데이터관리와 분석 프로젝트 #2

2017-[illegible]417 김 우

2018-[illegible]668 나 식

2016-[illegible]111 이 서

2019-[illegible]170 한창진

본 프로젝트는 프로젝트 #1에서 구축한 DB와 추가 제공하는 데이터를 바탕으로 DB-마이닝 및 Automated Recommendation System 구현을 목적으로 한다. 프로젝트는 크게 세가지 부분으로 나누어진다. PART#1에서는 의사결정나무 그리며 PART#2에서는 연관분석을 진행하며 PART#3에서는 user-based, item-based, matrix-factorization 추천시스템을 구한다. 각 파트 구현을 위해서 설정한 코드와 함께 결과에 대한 간단한 설명하도 록 하겠다.

PART#1) 의사결정나무 만들기

R1-1)

이 Requirement에서는 프로젝트#1에는 없었던 새로운 자료를 추가해야 한다. 사이트 A에서 활동을 열심히 한 멘토들에게 pro mentor 직위를 부여한다. Pro_mentor.txt에는 이 직위를 받아야하는 mentor들의 id가 적혀 있다. 우선 mentor table에 pro_mentor 칼럼을 추가한다. 이 text file에 id가 있는 경우에는 mentor table의 pro_mentor 값을 1, 아니면 0을 설정한다.

```
39    cursor.execute('USE %s;' % SCHEMA)
40
41    # TODO: Requirement 1-1. MAKE pro_mentor column
42     cursor.execute('ALTER TABLE mentor ADD pro_mentor TINYINT(1) DEFAULT 0')
43
44    promentor = open("C:/Users/82102/Desktop/DMA_project2/pro_mentor_list.txt", 'r')
45    p = promentor.readlines()
46
47    for var in range(0, len(p)):
48        a = p[var]
49        pro = a.replace('\n', '')
50        cursor.execute('UPDATE mentor SET pro_mentor=1 WHERE id=\'%s\'' % (pro))
51
52    promentor.close()
```

Pro mentor의 값이 0아니면 1 이기 때문에 자료형은 TINYINT(1)로 설정하며 기본값은 0으로 설정한다. 그리고 텍스트파일에서 읽은 id들은 pro mentor 값을 1로 updated 해주면 된다. 이후 SQL Workbench에서 SELECT id FROM mentor WHERE pro_mentor=1에서 나온 값들을 txt의 값들과 비교하면서 검토해볼 수 있다.

R1-2)

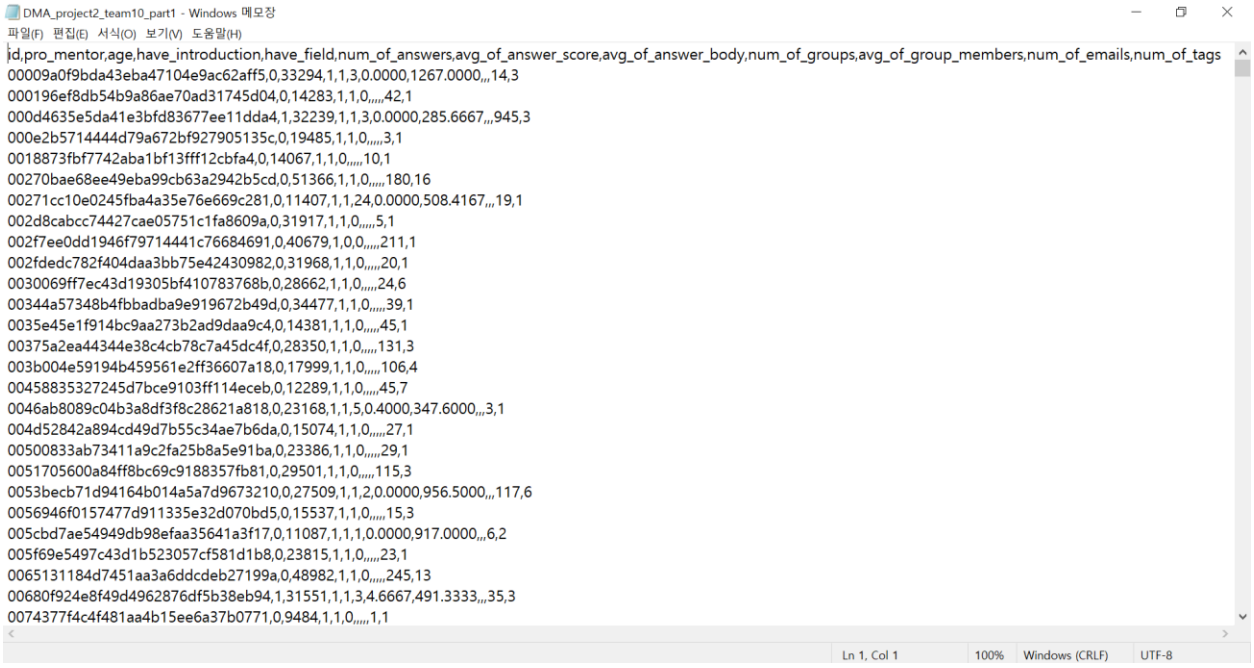
이 Requirement에서는 특정 칼럼들로 구성된 결과를 반환하여 이를 csv 파일로 저장해야 한다. 하나의 nested query를 이용하라는 조건이 있기 때문에 LEFT JOIN을 이용하여 다른 테이블에 있는 칼럼들을 모았다. 작성여부나 입력여부를 확인하기 위해서는 if(isnull(mentor.field),'0','1') 을 이용하며 평균을 구하기 위해서는 집단함수 AVG 개수를 구하기 위해서는 집단함수 COUNT를 이용하면 된다. LEFT JOIN은 부모테이블의 레코드는 출력하면서 자식 테이블의 레코드는 매칭된 것들만 출력되며 매칭이 되지 않은 레코드에는 NULL 값이 출력된다.

```
# TODO: Requirement 1-2. WRITE MYSQL QUERY AND EXECUTE. SAVE to .csv file

fopen = open('DMA_project2_team%02d_part1.csv' % team, 'w', encoding='utf-8')

cursor.execute('''
    SELECT id, pro_mentor, age, have_introduction,have_field,num_of_answers,avg_of_answer_score,avg_of_answer_body,
    num_of_groups, avg_of_group_members, num_of_emails, num_of_tags
    FROM (SELECT mentor.id AS id, mentor.pro_mentor AS pro_mentor,
        truncate(( unix_timestamp('2020-01-01 00:00:00') - unix_timestamp(mentor.joined_date))/3600,0) AS age,
        if(isnull(mentor.introduction),'0','1') AS have_introduction, if(isnull(mentor.field),'0','1') AS have_field,
        COUNT(answer.mentor_id) AS num_of_answers, AVG(answer.score) AS avg_of_answer_score, AVG(answer.body) AS avg_of_answer_body
        FROM mentor LEFT JOIN answer ON answer.mentor_id = mentor.id
        GROUP BY mentor.id) AS A1
    LEFT JOIN
        (SELECT mg.mentor AS id2, COUNT(mg.id) AS num_of_groups
        FROM mentoring_group AS mg GROUP BY mg.mentor) AS A2
    ON A1.id=A2.id2
    LEFT JOIN (SELECT id AS id3,  AVG(num_of_group_members) AS avg_of_group_members
        FROM (SELECT mg.mentor AS id, COUNT(mentee_id) AS num_of_group_members
                FROM group_membership AS gm RIGHT JOIN mentoring_group AS mg ON gm.group_id=mg.id GROUP BY gm.group_id) AS temp
        GROUP BY id
        ) AS A3
    ON A1.id =A3.id3
    LEFT JOIN (SELECT email.recipient_id, COUNT(*) AS num_of_emails FROM mentor LEFT JOIN email ON mentor.id = email.recipient_id GROUP
    ON A1.id = A4.recipient_id
    LEFT JOIN (SELECT tag_mentor.mentor_id, COUNT(*) AS num_of_tags FROM mentor LEFT JOIN tag_mentor ON mentor.id = tag_mentor.mentor_id
    ON A1.id = A5.mentor_id;
''')
```

