Collaborative Filtering Recommender System for Restaurants

In this assignment, I develop a collaborative filtering-based recommender system for Yelp users. When users rated the restaurants where they have been to in Yelp app, their accounts have a record of their ratings. The recommender system can recommend new restaurants to a user by comparing this user's similarity to other users and predicting his rating to a new restaurant.

The dataset come from my final project of CSE6242 and the original data were crawled from Yelp API https://www.yelp.com/developers/documentation/v3/get_started.

The dataset contains all of the user reviews/ratings for 97 restaurants in Atlanta. Here is an example of the 'user_reviews.csv':

	restaurant_name	restaurant_id	user_id	friends	number_reviews	photos	area_AL	elite_user	date	rating	rating_mean
0	Fox Bros. Bar-B-Q	u-4wti774tFcYRLuQrnHEg	__1kMkvHH-kWVeokwZSFXw	115	3	NaN	1	0	3/18/2016	5	4.268636
1	South City Kitchen Midtown	eG-UO83g_5zDk70FIJbm2w	__48dJJcPvNgqUIEozwtpw	105	318	1219.0	0	0	8/23/2014	4	4.336328
2	The Vortex Bar And Grill - Midtown	Z2qMwUhnGt_2pA9uQbS7Uw	__48dJJcPvNgqUIEozwtpw	105	318	1219.0	0	0	8/23/2014	4	3.946784
3	Fat Matt's Rib Shack	ALYQ-uM_uMkKbkXlhWcgbQ	__48dJJcPvNgqUIEozwtpw	105	318	1219.0	0	0	8/23/2014	5	4.178538
4	Cypress Street Pint & Plate	1i63faxXI1TQ7pNILp3IPQ	__48dJJcPvNgqUIEozwtpw	105	318	1219.0	0	0	8/23/2014	4	4.028112

Figure 1. Screenshot of 5 examples of database.

To fit the collaborative filtering algorithms, I only need 3 columns in the dataset, 'restaurant_name', 'user_id' and 'rating':

	user_id	restaurant_name	rating
1	__48dJJcPvNgqUIEozwtpw	South City Kitchen Midtown	4
2	__48dJJcPvNgqUIEozwtpw	The Vortex Bar And Grill - Midtown	4
3	__48dJJcPvNgqUIEozwtpw	Fat Matt's Rib Shack	5
4	__48dJJcPvNgqUIEozwtpw	Cypress Street Pint & Plate	4
8	__bMs0nf3_hnhitK91gT4A	South City Kitchen Midtown	4
9	__bMs0nf3_hnhitK91gT4A	Atlanta Breakfast Club	5
10	__bMs0nf3_hnhitK91gT4A	Herban Fix - Vegan Kitchen	2
18	_02XN3yATdWwfMIbsGhMuQ	Slutty Vegan	5
19	_02XN3yATdWwfMIbsGhMuQ	26 Thai Kitchen & Bar	4
20	_02XN3yATdWwfMIbsGhMuQ	Sweet Georgia's Juke Joint	2

Figure 2. Screenshot of 10 examples of dataset for CF model fitting.

According to the assignment's requirement, I generated 3 size of data from this dataset, named as 'data1', 'data2' and 'data3' respectively in the title of my scripts:

```
print(data.shape)
```

```
(911, 3)
```

```
print(data.shape)
```

```
(9369, 3)
```

```
print(data.shape)
```

```
(33201, 3)
```

Figure 3. Sizes of 3 sub datasets (data1, data2, data3).

The running environment of my scripts is python/Jupyter notebook.

I used two popular CF algorithms to build the model, one is SVD algorithm in surprise (https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD),

another algorithm is lightFM 'Learning to Rank – WARP' model (https://making.lyst.com/lightfm/docs/lightfm.html).

I install surprise and lightfm library using "pip install" and import modules in Jupyter notebook.

