PART III
추천시스템

Requirement 3-1

get_top_n 함수 생성

```python
# TODO: Requirement 3-1. WRITE get_top_n
def get_top_n(algo, testset, id_list, n, user_based=True):

    results = defaultdict(list)
    if user_based:
        testset_id = []
        for user_id in testset:
            if user_id[0] in id_list:
                testset_id.append(user_id)
        predictions = algo.test(testset_id)
        for uid, cname, true_r, est, _ in predictions:
            results[uid].append((cname, est))
            pass
    else:
        testset_id = []
        for category_name in testset:
            if category_name[1] in id_list:
                testset_id.append(category_name)
        predictions = algo.test(testset_id)
        for uid, cname, true_r, est, _ in predictions:
            results[cname].append((uid, est))
            pass
    for id_, ratings in results.items():
        ratings = sorted(ratings, key = lambda x : x[-1], reverse = True)
        ratings = ratings[:n]
        results[id_] = ratings
        pass

    return results
```

user_based

item_based

top n 개 저장

```
trainset1 = data.build_full_trainset()
testset1 = trainset1.build_anti_testset()

trainset2 = data.build_full_trainset()
testset2 = trainset2.build_anti_testset()

trainset3 = data.build_full_trainset()
testset3 = trainset3.build_anti_testset()
```

Requirement 3-2-1 & 3-2-2
User-based Recommendation

- 알고리즘 : KNNBasic 유사도: cosine 파일명: 3-2-1.txt
- 알고리즘 : KNNWithMeans 유사도: pearson 파일명: 3-2-2.txt

알고리즘 설정

```
sim_options_UC = {'name' : 'cosine', 'user_based' : True}
sim_options_UP = {'name' : 'pearson', 'user_based' : True}
sim_options_UM = {'name' : 'msd', 'user_based' : True}
sim_options_UPb = {'name' : 'pearson_baseline', 'user_based' : True}

algo_UBC = surprise.KNNBasic(sim_options = sim_options_UC)
algo_UMP = surprise.KNNWithMeans(sim_options = sim_options_UP)
```

5명의 users에 대한 top-5 category 추천

```
uid_list = ['20384', '33306', '46833', '70628', '535']
```

KNN Basic, Cosine

```
algo = algo_UBC
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-2-1.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-2-1.txt

```
User ID 20384 top-5 results
Category NAME Latin American
        score 4.7277330440329886
Category NAME Irish
        score 3.7276160074780607
Category NAME Ethiopian
        score 2.952645081680763
Category NAME Brazilian
        score 2.6530735715526994
Category NAME Cambodian
        score 2.640610535398566

User ID 33306 top-5 results
Category NAME Sushi Bars
        score 7.011013876733049
Category NAME Latin American
        score 3.5770661441780276
Category NAME Southern
        score 3.0524401199182165
Category NAME Soul Food
        score 2.9943866992314345
Category NAME Scandinavian
        score 2.981588123685987

User ID 46833 top-5 results
Category NAME Gastropubs
        score 5.070975403666915
```

KNN WithMeans, Pearson

```
algo = algo_UMP
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-2-2.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-2-2.txt

```
User ID 20384 top-5 results
Category NAME Latin American
        score 4.993575476250616
Category NAME Ethiopian
        score 3.6403547500162956
Category NAME African
        score 3.6303795629318527
Category NAME Zoos
        score 3.0212488376917515
Category NAME Fondue
        score 2.9584343860090474

User ID 33306 top-5 results
Category NAME Sushi Bars
        score 8.61326673285692
Category NAME Latin American
        score 4.724408044103265
Category NAME Southern
        score 4.630860745262811
Category NAME Soul Food
        score 4.477084635670861
Category NAME African
        score 4.199149486083131

User ID 46833 top-5 results
Category NAME Gastropubs
        score 5.088726103521851
```

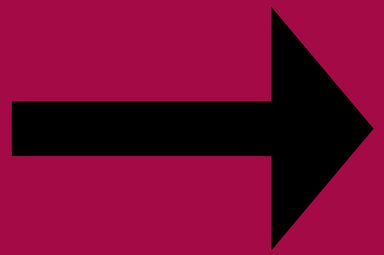
Requirement 3-2-3
User-based Recommendation Best Model