이후 다음과 같은 소스코드로 결과를 csv로 변환해주면 된다.

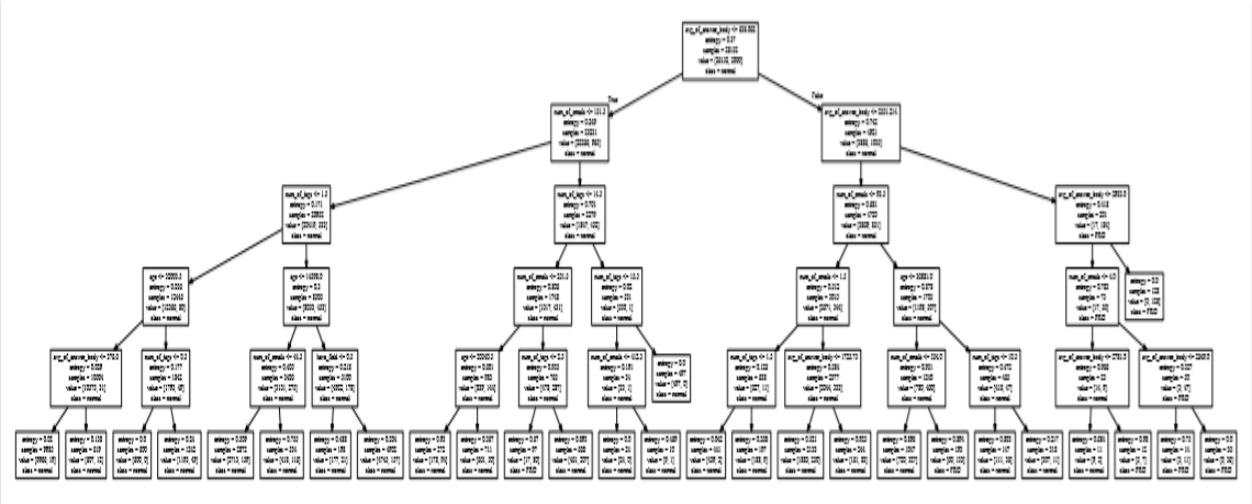
```
    rows = cursor.fetchall()
    column = [i[0] for i in cursor.description]
    w = csv.writer(fopen, lineterminator='\n')
    w.writerow(column)
    w.writerows(rows)
    fopen.close()
```

다음과 같이 DMA_project2_team10_part1이라는 csv 파일이 생성됨을 확인하였다.