My codes references are:

https://surprise.readthedocs.io/en/stable/FAQ.html

https://www.kaggle.com/malikasif123/amazon-reviews-recommendation-system

https://www.kaggle.com/podsyp/anime-recommendations-with-surprise

https://making.lyst.com/lightfm/docs/examples/dataset.html

https://www.kaggle.com/niyamatalmass/lightfm-hybrid-recommendation-system/execution#LightFM-Python-Library

The running time and accuracy of these two cases using different sizes of dataset are shown below:

```
# Fitting and evaluating Collobarative filtering
start_time = time.time()
def accuracy_over_n_queries(n = 10):
    for i in range(n):
        predictions = algo.fit(trainset).test(testset)
        accuracy.rmse(predictions)
    return predictions
predictions = accuracy_over_n_queries(10)

print(time.time() - start_time)
```

```
RMSE: 3.0871
RMSE: 3.0859
RMSE: 3.0863
RMSE: 3.0792
RMSE: 3.0858
RMSE: 3.0867
RMSE: 3.0873
RMSE: 3.0949
RMSE: 3.0763
RMSE: 3.0897
0.36601901054382324
```

Figure 4(a-1). Running time and accuracy (RMSE) values after 10 queries of data1, using surprise SVD.

```
from lightfm.evaluation import auc_score, precision_at_k
start_time = time.time()
scores=[]
for e in range(10):
    model.fit_partial(train, epochs=10, num_threads=4)
    auc_train= auc_score(model, train, num_threads=4).mean()
    auc_test= auc_score(model, test, num_threads=4).mean()
    scores.append((auc_train, auc_test))

scores = np.array(scores)
print(time.time() - start_time)
```

```
0.5994336605072021
```

```
print(scores)
```

```
[[0.83063036 0.1003603 ]
 [0.9856448  0.09905723]
 [0.9990808  0.09903321]
 [0.9998883  0.09903321]
 [0.99995923 0.09903321]
 [0.9999728  0.09903321]
 [0.9999804  0.09903321]
 [0.99997437 0.09903321]
 [0.9999789  0.09903321]
 [0.99997735 0.09903321]]
```

Figure 4(a-2). Running time and accuracy (AUC scores) after 10 queries of data1, using LightFM model.

```
# Fitting and evaluating Collobarative filtering
start_time = time.time()
def accuracy_over_n_queries(n = 10):
    for i in range(n):
        predictions = algo.fit(trainset).test(testset)
        accuracy.rmse(predictions)
    return predictions
predictions = accuracy_over_n_queries(10)

print(time.time() - start_time)
```

```
RMSE: 2.4537
RMSE: 2.5002
RMSE: 2.5259
RMSE: 2.5241
RMSE: 2.5899
RMSE: 2.5241
RMSE: 2.4662
RMSE: 2.4961
RMSE: 2.5070
RMSE: 2.5256
5.778536319732666
```

Figure 4(b-1). Running time and accuracy (RMSE) values after 10 queries of data2, using surprise SVD.

```
from lightfm.evaluation import auc_score, precision_at_k
start_time = time.time()
scores=[]
for e in range(10):
    model.fit_partial(train, epochs=10, num_threads=4)
    auc_train= auc_score(model, train, num_threads=4).mean()
    auc_test= auc_score(model, test, num_threads=4).mean()
    scores.append((auc_train, auc_test))

scores = np.array(scores)
print(time.time() - start_time)
```

```
586.130300283432
```

```
print(scores)
```

```
[[0.9999733  0.09988473]
 [0.9999754  0.0998848 ]
 [0.9999767  0.09988488]
 [0.99997765 0.09988491]
 [0.99997866 0.09988503]
 [0.9999801  0.09988509]
 [0.99998105 0.09988516]
 [0.99998176 0.09988522]
 [0.99998266 0.09988538]
 [0.999983   0.09988534]]
```

Figure 4(b-2). Running time and accuracy (AUC scores) after 10 queries of data2, using LightFM model.

```
# Fitting and evaluating Collobarative filtering
start_time = time.time()
def accuracy_over_n_queries(n = 10):
    for i in range(n):
        predictions = algo.fit(trainset).test(testset)
        accuracy.rmse(predictions)
    return predictions
predictions = accuracy_over_n_queries(10)

print(time.time() - start_time)
```

```
RMSE: 1.9620
RMSE: 1.9246
RMSE: 1.8917
RMSE: 1.9242
RMSE: 1.8958
RMSE: 1.9022
RMSE: 1.8611
RMSE: 1.8584
RMSE: 1.8621
RMSE: 1.8894
22.22751522064209
```

Figure 4(c-1). Running time and accuracy (RMSE) values after 10 queries of data3, using surprise SVD.

```
from lightfm.evaluation import auc_score, precision_at_k
start_time = time.time()
scores=[]
for e in range(10):
    model.fit_partial(train, epochs=10, num_threads=4)
    auc_train= auc_score(model, train, num_threads=4).mean()
    auc_test= auc_score(model, test, num_threads=4).mean()
    scores.append((auc_train, auc_test))

scores = np.array(scores)
print(time.time() - start_time)
```

```
586.130300283432
```

```
print(scores)
```

```
[[0.9999733  0.09988473]
 [0.9999754  0.0998848 ]
 [0.9999767  0.09988488]
 [0.99997765 0.09988491]
 [0.99997866 0.09988503]
 [0.9999801  0.09988509]
 [0.99998105 0.09988516]
 [0.99998176 0.09988522]
 [0.99998266 0.09988538]
 [0.999983   0.09988534]]
```

Figure 4(c-2). Running time and accuracy (AUC scores) after 10 queries of data3, using LightFM model.

