Project 2.
DB mining & Automated Recommendation System

TEAM 06 김세찬 오재원 이승아 표승희 홍지현

Part 1. Decision Tree

1. DT induction에 사용될 Training DB 생성

분류	세부 정보	null 값 여부
id	아이템의 id	X
ratings	아이템이 사용자들에게 받은 평가 점수	O
num_of_specs	아이템이 가진 스펙의 수	O
num_of_tags	아이템에 붙어 있는 태그의 수	O
num_of_users	아이템을 이용한 이력이 있는 사용자의 수	X
avg_usage_time	아이템을 이용한 사용자들의 인당 평균 이용시간	O
num_of_reviews	아이템에 작성된 리뷰의 수	O
sum_of_recommend	아이템에 작성된 리뷰의 recoomend 총합	O
avg_review_len	아이템에 작성된 리뷰의 평균 본문 길이	O
best_item	**아이템의 BEST item 선정 여부**	**X**

- 총 item 개수 : 9192개
- 각 column의 null값 여부 : item_id로 group by 시행 후 row 수와 총 item 개수와 비교

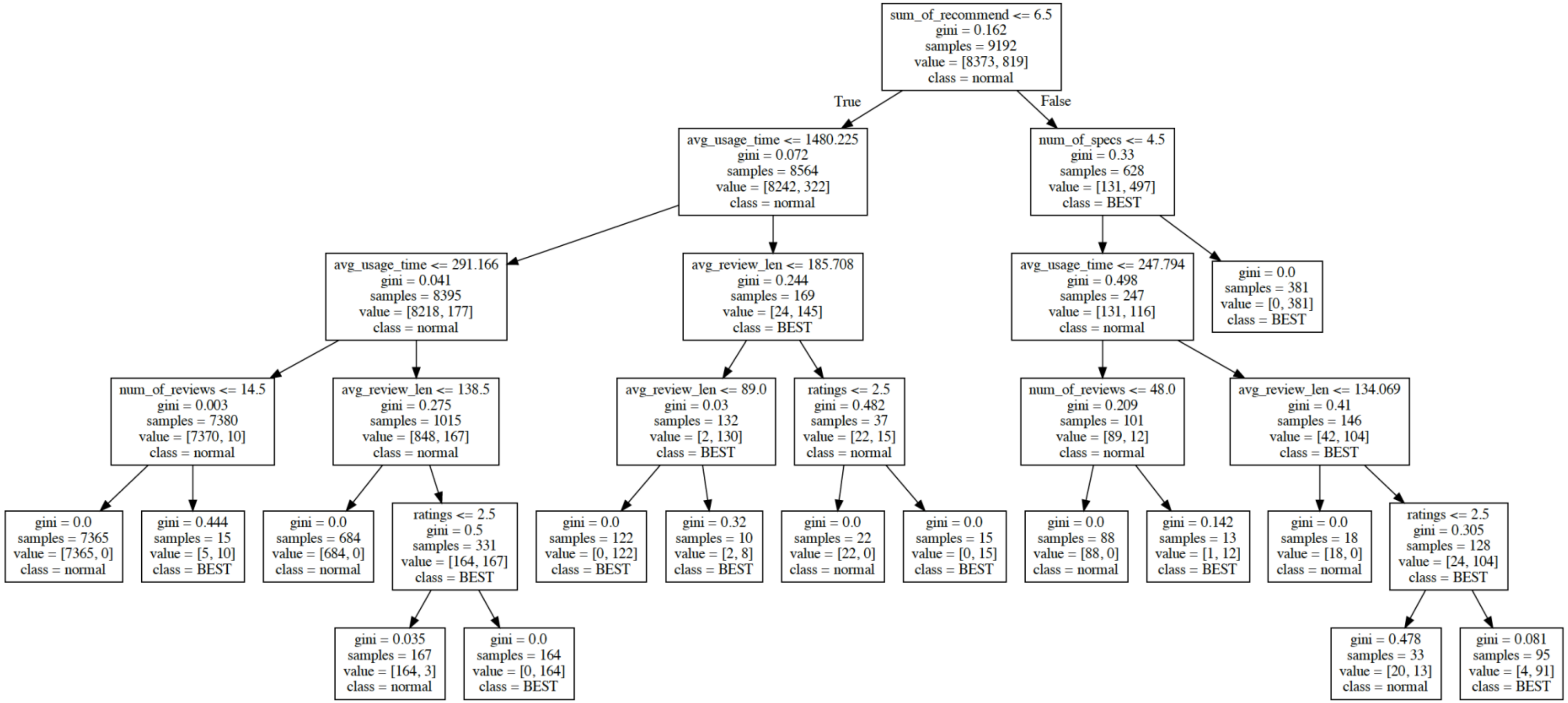
1. DT induction에 사용될 Training DB 생성

```
cursor.execute('''
SELECT id, best_item, ratings, num_of_specs, num_of_tags ,num_of_users, avg_usage_time,
num_of_reviews, sum_of_recommend, avg_review_len
FROM item
    NATURAL JOIN (SELECT item.id, COUNT(item_specs.spec_name) AS num_of_specs
        FROM item LEFT JOIN item_specs ON item.id=item_specs.item_id GROUP BY id)a
    NATURAL JOIN (SELECT item.id, COUNT(tag.tag_order) AS num_of_tags
        FROM item LEFT JOIN tag ON item.id=tag.item_id GROUP BY id)b
    NATURAL JOIN (SELECT item.id, COUNT(user_item.user_id) AS num_of_users
        FROM item LEFT JOIN user_item ON item.id=user_item.item_id GROUP BY id)c
    NATURAL JOIN (SELECT item.id, AVG(user_item.usagetime_total AS avg_usage_time
        FROM item LEFT JOIN user_item ON item.id = user_item.item_id GROUP BY id)d
    NATURAL JOIN (SELECT item.id, COUNT(review.id) AS num_of_reviews
        FROM item LEFT JOIN review ON item.id=review.item_id GROUP BY id)e
    NATURAL JOIN (SELECT item.id, SUM(review.recommend) AS sum_of_recommend
        FROM item LEFT JOIN review ON item.id=review.item_id GROUP BY id)f
    NATURAL JOIN (SELECT item.id, AVG(review.body AS avg_review_len
        FROM item LEFT JOIN review ON item.id=review.item_id GROUP BY id)g
''')
```