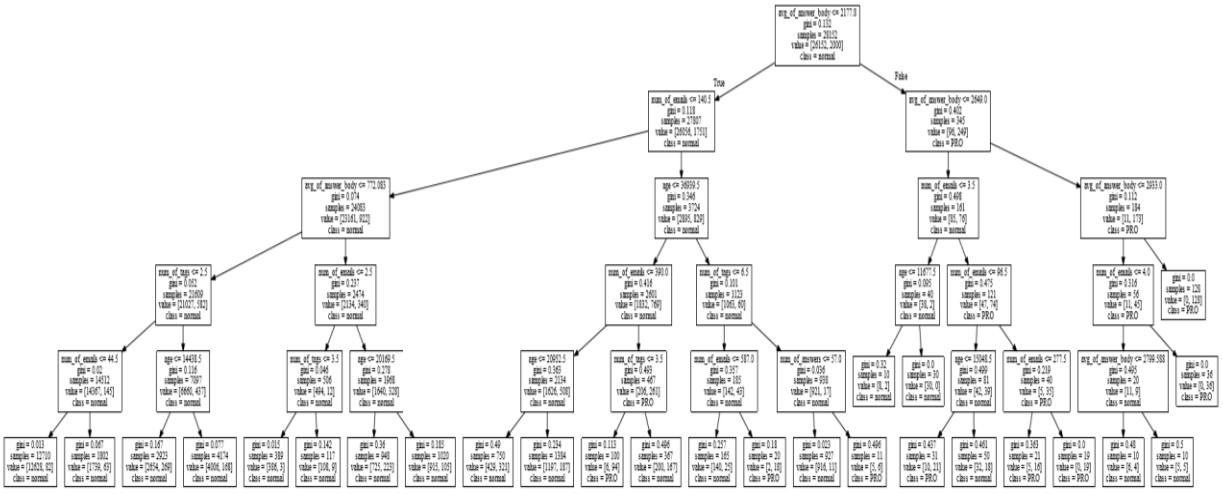


R1-3)

이 Requirement에서는 위에서 반환 받은 csv를 기준으로 의사결정나무를 생성해야 하며 그 결과를 graphiviz를 사용하여 저장해야 한다. Node impurity를 측정하는 두 가지 방식인 gini 와 entropy 두가지를 이용해서 만들어야야한다.



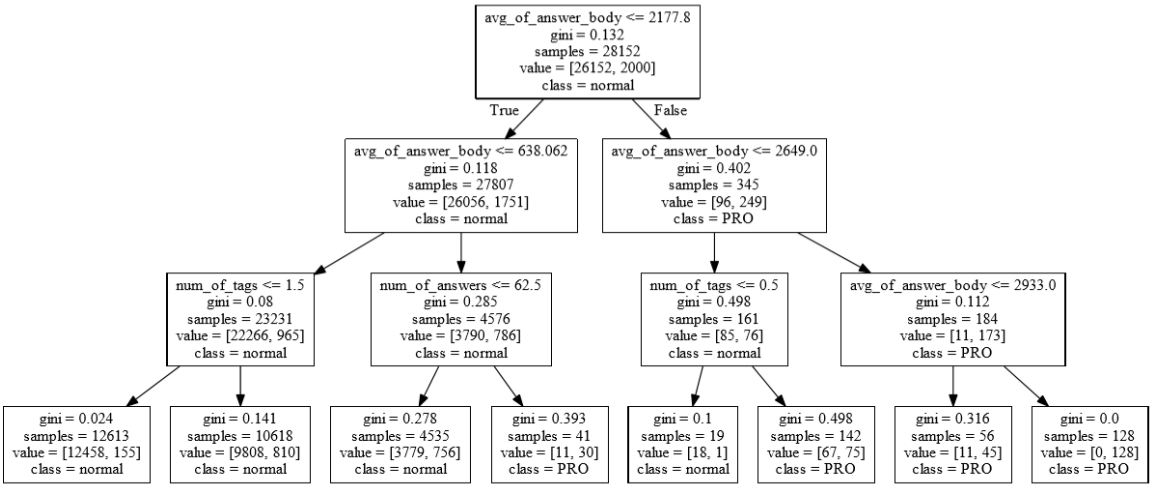
첫번째 의사결정나무의 경우 gini를 Node impurity 측정방식으로 선정하였다. 이 의사결정나무의 분석 목표는 PRO 멘토 선정 여부이다. min_samples_leaf:10, max_depth: 5로 설정되어있으며 feature names는 age, have_introduction, have_field, num_of_answers, avg_of_answer_score, avg_of_answer_body, num_of_groups, avg_of_group_members, num_of_emails, num_of_tags 가 사용되고 있다.



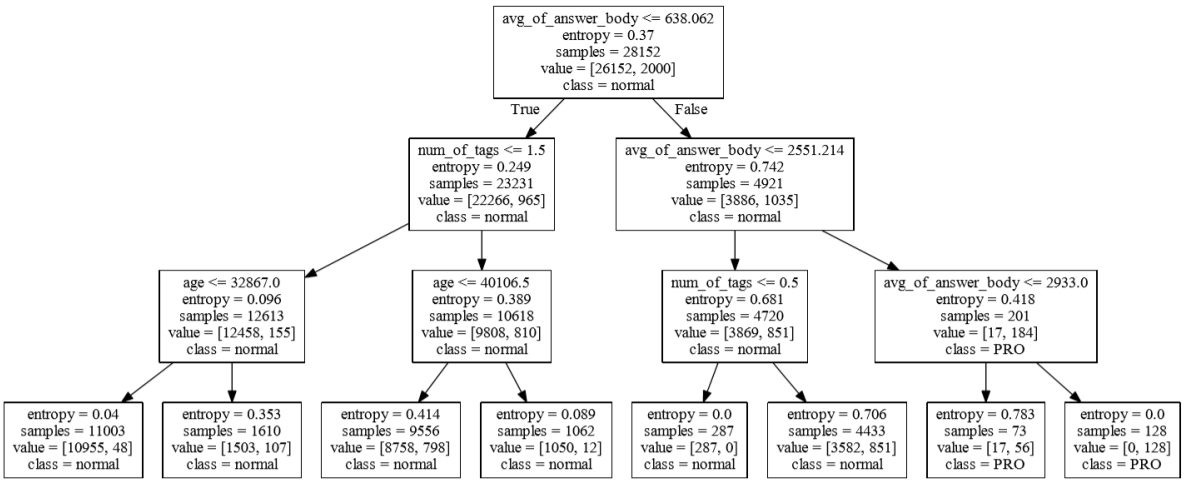
두번째 의사결정나무의 경우 entropy를 Node impurity 측정방식으로 선정하였다. 이 의사결정나무의 분석 목표는 PRO 멘토 선정 여부이다. min_samples_leaf:10, max_depth: 5로 설정되어있으며 feature names는 age, have_introduction, have_field, num_of_answers, avg_of_answer_score, avg_of_answer_body, num_of_groups, avg_of_group_members, num_of_emails, num_of_tags 가 사용되고 있다.