Testing accuracy analysis of models after fitting:

The surprise SVD algorithm describes accuracy by RMSE value while lightfm model uses AUC score. From the accuracy results of 10 queries for each dataset, I can conclude that the accuracy of testing increase while the size of data increase (Accuracy(data1, about 1000 samples) < Accuracy(data1, about 10000 samples) < Accuracy(data1, about 30000 samples)).

Finally, after model fitting and testing, the recommender system recommends restaurants to users by predict all the ratings for the pairs (user, item). The prediction results for 3 datasets by surprise algorithm were exported in 'rate_df_1', 'rate_df_2', 'rate_df_3', respectively. Prediction examples by surprise and LightFM models are shown below:

```
def get_Iu(uid):
    try:
        return len(trainset.ur[trainset.to_inner_uid(uid)])
    except ValueError:
        return 0

def get_Ui(iid):
    try:
        return len(trainset.ir[trainset.to_inner_iid(iid)])
    except ValueError:
        return 0

df_ = pd.DataFrame(predictions, columns=['uid', 'iid', 'rui', 'est', 'details'])
df_['Iu'] = df_.uid.apply(get_Iu)
df_['Ui'] = df_.iid.apply(get_Ui)
df_['err'] = abs(df_.est - df_.rui)
df_.sort_values(by='uid', ascending=True)
df_.head()
```

	uid	iid	rui	est	details	Iu	Ui	err
0	dFKnDJNWx-B_E_ByZwa3jg	Top Spice	4.0	1.000000	{'was_impossible': False}	1	55	3.000000
1	5NUQ5BKcRpFe4JhZ5Dm33w	Nuevo Laredo Cantina	5.0	1.114087	{'was_impossible': False}	1	102	3.885913
2	BOh3_pP0Di-txZomArm4yg	True Food Kitchen - Temporarily Closed	5.0	2.089089	{'was_impossible': False}	5	110	2.910911
3	9GVHc84vDE5s5JJ2kknhkA	Nuevo Laredo Cantina	3.0	1.464628	{'was_impossible': False}	3	102	1.535372
4	CYqu48u1kHLtVcFtxn5Ctw	Der Biergarten	3.0	1.000000	{'was_impossible': False}	1	58	2.000000

```
# prediction

prediction= model.predict(np.array(data['user_int']), np.array(data['restaurant_int']))
preds= pd.DataFrame(zip(prediction, data['user_id'],data['restaurant_name'].tolist()), columns=['preds', 'user_id', 'restaurant_r
preds= preds.sort_values('preds', ascending= False)
```

```
#creating function to get top 5 Product Recommendation for each user.

preds
```

	preds	user_id	restaurant_name
31154	2.720504	xToLPRBZE9gSDS16Cf68FQ	Barcelona Inman Park
11127	2.685642	F1dzz6HTID6PLyaZeLD10Q	Antico Pizza
8401	2.679242	cjzfJV84KRuvWrht9oyNKQ	Ginya Izakaya
13113	2.677332	gvOCruiobHFJdF3rxTRgpg	Sweet Georgia's Juke Joint
26096	2.672404	SzQLyMhyfvVe2EwlHgA1pg	True Food Kitchen - Temporarily Closed
...	...	...	...
25306	-0.750487	SdLpyQrCi9uHBHAUukbmrQ	Two Urban Licks
5307	-0.751567	91TK74T3vZnliL9GHY_QTQ	5Church Atlanta
19010	-0.756937	mfCtTH4SsOHEWNjWWKrgRw	Cypress Street Pint & Plate
21720	-0.772339	OvslvzLFz1boCXm2weA_Fw	Cafe Agora
32970	-0.777127	ZmOD-CL2iYck51Yh3v08rg	Desta Ethiopian Kitchen

33201 rows × 3 columns

Figure 5. Screenshots of prediction examples provided by restaurant recommendation systems.