Recommended Items
B004BKLHOS                  0.0          0.823791
B0061IUIDY                  0.0          0.622365
B004JRO1S2                  0.0          0.538305
B0061IUKDM                  0.0          0.534249
B002DZIL24                  0.0          0.529929
```

6. Results and Discussion

Model-based Collaborative Filtering is a personalized recommender system, the recommendations are based on the past behavior of the user and it is not dependent on any additional information.

The Popularity-based recommender system is non-personalized and the recommendations are based on frequency counts, which may be not suitable to the user. You can see the difference above for the user id 121 & 200, The Popularity based model has recommended the same set of 5 products to both but Collaborative Filtering based model has recommended entirely different list based on the user past purchase history.

7. GitHub Repository Link (where your j comp project work can be seen for assessment)

https://github.com/phanindrasai27/Amazon-food-recommendation-system

REFERENCES

[1] Pang, Bo, and Lillian Lee.” Opinion mining and sentiment analysis.” Foundations and trends in information
retrieval 2.1-2 (2008): 1-135.

[2] Lam, Savio LY, and Dik Lun Lee.” Feature reduction for neural network-based text categorization.”
Database Systems for Advanced Applications, 1999. Proceedings., 6th International Conference on. IEEE,
1999.

[3] Sundermeyer, Martin, Ralf Schlter, and Hermann Ney.” LSTM Neural Networks for Language Modeling.”
INTERSPEECH. 2012.
[4] Schafer, J. Ben, et al.” Collaborative filtering recommender systems.” The adaptive web. Springer Berlin
Heidelberg, 2007. 291-324.
[5] Lops, Pasquale, Marco De Gemmis, and Giovanni Semeraro. ” Content-based recommender systems: State
of the art and trends.” Recommender systems handbook. Springer US, 2011. 73-105.
[6] Koren, Yehuda, Robert Bell, and Chris Volinsky. ” Matrix factorization techniques for recommender systems.” Computer 8 (2009): 30-37.
[7] Van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. ” Deep content-based music recommendation.” Advances in Neural Information Processing Systems. 2013.
[8] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).
[9] Christopher Olah, Understanding LSTM Networks, retrieved from
http://colah.github.io/posts/2015-08-
Understanding-LSTMs/
[10] Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based Recommendations with
Recurrent Neural Networks. arXiv preprint arXiv:1511.06939.
[11] Chellapilla, K., Puri, S., & Simard, P. (2006, October). High performance convolutional neural networks
for document processing. In Tenth International Workshop on Frontiers in Handwriting Recognition. Suvisoft.
[12] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing
internal covariate shift. arXiv preprint arXiv:1502.03167. Chicago
[13] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13(1), 281-305. Chicago

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple
way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.
[15] Derks, E. P. P. A., Pastor, M. S., & Buydens, L. M. C. (1995). Robustness analysis of radial base function
and multi-layered feed-forward neural network models. Chemometrics and Intelligent Laboratory Systems,
28(1), 49-60.