R1-4)

이 Requirement에서는 목표는 똑같이 PRO 멘토 선정 여부이지만 속성과 input feature가 R1-3)와는 다른 의사결정나무를 도출해보았다.



위의 의사결정나무는 gini를 Node impurity 측정방식으로 선정하였다. min_samples_leaf:7, max_depth: 3으로 설정되어있으며 feature names는 'age','have_field','num_of_answers', 'avg_of_answer_body', 'num_of_groups','avg_of_group_members','num_of_tags'로 줄어들었다.



이 의사결정나무는 entropy를 Node impurity 측정방식으로 선정하였다. min_samples_leaf:7, max_depth: 3으로 설정되어있으며 feature names는 'age','have_field','num_of_answers', 'avg_of_answer_body', 'num_of_groups','avg_of_group_members','num_of_tags'로 줄어들었다.

R1-3)에서 논의된 두 의사결정나무를 비교함으로써 node impurity 측정방식에 따라서 어떻게 결정나무가 달라지는 지 확인할 수 있었다. R1-4)는 R1-3)과 비교했을 때 min_samples와 max_depth가 줄어들었으며 feature names 중 몇몇 항목은 제외되었다. 이러한 차이가 의사결정나무에 어떻게 나타나는 지 확인해볼 수 있다. R1-4)의 두 의사결정나무를 구하기 위한 소스코드는 다음과 같다.

<소스코드>

```
# TODO: Requirement 1-3. MAKE AND SAVE DECISION TREE
# gini file name: DMA_project2_team##_part1_gini.pdf
# entropy file name: DMA_project2_team##_part1_entropy.pdf

data = pd.read_csv("DMA_project2_team10_part1.csv")
feature_names = ['age','have_field','num_of_answers',
            'avg_of_answer_body', 'num_of_groups','avg_of_group_members','num_of_tags']
label_name = ['pro_mentor']

data = data.fillna(0)

df = pd.DataFrame(data)
x = df[feature_names].values.tolist()
y = df[label_name].values.tolist()

DT1 = tree.DecisionTreeClassifier(criterion='gini', max_depth=3, min_samples_leaf=7)
DT1.fit(x, y)
DT2 = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=7)
DT2.fit(x, y)

graph = tree.export_graphviz(DT1, out_file=None,
                             feature_names=['age','have_field','num_of_answers',
            'avg_of_answer_body', 'num_of_groups','avg_of_group_members','num_of_tags'], class_names=['normal', 'PRO'])
graph = graphviz.Source(graph)
graph.render('DMA_project2_team10_part2_gini', view=True)

graph = tree.export_graphviz(DT2, out_file=None,
                             feature_names=['age','have_field','num_of_answers',
            'avg_of_answer_body', 'num_of_groups','avg_of_group_members','num_of_tags'], class_names=['normal', 'PRO'])
graph = graphviz.Source(graph)
graph.render('DMA_project2_team10_part2_entropy', view=True)
```

PART#2) TAG들간의 연관분석

R2-1) tag_score view 생성하기

```
# TODO: Requirement 2-1. CREATE VIEW AND SAVE to .csv file
cursor.execute('''
CREATE VIEW tag_score
AS SELECT id AS tag_id, name AS tag_name, num_mentor, num_mentee, num_question, num_mentor+num_mentee+num_question AS score
FROM tag,
(
SELECT tag_id, COUNT(*) AS num_mentor
FROM tag_mentor
GROUP BY tag_id
) AS mr,
(
SELECT tag_id, COUNT(*) AS num_mentee
FROM tag_mentee
GROUP BY tag_id
) AS me,
(
SELECT tag_id, COUNT(*) AS num_question
FROM tag_question
GROUP BY tag_id
) AS q
WHERE tag.id = mr.tag_id AND tag.id = me.tag_id AND tag.id = q.tag_id
ORDER BY score DESC
LIMIT 50;
''')

cursor.execute('SELECT * FROM tag_score;')
view = pd.DataFrame(cursor.fetchall())
view.columns = cursor.column_names
print('tag_score: \n', view)

# write a csv file
view.to_csv('DMA_project2_team%02d_part2_tag.csv' % team, sep=',', encoding='utf-8', index=False)
# ------
```

먼저 tag_score view를 생성하기에 앞서 'DMA_team10'이라는 이름의 데이터 스키마를 불러오고 위의 코드들을 실행하였다. tag_score view에는 tag_id, tag_name, num_mentor, num_mentee, num_question,

score가 저장되어야 하는데 이를 위해 DMA_team10에서 tag, tag_mentor, tag_mentee, tag_question table을 불러와 tag에서 tag_id, tag_name 의 값을 추출했으며, 나머지는 tag_id로 GROUP BY를 한 후에 COUNT 함수를 이용해 계산해주었다. 또, score는 COUNT로 계산된 값들의 합으로 나타내었다. 문제에서 score 를 기준으로 상위 50개만을 저장하라고 했으므로 'ORDER BY score DESC LIMIT 50' 구문을 활용해 view의 row들을 score로 내림차순 정렬을 한 뒤, 위에서부터 50개만 남도록 하였다. tag_score view를 저장하기 위해서 'SELECT * FROM tag_score' 라는 query를 이용해 불러와서 pandas.DataFrame 형식으로 변환한 뒤 내장함수인 to_csv 함수를 이용해 csv 파일로 저장하였다. 저장된 파일의 일부를 캡쳐한 화면은 아래와 같다. 의도했던대로 column의 정보들이 들어가 있는 것을 볼 수 있고 score 기준으로 내림차순 정렬된 것 또한 알 수 있다.