1. 알고리즘 종류

- KNNBaseline
- KNNBasic
- KNNWithMeans
- KNNWithZScore

2. 유사도 옵션 종류

- cosine
- pearson
- msd
- pearson_baseline



```
sim_options_UC = {'name' : 'cosine', 'user_based' : True}
sim_options_UP = {'name' : 'pearson', 'user_based' : True}
sim_options_UM = {'name' : 'msd', 'user_based' : True}
sim_options_UPb = {'name' : 'pearson_baseline', 'user_based' : True}

algo_UBC = surprise.KNNBasic(sim_options = sim_options_UC)
algo_UMP = surprise.KNNWithMeans(sim_options = sim_options_UP)

algo_UBP = surprise.KNNBasic(sim_options = sim_options_UP)
algo_UBM = surprise.KNNBasic(sim_options = sim_options_UM)
algo_UBPb = surprise.KNNBasic(sim_options = sim_options_UPb)

algo_UMC = surprise.KNNWithMeans(sim_options = sim_options_UC)
algo_UMM = surprise.KNNWithMeans(sim_options = sim_options_UM)
algo_UMPb = surprise.KNNWithMeans(sim_options = sim_options_UPb)

algo_UBlC = surprise.KNNBaseline(sim_options = sim_options_UC)
algo_UBlP = surprise.KNNBaseline(sim_options = sim_options_UP)
algo_UBlM = surprise.KNNBaseline(sim_options = sim_options_UM)
algo_UBlPb = surprise.KNNBaseline(sim_options = sim_options_UPb)

algo_UZC = surprise.KNNWithZScore(sim_options = sim_options_UC)
algo_UZP = surprise.KNNWithZScore(sim_options = sim_options_UP)
algo_UZM = surprise.KNNWithZScore(sim_options = sim_options_UM)
algo_UZPb = surprise.KNNWithZScore(sim_options = sim_options_UPb)
```

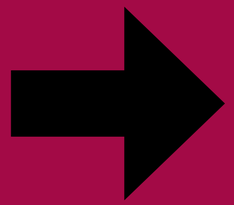
```python
UB = {}

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_UBC.fit(trainset)
    predictions = algo_UBC.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
UB['KNNBasic, Cosine'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_UBP.fit(trainset)
    predictions = algo_UBP.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
UB['KNNBasic, Pearson'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_UBM.fit(trainset)
    predictions = algo_UBM.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
UB['KNNBasic, MSD'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
```



```
{'KNNBasic, Cosine': 7.378832456650865, 'KNNBasic, Pearson':
7.3943491507343095, 'KNNBasic, MSD': 7.342860853547554,
'KNNBasic, Pearson_baseline': 7.284292265060229, 'KNNWithMeans,
Cosine': 7.458501792881423, 'KNNWithMeans, Pearson':
7.464770466182013, 'KNNWithMeans, MSD': 7.495739771919771,
'KNNWithMeans, Pearson_baseline': 7.369166125178383,
'KNNBaseline, Cosine': 7.3151877392732345, 'KNNBaseline,
Pearson': 7.31671242831038, 'KNNBaseline, MSD':
7.3031737588786925, 'KNNBaseline, Pearson_baseline':
7.270287364918886, 'KNNWithZScore, Cosine': 7.473197731980001,
'KNNWithZScore, Pearson': 7.478074657438232, 'KNNWithZScore,
MSD': 7.5244797668742125, 'KNNWithZScore, Pearson_baseline':
7.422843291064436}

The best model for user-based recommendation is KNNBaseline,
Pearson_baseline.
```

```python
print(UB, '\n')
best_algo_ub = min(UB, key=UB.get)
print('The best model for user-based recommendation is', best_algo_ub + '.\n')
```

Requirement 3-3-1 & 3-3-2
Item-based Recommendation

- 알고리즘 : KNNBasic 유사도: cosine 파일명: 3-3-1.txt
- 알고리즘 : KNNWithMeans 유사도: pearson 파일명: 3-3-2.txt