- Null 값 존재 column : item table과 left join한 후 다시 natural join
- 모든 item_id에 대해 training DB가 생성되도록 함.

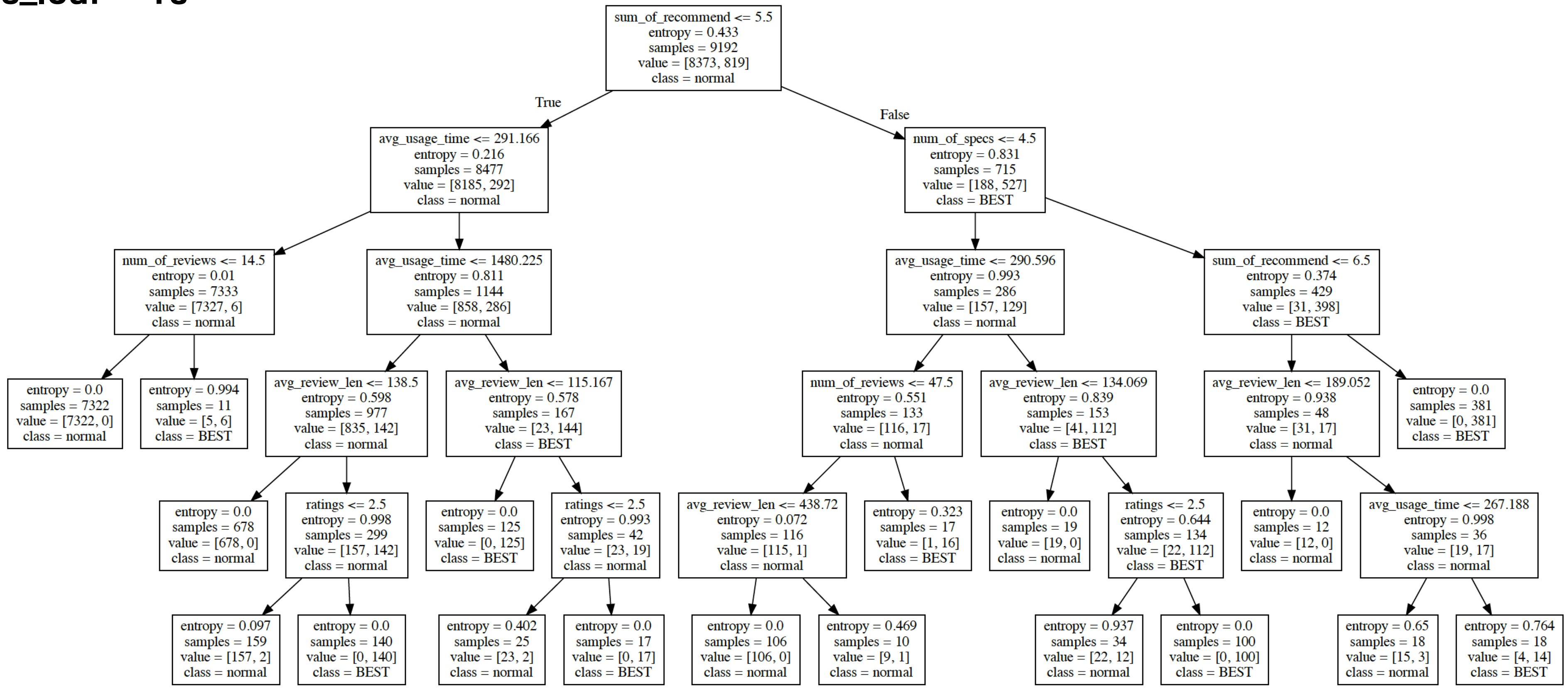
2. DT Induction (Criterion : ‘gini’)

- input features : training DB에서 id column 제외한 모든 column
- max_depth : 5
- mean_samples_leaf = 10



2. DT Induction (Criterion : ‘entropy’)

- input features : training DB에서 id column 제외한 모든 column
- max_depth : 5
- mean_samples_leaf = 10



3. 실험 : Input features, max_depth, mean_samples_leaf에 따른 mean_node_impurity 변화

1) 실험방법

- input features : training DB에서 id column 제외한 모든 column(8개)
- max_depth : 1~7
- mean_samples_leaf = 1,5,10
- Criterion : 'gini' , 'entropy'

경우의 수 :
2^8*7*3 = 5355가지

2) 실험 결과 분석 방법 :

- 각 실험에서 나오는 node별 impurity로부터 node당 sample 수를 고려해 mean_node_impurity 계산
- Mean node impurity에 대해 sorting 후 csv로 저장 & DT 시각화

3. 실험 : Input features, max_depth, mean_samples_leaf에 따른 mean_node_impurity 변화

3) 결과 : mean_node impurity로 내림차순 정렬한 data frame (gini)

```
df.sort_values(by='mean_gini')
```