	A	B	C	D	E	F
1	tag_id	tag_name	num_mentor	num_mentee	num_questior	score
2	27490	college	1192	469	3728	5389
3	20141	telecommuni	3133	2	5	3140
4	27292	business	1054	211	986	2251
5	129	career	517	169	1560	2246
6	54	engineering	539	203	1081	1823
7	89	medicine	308	189	1323	1820
8	18073	technology	1160	138	500	1798
9	593	computer-sof	1361	49	213	1623
10	162	accounting	1047	67	314	1428
11	20227	college-majo	469	109	775	1353
12	18145	management	1091	46	195	1332
13	692	higher-educa	1006	22	263	1291
14	17972	marketing	936	84	247	1267
15	29121	hospital-and-	807	11	372	1190
16	18351	finance	638	79	468	1185
17	18360	psychology	304	198	679	1181
18	18217	doctor	29	195	946	1170
19	27510	information-t	938	42	123	1103
20	53	science	227	169	696	1092

R2-2) user_item_rating, partial_user_item_rating view 생성하기

```
# TODO: Requirement 2-2. CREATE 2 VIEWS AND SAVE partial one to .csv file
# User item rating view
cursor.execute('''
CREATE VIEW user_item_rating
AS SELECT user, item, SUM(cnt) AS rating
FROM (
    (
    SELECT tag_name AS item, mentor_id AS user, 5*COUNT(*) AS cnt
    FROM tag_mentor AS tm, tag_score AS ts
    WHERE tm.tag_id = ts.tag_id
    GROUP BY tag_name, mentor_id
    )
    UNION ALL
    (
    SELECT tag_name AS item, mentee_id AS user, 5*COUNT(*) AS cnt
    FROM tag_mentee AS tm, tag_score AS ts
    WHERE tm.tag_id = ts.tag_id
    GROUP BY tag_name, mentee_id
    )
    UNION ALL
    (
    SELECT tag_name AS item, mentor_id AS user, LEAST(5,COUNT(*)) AS cnt
    FROM answer AS a, tag_score AS ts, tag_question AS tq
    WHERE ts.tag_id = tq.tag_id AND tq.question_id = a.question_id
    GROUP BY tag_name, mentor_id
    )
    UNION ALL
    (
    SELECT tag_name AS item, mentee_id AS user, LEAST(5,COUNT(*)) AS cnt
    FROM question AS q, tag_score AS ts, tag_question AS tq
    WHERE ts.tag_id = tq.tag_id AND tq.question_id = q.id
    GROUP BY tag_name, mentee_id
    )
    ) AS uir
GROUP BY user, item;
```

각 사용자(mentor, mentee)가 tag에 대해 얼마나 관심을 가지고 있는지를 나타내는 user, item, rating을 컬럼으로 가지는 view를 생성했다. user(mentor와 mentee의 id)와 item(tag의 name) 정보는 각각 tag_mentor(or tag_mentee), tag_score에서 추출하였으며 rating을 계산하는 과정을 크게 두 과정으로 나누었다. rating을 계산하기 위해 부수적으로 필요한 user가 tag를 follow하고 있는지 여부, user가 질문 혹은 답변한 question이 tag를 태그한 개수를 알아내는 쿼리문을 분리시켜서 각각의 SELECT문으로 구현한 뒤 column name을 일치시켜서 UNION ALL을 하는 방식으로 하나의 view를 만들었다. 여기서 mentor와 mentee에 대해 위의 두 과정을 실행하게 되므로 결과적으로 위의 그림과 같이 총 4개의 SELECT 문이 합쳐지고 있는 것을 알 수 있다.

```
# Partial user item rating view
cursor.execute('''
CREATE VIEW partial_user_item_rating
AS SELECT user, item, rating
FROM user_item_rating
WHERE user IN (SELECT user FROM user_item_rating GROUP BY user HAVING COUNT(rating) > 3)
ORDER BY user;
''')

cursor.execute('SELECT * FROM partial_user_item_rating;')
view = pd.DataFrame(cursor.fetchall())
view.columns = cursor.column_names
print('partial_user_item_rating: \n', view)

# write a csv file
view.to_csv('DMA_project2_team%02d_part2_UIR.csv' % team, sep=',', encoding='utf-8', index=False)
# ------
cursor.close()
```

앞서 만들어진 view는 rating이 1 이상인 user-item set을 모두 포함하고 있는데 partial_user_item_rating 에는 4개 이상의 rating 정보를 가진 user에 대해서만 정보를 남기는 partial view를 저장해야 한다. 그러므로 'SELECT user FROM user_item_rating GROUP BY user HAVING COUNT(rating)>3' 구문을 이용해 rating의 수가 4개 이상인 user들을 추출한 뒤 이 user들에 대해서만 view가 생성될 수 있도록 WHERE를 이용해 조건을 설정해주었다. 이 또한 view가 생성된 뒤에는 위와 같은 방식으로 csv파일로 저장하였으며 다음은 파일의 일부를 캡쳐한 것이다. 원하던 것처럼 user, item, rating 정보가 잘 기록되어 있고 rating 값이 4개 이상 존재하는 user들만 남아있는 것을 확인할 수 있다.