알고리즘 설정

```
sim_options_IC = {'name' : 'cosine', 'user_based' : False}
sim_options_IP = {'name' : 'pearson', 'user_based' : False}
sim_options_IM = {'name' : 'msd', 'user_based' : False}
sim_options_IPb = {'name' : 'pearson_baseline', 'user_based' : False}

algo_IBC = surprise.KNNBasic(sim_options = sim_options_IC)
algo_IMP = surprise.KNNWithMeans(sim_options = sim_options_IP)
```

5개의 bundle에 대한 top-10 user 추천

```
cname_list = ['Irish', 'Ethiopian','Wine Bars','Vegetarian','Sushi Bars']
```

KNN Basic, Cosine

```
algo = algo_IBC
algo.fit(trainset)
results = get_top_n(algo, testset, cname_list, n=10, user_based=False)
with open('3-3-1.txt', 'w') as f:
    for cname, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('Category NAME %s top-10 results\n' % cname)
        for uid, score in ratings:
            f.write('User ID %s\n\tscore %s\n' % (uid, str(score)))
        f.write('\n')
```

3-3-1.txt

```
Category NAME Ethiopian top-10 results
User ID 38778
        score 10
User ID 40506
        score 9.210966449269485
User ID 15329
        score 8.926318374246812
User ID 24096
        score 8.586472283181395
User ID 37811
        score 7.347838643254235
User ID 39479
        score 7.08952045905795
User ID 47868
        score 7.082936170805639
User ID 53391
        score 6.947261624616409
User ID 11320
        score 6.878590096675335
User ID 36633
        score 6.4711647372405

Category NAME Irish top-10 results
User ID 25745
        score 10
```

KNN WithMeans, Pearson

```
algo = algo_IMP
algo.fit(trainset)
results = get_top_n(algo, testset, cname_list, n=10, user_based=False)
with open('3-3-2.txt', 'w') as f:
    for cname, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('Category NAME %s top-10 results\n' % cname)
        for uid, score in ratings:
            f.write('User ID %s\n\tscore %s\n' % (uid, str(score)))
        f.write('\n')
```

3-3-2.txt

```
Category NAME Ethiopian top-10 results
User ID 50208
        score 10
User ID 15329
        score 9.959263698605998
User ID 38778
        score 9.113074999361384
User ID 63024
        score 5.8824727932434815
User ID 10368
        score 5.817975982853229
User ID 37811
        score 5.52895952629482
User ID 40506
        score 5.458050367770721
User ID 47868
        score 5.456611808283396
User ID 24096
        score 4.730571055112428
User ID 39479
        score 4.727224843779154

Category NAME Irish top-10 results
User ID 34916
        score 10
```

Requirement 3-3-3
Item-based Recommendation Best Model

1. 알고리즘 종류

- KNNBaseline
- KNNBasic
- KNNWithMeans
- KNNWithZScore

2. 유사도 옵션 종류

- cosine
- pearson
- msd
- pearson_baseline

```
sim_options_IC = {'name' : 'cosine', 'user_based' : False}
sim_options_IP = {'name' : 'pearson', 'user_based' : False}
sim_options_IM = {'name' : 'msd', 'user_based' : False}
sim_options_IPb = {'name' : 'pearson_baseline', 'user_based' : False}

algo_IBC = surprise.KNNBasic(sim_options = sim_options_IC)
algo_IMP = surprise.KNNWithMeans(sim_options = sim_options_IP)

algo_IBP = surprise.KNNBasic(sim_options = sim_options_IP)
algo_IBM = surprise.KNNBasic(sim_options = sim_options_IM)
algo_IBPb = surprise.KNNBasic(sim_options = sim_options_IPb)

algo_IMC = surprise.KNNWithMeans(sim_options = sim_options_IC)
algo_IMM = surprise.KNNWithMeans(sim_options = sim_options_IM)
algo_IMPb = surprise.KNNWithMeans(sim_options = sim_options_IPb)

algo_IBlC = surprise.KNNBaseline(sim_options = sim_options_IC)
algo_IBlP = surprise.KNNBaseline(sim_options = sim_options_IP)
algo_IBlM = surprise.KNNBaseline(sim_options = sim_options_IM)
algo_IBlPb = surprise.KNNBaseline(sim_options = sim_options_IPb)

algo_IZC = surprise.KNNWithZScore(sim_options = sim_options_IC)
algo_IZP = surprise.KNNWithZScore(sim_options = sim_options_IP)
algo_IZM = surprise.KNNWithZScore(sim_options = sim_options_IM)
algo_IZPb = surprise.KNNWithZScore(sim_options = sim_options_IPb)
```