	ratings	num_of_specs	num_of_tags	num_of_users	avg_usage_time	num_of_reviews	sum_of_recommend	avg_review_len	max_depth	min_samples_leaf	mean_gini	mean_entropy
4260	O	O	X	X	O	X	O	O	7	1	0.051856	0.234295
4261	O	O	X	X	O	X	O	O	7	5	0.052045	0.235752
4262	O	O	X	X	O	X	O	O	7	10	0.052099	0.236188
4239	O	O	X	X	O	X	O	X	7	1	0.056847	0.183421
4240	O	O	X	X	O	X	O	X	7	5	0.057031	0.183670
...	...	...	...	...	...	...	...	...	...	...	...	...
1343	X	O	X	X	X	X	X	X	7	10	0.166306	0.429674
1339	X	O	X	X	X	X	X	X	6	5	0.166341	0.428891
1342	X	O	X	X	X	X	X	X	7	5	0.166341	0.429727
1338	X	O	X	X	X	X	X	X	6	1	0.167144	0.428709
1341	X	O	X	X	X	X	X	X	7	1	0.167607	0.429518

5355 rows × 12 columns

3. 실험 : Input features, max_depth, mean_samples_leaf에 따른 mean_node_impurity 변화

3) 결과 : mean_node impurity로 내림차순 정렬한 data frame (entropy)

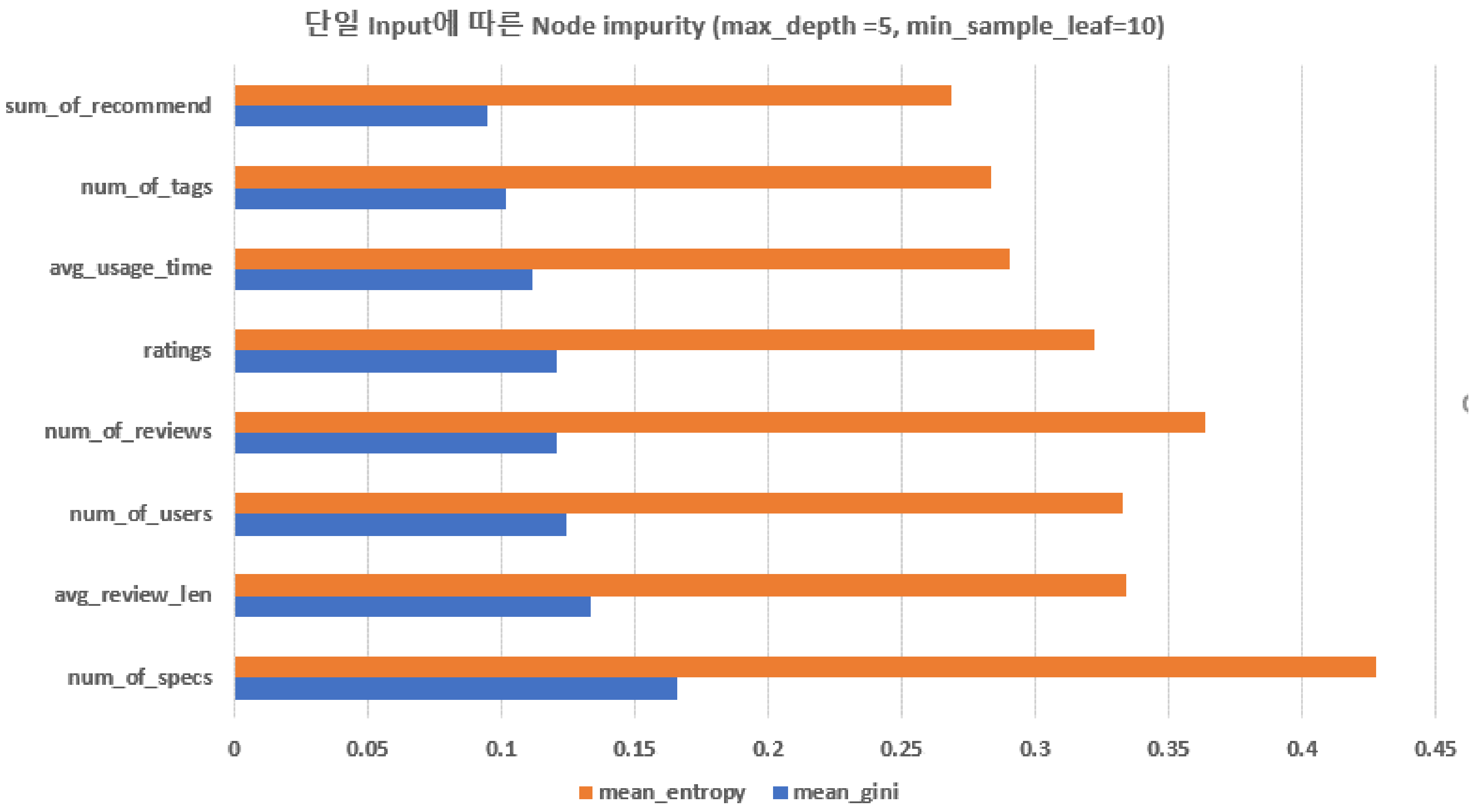
```
df.sort_values(by='mean_entropy')
```