	A	B	C
1	user	item	rating
2	00271cc10e0	college	4
3	00271cc10e0	career	2
4	00271cc10e0	art	5
5	00271cc10e0	higher-educa	1
6	00271cc10e0	career-counse	2
7	00271cc10e0	technology	1
8	0030069ff7e	telecommuni	5
9	0030069ff7e	tech	5
10	0030069ff7e	engineering	5
11	0030069ff7e	science	5
12	0030069ff7e	technology	5
13	0046ab8089	college	1
14	0046ab8089	marketing	1
15	0046ab8089	business	1
16	0046ab8089	management	1
17	0046ab8089	teacher	1
18	0046ab8089	computer	1
19	0046ab8089	finance	1
20	0065131184	business	5

R2-3). Horizontal table 생성하기

```
# TODO: Requirement 2-3. MAKE HORIZONTAL VIEW
df = view.copy()
df = pd.concat([df['user'], pd.get_dummies(df['item'])], axis=1)
hor_df = df.groupby(['user']).sum()
print('horizontal_view: \n', hor_df)

# file name: DMA_project2_team##_part2_horizontal.pkl
hor_df.to_pickle('DMA_project2_team%02d_part2_horizontal.pkl' % team)
# ------
```

partial_user_item_rating view가 저장되어 있는 dataframe을 horizontal table의 형태로 변환하기 위해 pandas의 get_dummies를 활용하였다. dataframe의 item column을 dummy 변수로 바꾸게 되면 item의 개수인 50개만큼 column이 생성되며 column name은 각 item의 이름이 된다. 그리고 원래 item의 값과 일치하는 이름의 column의 값만 1이고 나머지는 0으로 변환이 된다. 이를 dataframe의 user 컬럼과 concatenate(컬럼 기준)을 하게 되면 user는 중복값이 존재하고 각 row에는 하나의 item에만 1이 표시되는데 이를 groupby함수를 통해 sum을 해주게 되면 각 user 별로 어떤 item들을 rating하고 있는지 알려주는 horizontal table이 완성된다. 문제의 조건에 맞게 pickle 파일로 저장하는 코드도 위의 그림과 같이 구현되었다.

R2-4) 연관 분석 수행하기

```
# TODO: Requirement 2-4. ASSOCIATION ANALYSIS
# filename: DMA_project2_team##_part2_association.pkl (pandas dataframe)
frequent_itemsets = apriori(hor_df, min_support=0.01, use_colnames=True)
print('frequent_itemsets: \n', frequent_itemsets)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
print('association_rules: \n', rules)

# write a pickle file
rules.to_pickle('DMA_project2_team%02d_part2_association.pkl' % team)

#### Use to Assessment ####
# get a string from frozenset type
rules["antecedents"] = rules["antecedents"].apply(lambda x: list(x)).astype("unicode")
rules["consequents"] = rules["consequents"].apply(lambda x: list(x)).astype("unicode")
# write a csv file
rules.to_csv('DMA_project2_team%02d_part2_association.csv' % team, sep=',', index=False)
```

R2-3에서 만들어진 horizontal table을 가지고 문제에서 요구하는대로 mlxtend의 apriori 함수를 활용해 min_support가 0.01인 frequent itemset을 생성했다. 그리고 mlxtend의 association_rules 함수로 연관분석하는 코드를 구현하였으며 이 때 lif가 1 이상인 것들만 출력하도록 metric='lift', min_threshold=1로 설정해주었다. 그리고 연관분석 결과를 pickle 파일로 저장하였으며 이에 대한 정

성적, 정량적 평가를 수행하기 위해 보기 쉬운 csv 파일을 별도로 저장했다.

	A	B	C	D	E	F	G	H	I
1	antecedents	consequents	antecedent su	consequent s	support	confidence	lift	leverage	conviction
2	['engineering', 'telecommunications', 'tech']	['technology']	0.01078384	0.24271688	0.01078384	1	4.12002653	0.00816642	inf
3	['art', 'career-counseling', 'teacher']	['career', 'college']	0.01046194	0.24448737	0.01046194	1	4.09019092	0.00790412	inf
4	['career-choice', 'science', 'teacher']	['career', 'college']	0.01271527	0.24448737	0.01271527	1	4.09019092	0.00960655	inf
5	['career-choice', 'science', 'business', 'teacher']	['career', 'college']	0.01110575	0.24448737	0.01110575	1	4.09019092	0.00839053	inf
6	['career-counseling', 'higher-education', 'doctor', 'business']	['career', 'college']	0.01014003	0.24448737	0.01014003	1	4.09019092	0.00766092	inf
7	['doctor', 'career-counseling', 'science', 'business']	['career', 'college']	0.01191051	0.24448737	0.01191051	1	4.09019092	0.00899854	inf
8	['career-counseling', 'technology', 'business', 'teacher']	['career', 'college']	0.01062289	0.24448737	0.01062289	1	4.09019092	0.00802573	inf
9	['career-choice', 'career-counseling', 'science', 'teacher']	['career', 'college']	0.01110575	0.24448737	0.01110575	1	4.09019092	0.00839053	inf
10	['college-major', 'career-choice', 'science', 'teacher']	['career', 'college']	0.01062289	0.24448737	0.01062289	1	4.09019092	0.00802573	inf
11	['medicine', 'career-counseling', 'college-advice', 'doctor']	['career', 'college']	0.01030098	0.24448737	0.01030098	1	4.09019092	0.00778252	inf
12	['psychology', 'doctor', 'career-counseling', 'science']	['career', 'college']	0.01078384	0.24448737	0.01078384	1	4.09019092	0.00814733	inf
13	['medicine', 'career-counseling', 'psychology', 'teacher']	['career', 'college']	0.01014003	0.24448737	0.01014003	1	4.09019092	0.00766092	inf
14	['psychology', 'career-counseling', 'science', 'teacher']	['career', 'college']	0.01094479	0.24448737	0.01094479	1	4.09019092	0.00826893	inf
15	['college-major', 'psychology', 'science', 'teacher']	['career', 'college']	0.01046194	0.24448737	0.01046194	1	4.09019092	0.00790412	inf
16	['science', 'business', 'career-counseling', 'doctor', 'medicine']	['career', 'college']	0.01062289	0.24448737	0.01062289	1	4.09019092	0.00802573	inf
17	['education', 'career-counseling', 'accounting']	['career']	0.01078384	0.33735715	0.01078384	1	2.96421756	0.00714584	inf
18	['art', 'career-counseling', 'higher-education']	['career']	0.01014003	0.33735715	0.01014003	1	2.96421756	0.00671922	inf
19	['art', 'career-counseling', 'teacher']	['career']	0.01046194	0.33735715	0.01046194	1	2.96421756	0.00693253	inf
20	['engineering', 'career-choice', 'doctor']	['career']	0.01110575	0.33735715	0.01110575	1	2.96421756	0.00735914	inf
21	['career-choice', 'doctor', 'teacher']	['career']	0.01191051	0.33735715	0.01191051	1	2.96421756	0.00789241	inf