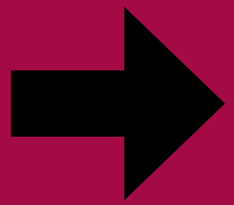
```python
IB = {}

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_IBC.fit(trainset)
    predictions = algo_IBC.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
IB['KNNBasic, Cosine'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_IBP.fit(trainset)
    predictions = algo_IBP.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
IB['KNNBasic, Pearson'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
acc=[]
for i, (trainset, testset) in enumerate(kf.split(data)):
    algo_IBM.fit(trainset)
    predictions = algo_IBM.test(testset)
    acc.append(surprise.accuracy.rmse(predictions, verbose=True))
IB['KNNBasic, MSD'] = np.mean(acc)

np.random.seed(0)
kf = KFold(n_splits=5)
```



```python
print(IB, '\n')
best_algo_ib = min(IB, key=IB.get)
print('The best model for item-based recommendation is', best_algo_ib + '.\n')
```

```
{'KNNBasic, Cosine': 8.668039893954672, 'KNNBasic, Pearson':
8.607581057506028, 'KNNBasic, MSD': 8.200015437869327, 'KNNBasic,
Pearson_baseline': 8.026819050556464, 'KNNWithMeans, Cosine':
7.409681112625191, 'KNNWithMeans, Pearson': 7.456670065408039,
'KNNWithMeans, MSD': 7.295517709350745, 'KNNWithMeans,
Pearson_baseline': 7.414168887712672, 'KNNBaseline, Cosine':
7.421570441844773, 'KNNBaseline, Pearson': 7.467820371171255,
'KNNBaseline, MSD': 7.29983187806331, 'KNNBaseline,
Pearson_baseline': 7.412552463388354, 'KNNWithZScore, Cosine':
7.296907981785795, 'KNNWithZScore, Pearson': 7.273344409914809,
'KNNWithZScore, MSD': 7.316797437223248, 'KNNWithZScore,
Pearson_baseline': 7.298836736899913}

The best model for item-based recommendation is KNNWithZScore,
Pearson.
```

Requirement 3-4-1, 3-4-2, 3-4-3, 3-4-4

Matrix-based Recommendation

- SVD(n_factors=100, n_epoch=20, biased=False) 파일명: 3-4-1.txt
- SVD(n_factors=200, n_epoch=20, biased=True) 파일명: 3-4-2.txt
- SVD++(n_factors=100, n_epoch=20) 파일명: 3-4-3.txt
- SVD++(n_factors=200, n_epoch=20) 파일명: 3-4-4.txt

알고리즘 설정

```
algo1 = surprise.SVD(n_factors=100, n_epochs=20, biased=False)
algo2 = surprise.SVD(n_factors=200, n_epochs=20, biased=True)
algo3 = surprise.SVDpp(n_factors=100, n_epochs=20)
algo4 = surprise.SVDpp(n_factors=200, n_epochs=20)
```

5명의 user에 대한 top-5 category 추천

```
uid_list = ['20384', '33306', '46833', '70628', '535']
```

SVD(n_factors = 100, n_epoch = 20, biased = False)

```
algo = algo1
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-4-1.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-4-1.txt

```
User ID 20384 top-5 results
Category NAME Vegetarian
        score 6.0237470159571345
Category NAME Breweries
        score 4.769009665319729
Category NAME Gastropubs
        score 3.0393302473539636

User ID 33306 top-5 results
Category NAME Wine Bars
        score 10
Category NAME Irish
        score 4.031160304225664
Category NAME Gastropubs
        score 4.014878225843182

User ID 46833 top-5 results
Category NAME Wine Bars
        score 8.713090707969352
Category NAME Pubs
        score 7.435734940238504
Category NAME Latin American
        score 5.263012839902044
Category NAME Breweries
        score 3.558235497734727
```

SVD(n_factors = 200, n_epoch = 20, biased = True)