	ratings	num_of_specs	num_of_tags	num_of_users	avg_usage_time	num_of_reviews	sum_of_recommend	avg_review_len	max_depth	min_samples_leaf	mean_gini	mean_entropy
4236	O	O	X	X	O	X	O	X	6	1	0.061852	0.180551
4237	O	O	X	X	O	X	O	X	6	5	0.062026	0.181058
4238	O	O	X	X	O	X	O	X	6	10	0.062372	0.182171
4233	O	O	X	X	O	X	O	X	5	1	0.068732	0.183065
4234	O	O	X	X	O	X	O	X	5	5	0.068926	0.183181
...	...	...	...	...	...	...	...	...	...	...	...	...
1340	X	O	X	X	X	X	X	X	6	10	0.166306	0.428838
1339	X	O	X	X	X	X	X	X	6	5	0.166341	0.428891
1341	X	O	X	X	X	X	X	X	7	1	0.167607	0.429518
1343	X	O	X	X	X	X	X	X	7	10	0.166306	0.429674
1342	X	O	X	X	X	X	X	X	7	5	0.166341	0.429727

5355 rows × 12 columns

3. 실험 : Input features, max_depth, mean_samples_leaf에 따른 mean_node_impurity 변화

4) 고찰



- Input feature에 따라서 data의 feature를 더 잘 설명하는 input이 있음을 확인

3. 실험 : Input features, max_depth, mean_samples_leaf에 따른 mean_node_impurity 변화

4) 고찰

ratings	num_of_specs	num_of_tags	num_of_users	avg_usage_time	num_of_reviews	sum_of_recommend	avg_review_len	mean_gini	mean_entr	label(등수)
X	X	X	X	O	X	O	X	0.075594	0.199016	30
X	O	X	X	X	X	O	X	0.092602	0.262056	1379
O	X	X	X	X	X	O	X	0.096439	0.266511	1758
X	X	X	O	X	X	O	X	0.096592	0.267733	1905
X	X	X	X	X	X	O	O	0.095645	0.268085	1920
X	X	X	X	X	X	O	X	0.094813	0.268642	1938
X	X	O	X	X	X	O	X	0.096551	0.27008	2016
X	X	X	X	X	O	O	X	0.09697	0.271794	2089

- 여러 개의 input feature를 사용할 때, input feature 간의 상관관계에 따라 data를 더 잘 구별하거나 그렇지 못할 수 있음.