연관분석 결과가 저장된 csv 의 내용이다. 앞서 lift 가 1 이상인 결과들만 저장되도록 하였기 때문에 여기에 존재하는 목록은 선행사건(csv 의 1 열)과 후행사건(csv 의 2 열)이 독립이 아니고 밀접한 관련이 있음을 전제로 한다. 그렇기 때문에 선행사건이 일어났을 때 후행사건이 일어날 확률을 의미하는 confidence 값을 기준으로 정렬하여 데이터를 살펴보았다. 그 중 top20 을 캡쳐한 화면은 다음과 같은데 top20 의 목록은 모두 confidence 가 1 로 선행사건이 일어났을 때 후행사건이 무조건 일어났음을 의미한다. 즉, 예를 들어 총 rating 이 4 개 이상인 user 중에서 ['engineering', 'telecommunications', 'tech']에 rating 을 한 user 는 'technology' 라는 tag 도 함께 rating 을 하고 있음을 알 수 있다. 그리고 같은 confidence 값을 가지더라도 lift 를 이용해 내림차순 정렬되었기 때문에 상위에 위치할수록 선행사건과 후행사건이 독립적이지 않고 함께 일어날 확률이 큼을 의미한다. 따라서 위의 예시에서 3 번째 행에 있는 ['art', 'career-counseling', 'teacher'] tag set 을 rating 했을 때 ['career', 'college']를 rating 하고 있을 확률과 20 번째 행에 있는 ['engineering', 'career-choice', 'doctor']라는 tag set 을 rating 했을 경우 ['career']를 rating 하고 있을 확률은 100%로 동일하지만, lift 값을 비교해보면 전자의 경우가 4.09, 후자의 경우가 2.96 으로 전자의 경우가 더 연관성이 높다고 해석할 수 있다.

이 밖에도 연관분석 결과를 전체적으로 살펴보면서 'medicine'과 'doctor' 같이 관련있다고 생각되는 tag 들간의 confidence 는 높았고, 'college 와 'healthcare' 같이 관련이 없다고 생각되는 tag 들간의 confidence 가 낮은 것으로 보아 이 사이트의 mentor 와 mentee 들은 본인들의 일관된 관심사를 가지고 질문과 답변에 참여하고 있음을 짐작할 수 있었다.

PART#3) 추천시스템 구현

R3-1) get_top_n 함수 작성

이 Requirement에서는 점수 예측 결과 top-n개 결과를 반환하는 get_top_n 함수를 작성하였다. 이 함수는 이후 추천시스템을 학습시키기 위해서는 이 get_top_n 함수이 사용된다.

```python
# TODO: Requirement 3-1. WRITE get_top_n
def get_top_n(algo, testset, id_list, n=10, 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, item_id, default_rating)의 tuple을 요소로 갖는 list
        testset_id = []
        for i in range(len(testset)):
            if testset[i][0] in id_list:
                testset_id.append(testset[i])

        predictions = algo.test(testset_id)
        for uid, iid, true_r, est, _ in predictions:
            results[uid].append((iid, est))
            # TODO: results는 user_id를 key로,  [(item_id, estimated_rating)의 tuple이 모인 list]를 value로 갖는 dictionary
    else:
        # TODO: testset의 데이터 중 item id가 id_list 안에 있는 데이터만 따로 testset_id라는 list로 저장
        # Hint: testset은 (user_id, item_id, default_rating)의 tuple을 요소로 갖는 list
        testset_id = []
        for i in range(len(testset)):
            if testset[i][1] in id_list:
                testset_id.append(testset[i])

        predictions = algo.test(testset_id)
        for uid, iid, true_r, est, _ in predictions:
            results[iid].append((uid, est))
            # TODO - results는 item_id를 key로, [(user_id, estimated_rating)의 tuple이 모인 list]를 value로 갖는 dictionary(3점)

    for id_, ratings in results.items():
        # TODO: rating 순서대로 정렬하고 top-n개만 유지
        results[id_] = sorted(ratings, key=lambda x: x[1], reverse=True)[:n]

    return results
```

Permissions: RW End-of-lines: LF Encoding: UTF-8-BOM Line: 164 Column: 74 Memory: 81 %