```
algo = algo2
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-4-2.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-4-2.txt

```
User ID 20384 top-5 results
Category NAME Vegetarian
        score 6.037415159270821
Category NAME Gastropubs
        score 3.790579745780415
Category NAME Breweries
        score 3.741387481218945

User ID 33306 top-5 results
Category NAME Wine Bars
        score 9.941408672420396
Category NAME Gastropubs
        score 5.0889820210486825
Category NAME Irish
        score 3.985181080181816 6

User ID 46833 top-5 results
Category NAME Pubs
        score 9.63823465928592
Category NAME Wine Bars
        score 9.30620820198328
Category NAME Breweries
        score 6.392883853401759
Category NAME Latin American
        score 5.172049917331964
```

SVD++(n_factors = 100, n_epoch = 20)

```
algo = algo1
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-4-1.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-4-3.txt

```
User ID 20384 top-5 results
Category NAME Vegetarian
        score 4.535270722162687
Category NAME Breweries
        score 3.9676834469808293
Category NAME Gastropubs
        score 3.5995410871497575

User ID 33306 top-5 results
Category NAME Wine Bars
        score 9.420302457767328
Category NAME Gastropubs
        score 4.963949110722694
Category NAME Irish
        score 3.987412217366547

User ID 46833 top-5 results
Category NAME Pubs
        score 7.378533150468922
Category NAME Wine Bars
        score 6.736808750819844
Category NAME Latin American
        score 5.623835733646488
Category NAME Breweries
        score 4.777974297083978
```

SVD++(n_factors = 200, n_epoch = 20)

```
algo = algo2
algo.fit(trainset)
results = get_top_n(algo, testset, uid_list, n=5, user_based=True)
with open('3-4-2.txt', 'w') as f:
    for uid, ratings in sorted(results.items(), key=lambda x: x[0]):
        f.write('User ID %s top-5 results\n' % uid)
        for cname, score in ratings:
            f.write('Category NAME %s\n\tscore %s\n' % (cname, str(score)))
        f.write('\n')
```

3-4-4.txt

```
User ID 20384 top-5 results
Category NAME Vegetarian
        score 5.9184758088005305
Category NAME Gastropubs
        score 4.700591344179813
Category NAME Breweries
        score 3.927235057114231

User ID 33306 top-5 results
Category NAME Wine Bars
        score 8.62748551394412
Category NAME Irish
        score 5.248229483696523
Category NAME Gastropubs
        score 4.979251054248358

User ID 46833 top-5 results
Category NAME Pubs
        score 8.499214512233433
Category NAME Wine Bars
        score 7.789780040695504
Category NAME Latin American
        score 5.499225726388636
Category NAME Breweries
        score 5.003254681127903
```

Requirement 3-4-5
Matrix-based Recommendation Best Model

```
svd_distributions = {'n_epochs' : range(100), 'n_factors' : range(200), 'biased' : ['False', 'True']}
svdRs = surprise.model_selection.search.RandomizedSearchCV(SVD, svd_distributions, random_state=0)
svdRs.fit(data)

print('SVD')
print(svdRs.best_score['rmse'])
print(svdRs.best_params['rmse'], '\n')

svdpp_distributions = {'n_epochs' : range(100), 'n_factors' : range(200)}
svdppRs = surprise.model_selection.search.RandomizedSearchCV(SVDpp, svdpp_distributions, random_state=0)
svdppRs.fit(data)

print('SVD++')
print(svdppRs.best_score['rmse'])
print(svdppRs.best_params['rmse'], '\n')

nmf_distributions = {'n_epochs' : range(100), 'n_factors' : range(200), 'biased' : ['False', 'True']}
nmfRs = surprise.model_selection.search.RandomizedSearchCV(NMF, nmf_distributions, random_state=0)
nmfRs.fit(data)

print('NMF')
print(nmfRs.best_score['rmse'])
print(nmfRs.best_params['rmse'], '\n')
```

SVD, SVD++, NMF에 대해 RandomizedSearchCV 실행

```
SVD
7.32311378956239
{'n_epochs': 23, 'n_factors': 93, 'biased': 'False'}

SVD++
7.278472075825469
{'n_epochs': 84, 'n_factors': 74}

NMF
7.301872829440432
{'n_epochs': 15, 'n_factors': 11, 'biased': 'False'}
```