Part 2. Association Analysis

R2-1, R2-2 query

① bundle_score view 생성

```
CREATE VIEW bundle_score AS
    SELECT
        a.bundle_id,
        bundle_name,
        num_item,
        IFNULL(num_genre, 0) AS num_genre,
        num_user,
        (num_item + IFNULL(num_genre, 0) + num_user) AS score
    FROM
1       (SELECT
            id AS bundle_id, bundle_name
        FROM
            bundle) a
            NATURAL JOIN
2       (SELECT
            bundle_id, COUNT(item_id) * 100 AS num_item
        FROM
            bundle_item
        GROUP BY bundle_id) b
            NATURAL JOIN
3       (SELECT
            bundle_id, (COUNT(user_id) / COUNT(DISTINCT d.item_id)) AS num_user
        FROM
            bundle_item d, user_item e
        WHERE
            d.item_id = e.item_id
        GROUP BY bundle_id) c
4           LEFT JOIN
        (SELECT
            bundle_id, COUNT(DISTINCT genre_id) * 100 AS num_genre
        FROM
            bundle_genre
        GROUP BY bundle_id) f ON a.bundle_id = f.bundle_id
    ORDER BY score DESC
    LIMIT 30
```

② user_bundle_rating view 및 partial_user_bundle_rating view 생성

```
CREATE VIEW user_bundle_rating AS
    SELECT
        IFNULL(user_idA, user_idB) AS user,
        IFNULL(bundle_nameA, bundle_nameB) AS bundle,
        IFNULL(cnt_rec, 0) * 5 + IF(IFNULL(cnt_use, 0) < 5,
            IFNULL(cnt_use, 0),
            5) AS rating
    FROM
        (SELECT
            *
        FROM
1           (SELECT
            user_id AS user_idA,
                bundle_id AS bundle_idA,
                bundle_name AS bundle_nameA,
                COUNT(item_id) AS cnt_rec
        FROM
            bundle_score
        NATURAL JOIN bundle_item
        NATURAL JOIN review
        WHERE
            recommend = 1
        GROUP BY user_id , bundle_id) a
3       LEFT JOIN (SELECT
2           user_id AS user_idB,
                bundle_id AS bundle_idB,
                bundle_name AS bundle_nameB,
                COUNT(item_id) AS cnt_use
        FROM
            bundle_score
        NATURAL JOIN bundle_item
        NATURAL JOIN user_item
        GROUP BY user_id , bundle_id) b ON a.user_idA = b.user_idB
            AND a.bundle_idA = b.bundle_idB UNION SELECT
            *
        FROM
            (SELECT
            user_id AS user_idA,
                bundle_id AS bundle_idA,
                bundle_name AS bundle_nameA,
                COUNT(item_id) AS cnt_rec
        FROM
            bundle_score
        NATURAL JOIN bundle_item
        NATURAL JOIN review
        WHERE
            recommend = 1
        GROUP BY user_id , bundle_id) a
        RIGHT JOIN (SELECT
            user_id AS user_idB,
                bundle_id AS bundle_idB,
                bundle_name AS bundle_nameB,
                COUNT(item_id) AS cnt_use
        FROM
            bundle_score
        NATURAL JOIN bundle_item
        NATURAL JOIN user_item
        GROUP BY user_id , bundle_id) b ON a.user_idA = b.user_idB
            AND a.bundle_idA = b.bundle_idB) d;
```

```
CREATE VIEW partial_user_bundle_rating AS
SELECT
    *
FROM
    user_bundle_rating
        NATURAL JOIN
    (SELECT
        user
    FROM
        user_bundle_rating
    GROUP BY user
    HAVING COUNT(*) >= 20) a;
```

R2-3, R2-4 code

③ horizontal table 생성

```
cursor.execute('SELECT * FROM partial_user_bundle_rating')
  df = pd.DataFrame(cursor.fetchall())
  df.columns = cursor.column_names
  df = df.set_index('user')

  df['rating'] = df['rating'] / df['rating']
  hor_view = pd.pivot_table(df, index='user', columns='bundle', values='rating', fill_value=0)

  filename = 'DMA_project2_team06_part2_horizontal.pkl'
  hor_view.to_pickle(filename)
```

④ frequent itemset 및 association rule 생성

```
frequent_itemsets = apriori(hor_view, min_support = 0.35, use_colnames = True)
  rules = association_rules(frequent_itemsets, metric = 'lift', min_threshold = 2)
```

R2-4, 연관분석 결과

- 총 199,369개의 association rules이 생성됨
- Support, Confidence, Lift만을 고려하여 평가
- Support(0.35, 0.46), confidence(0.75, 1), lift(2.11, 2.32)
  -> lift는 rule간에 큰 차이를 보이지 않음
  -> support와 confidence를 비교하여 rule의 중요도 및 유용성을 평가

maximum support(0.46), maximum confidence(1)를 갖는 rule의 일부

	antecedents	consequents
0	'Sid Meiers Civilization V: Complete', 'Grand Theft Auto V & Great White Shark Cash Card'	'Grand Theft Auto V & Whale Shark Cash Card', 'Sid Meiers Civilization Anthology', 'Grand Theft Auto V & Megalodon Shark Cash Card'
1	'Grand Theft Auto V & Whale Shark Cash Card', 'Sid Meiers Civilization V: Complete', 'Grand Theft Auto V & Great White Shark Cash Card'	'Grand Theft Auto V & Whale Shark Cash Card', 'Sid Meiers Civilization Anthology', 'Grand Theft Auto V & Megalodon Shark Cash Card', 'Valve Complete Pack'
2	'Grand Theft Auto V & Whale Shark Cash Card', 'Sid Meiers Civilization V: Complete', 'Valve Complete Pack'	'Sid Meiers Civilization Anthology', 'Grand Theft Auto V & Great White Shark Cash Card'

Sid Meiers Civilization series
Grand Theft Auto V series
Valve Complete Pack

Part 3. Recommendation System

1. Get-top-n 함수

```
# PART 3:
# 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:
        # TODO: testset의 데이터 중에 user id가 id_list 안에 있는 데이터만 따로 testset_id로 저장
        # Hint: testset은 (user_id, bundle_name, default_rating)의 tuple을 요소로 갖는 list

        testset_id = []
        print("makes a testset_id list...")
        for i in testset:
            if i[0] in id_list:
                testset_id.append(i)
            else:
                pass

        predictions = algo.test(testset_id)
        print("make a prediction results...")

        # TODO: results는 user_id를 key로, [(bundle_name, estimated_rating)의 tuple이 모인 list]를 value로 갖는 dictionary
        for uid, bname, true_r, est, _ in predictions:
            results[uid].append((bname, est))
```

- get-top-n 함수 중 user-based=True 인 경우에 대한 코드
- User-based=False 인 경우에 대해서도 위와 같은 형식으로 requirement만족
- Sort (reverse=True), 인덱싱

2. User-based Recommendation

1) KNNBasic [Sim = ‘cosine’]

```
sim_options = {'name': 'cosine', 'user_based': True}
algo = surprise.prediction_algorithms.knns.KNNBasic(sim_options=sim_options)
```

- 결과값 (txt) 일부

```
User ID 7998746368 top-5 results
Bundle NAME Sid Meiers Civilization Anthology
        score 1.7021921479492306
Bundle NAME Grand Theft Auto V & Megalodon Shark Cash Card
        score 1.5008325540188168
Bundle NAME Grand Theft Auto V & Whale Shark Cash Card
        score 1.5008325540188168
Bundle NAME Grand Theft Auto V & Great White Shark Cash Card
        score 1.5008325540188168
Bundle NAME The Binding of Isaac : Rebirth + Afterbirth Bundle
        score 1.4934661745854763
```

- 2) KNNWithMeans [Sim = ‘pearson’] 에 대해서도 동일한 형식으로 진행

2. User-based Recommendation

3) Algorithm 실험과 Cross-validation

```python
print("Cross validation...")
np.random.seed(0)
kf = surprise.model_selection.split.KFold(n_splits=5, random_state=0)

KBs_sim_list = ['cosine', 'pearson', 'MSD', 'pearson_baseline']
KBs_mean_acc_list = []

for x in KBs_sim_list:
    acc = []
    sim_options = {'name': x, 'user_based': True}
    algo = surprise.KNNBasic(sim_options=sim_options)

    for i, (trainset, testset) in enumerate(kf.split(data)):
        algo.fit(trainset)
        predictions = algo.test(testset)
        acc.append(surprise.accuracy.rmse(predictions, verbose=True))
    A = np.mean(acc)
    KBs_mean_acc_list.append(A)
print("KNNBasic Done")
```

Algorithm	변경값	RMSE 평균
KNNBasic	cosine	1.0112
	pearson	1.0216
	MSD	1.0323
	pearson_baseline	0.9493
KNNWithMeans	cosine	1.0238
	pearson	1.0302
	MSD	1.0486
	pearson_baseline	0.9616
KNNBaseline (sim : pearson_ baseline)	n_epochs:5, reg_u:10, reg_i=5	0.9443
	n_epochs:10, reg_u:20, reg_i=15	0.9453
	n_epochs:20, reg_u:30, reg_i=20	0.9459

- 각 실험에 대한 5-fold의 RMSE 평균값 기준으로 Best Model 판별

3. Item-based Recommendation

- 앞선 User-based 코드와 형식 동일
- 단, user-based=False 로 설정

- 결과값 (txt) 일부

```
Bundle NAME Borderlands Triple Pack top-10 results
User ID 8027368512
              score 3.852004306804694
User ID 7961040696
              score 3.733905223103874
User ID 8095204217
              score 3.188582531780235
User ID 8070632131
              score 2.8401860927451734
User ID 8053431839
              score 2.692104697078125
User ID 8054433999
              score 2.599298601983482
User ID 8035567225
              score 2.5759587682060587
User ID 8036934637
              score 2.5256376165942007
User ID 8042119839
              score 2.3961460375643573
User ID 8057371706
              score 2.371123022025473
```

Algorithm 실험과 Cross-validation

Algorithm	변경값	RMSE 평균
KNNBasic	cosine	1.5914
	pearson	1.6232
	MSD	1.4236
	pearson_baseline	1.6791
KNNWithMeans	cosine	1.0794
	pearson	1.0241
	MSD	1.0552
	pearson_baseline	1.0522
KNNBaseline (sim : pearson_ Baseline)	n_epochs:5, reg_u:10, reg_i=5	1.0510
	n_epochs:10, reg_u:20, reg_i=15	1.0534
	n_epochs:20, reg_u:30, reg_i=20	1.0541

4. Matrix-factorization recommendation

1&2) SVD

```
algo = surprise.prediction_algorithms.matrix_factorization.SVD(n_factors=100, n_epochs=50, biased=False)
```

3&4) SVD++

```
algo = surprise.prediction_algorithms.matrix_factorization.SVDpp(n_factors=100, n_epochs=50)
```

- 결과물 : user-based recommendation과 동일한 형식의 txt

4. Matrix-factorization recommendation

3) Algorithm 실험과 Cross-validation

Algorithm	변경값	RMSE 평균
SVD	n_factors=100, n_epoch=50, biased=True	0.9395
	n_factors=100, n_epoch=50, biased=False	0.9466
	n_factors=200, n_epoch=100, biased=True	0.9335
	n_factors=200, n_epoch=100, biased=False	0.9443
SVDpp	n_factors=70, n_epoch=30	0.9405
	n_factors=100, n_epoch=50	0.9306
	n_factors=150, n_epoch=100	0.9278
	n_factors=150, n_epoch=150	0.9257
NMF	n_factors=100, n_epoch=50, biased=True	1.3224
	n_factors=100, n_epoch=50, biased=False	0.9286
	n_factors=200, n_epoch=100, biased=True	1.2989
	n_factors=200, n_epoch=100, biased=False	0.9434

- 앞선 실험과 동일한 방식

Project 2.

DB mining & Automated Recommendation System

TEAM 06 김세찬 오재원 이승아 표승희 홍지현