R3-2) User-based 추천시스템

우선 두가지 알고리즘을 통하여 추천 결과를 얻었다. 첫번째 surprise.KNNBasic을 이용하였다. 이때 유사도는 cosine으로 k=40, min_k=1, sim_options={'name': 'cosine','user_based': True}, verbose=True로 지정하였다. 이를 통해서 출력된 추천 결과 top-10 item은 다음과 같다.

```
User ID ffca7b070c9d41e98eba01d23a920d52 top-10 results
Item ID telecommunications    score 4.875
Item ID leadership    score 4.85
Item ID sales         score 4.7
Item ID project-management    score 4.675
Item ID marketing-and-advertising         score 4.55
Item ID marketing     score 4.5
Item ID education-management          score 4.5
Item ID financial-services        score 4.225
Item ID accounting    score 4.05
Item ID tech          score 3.95

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID marketing-and-advertising         score 5.2
Item ID telecommunications    score 4.975
Item ID project-management    score 4.85
Item ID management            score 4.55
Item ID leadership    score 4.35
Item ID sales         score 4.1
Item ID programming           score 4.1
Item ID financial-services        score 3.95
Item ID education-management          score 3.925
Item ID design        score 3.625

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID education-management          score 5.075
Item ID telecommunications    score 5.05
Item ID leadership    score 4.85
Item ID project-management    score 4.75
Item ID management            score 4.675
Item ID sales         score 4.6
Item ID information-technology            score 4.175
Item ID marketing-and-advertising         score 4.175
Item ID computer-software     score 4.1
Item ID marketing     score 3.7
```

```
User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID sales         score 4.9
Item ID telecommunications    score 4.875
Item ID project-management    score 4.85
Item ID leadership    score 4.7
Item ID financial-services        score 4.7
Item ID education-management          score 4.575
Item ID marketing-and-advertising         score 4.55
Item ID marketing     score 4.375
Item ID music         score 4.2
Item ID computer-software     score 4.05

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID leadership    score 5.05
Item ID project-management    score 5.0
Item ID telecommunications    score 4.925
Item ID sales         score 4.9
Item ID education-management          score 4.75
Item ID management            score 4.725
Item ID financial-services        score 4.55
Item ID marketing     score 4.125
Item ID computer-software     score 4.05
Item ID tech          score 4.0
```

두번째 surprise.KNNWithMeans를 이용하였다. 이때 유사도는 pearson으로 k=40, min_k=1, sim_options={'name': 'pearson','user_based': True}, verbose=True로 지정하였다. 이를 통해서 출력된 추천 결과 top-10 item은 다음과 같다.

User ID ffca7b070c9d41e98eba01d23a920d52 top-10 results
Item ID telecommunications score 4.259262650848446
Item ID programming score 2.961351938502674
Item ID sales score 2.9218687848967226
Item ID leadership score 2.9169222019086023
Item ID management score 2.747600294479108
Item ID project-management score 2.6872658359840327
Item ID computer-science score 2.537698273948274
Item ID design score 2.4003294408336866
Item ID human-resources score 2.25360533871219
Item ID marketing score 2.106392059339026

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID career score 1
Item ID art score 1
Item ID technology score 1
Item ID college score 1
Item ID higher-education score 1
Item ID career-counseling score 1
Item ID telecommunications score 1
Item ID tech score 1
Item ID engineering score 1
Item ID management score 1

User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID telecommunications score 4.756719798421822
Item ID information-technology score 4.2405252575293195
Item ID project-management score 4.169412854156228
Item ID leadership score 4.068215307114138
Item ID computer-software score 3.7421583971583967
Item ID sales score 3.598395862353718
Item ID programming score 3.5397382203037697
Item ID technology score 3.3832370407370407
Item ID research score 3.3366117299068234
Item ID marketing score 3.260600996225996

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID telecommunications score 4.754242650247052
Item ID information-technology score 4.418526354060775
Item ID leadership score 4.322577180399227
Item ID project-management score 4.272842776016482
Item ID sales score 4.2603067360150835
Item ID marketing score 3.7588112764428554
Item ID human-resources score 3.7084760554468748
Item ID management score 3.52234264091172
Item ID marketing-and-advertising score 3.4830438617602084
Item ID technology score 3.413642643905802

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID telecommunications score 5.416929801057668
Item ID leadership score 5.111666598559274
Item ID project-management score 5.09217990386294
Item ID computer-software score 4.897648500351511
Item ID sales score 4.561746825697882
Item ID human-resources score 4.31849575311271
Item ID education-management score 4.285523478899825
Item ID marketing score 4.261742424242424
Item ID research score 4.225909485255836
Item ID management score 4.2133402014652015

이외에도 다양한 알고리즘과 유사도를 적용해보며 parameters = {'k': [20, 30, 40, 50], 'min_k': [1], 'sim_options':{'name': ['pearson','cosine'], 'user_based': [True]}}로 다른 설정에서 User-based 추천시스템을 적용시켜보았다. 다양한 조건속에서 진행해본 모델 중에서 cross validation(k=5, random_state=0)을 기준으로 가장 좋은 성능을 보이는 모델을 구하기 위해서 best_KNNBasic_score=grid_KNNBasic.best_score['RMSE'] 코드를 통해서 모델의 성능을 정량화 하였다. 가장 낮은 점수를 받은 모델을 선택하였더니 다음과 같은 결과가 나왔다.

```
The best UB algorithm is KNNWithMeans with {'k': 50,
'min_k': 1, 'sim_options': {'name': 'pearson',
'user_based': True}}
score: 2.1734439075876657
```

R3-3) Item-based 추천시스템

이번에도 두가지 알고리즘을 통하여 추천 결과를 얻었다. 첫번째 surprise.KNNBasic을 이용하였다. 이때 유사도는 cosine으로 k=40, min_k=1, sim_options={'name': 'cosine','user_based': False}, verbose=True로 지정하였다. 이를 통해서 출력된 추천 결과 top-10 item은 다음과 같다.

Item ID art top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.523189045845493
User ID b3bc740ed50849f1bb58afc0860106d4 score 8.020506217150732
User ID d8676c764b0f4f4fab4f2df45b950daf score 7.565842139692372
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 7.435213860337858
User ID aa53710fec054a8fb2e6eab6c605c571 score 7.2152419894896145
User ID 690e7e01008b472d839fb2ceb1823860 score 7.190116779444491
User ID de99f2c4d1314fc88bd2f972c6b8edfa score 7.158054129243542
User ID 0eefd053f1a34cf3a6bb8e7180c19ae3 score 7.128906526877622
User ID b1e7bc0aaf204a498954922e51bbd95e score 7.023143168010835
User ID 0d459058e7924332a623f39cfe37dad8 score 7.012367817056784

Item ID career top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.520350930050393
User ID d2172296954d41369ac4e563f379cdc6 score 8.347875979460136
User ID b3bc740ed50849f1bb58afc0860106d4 score 8.142895874973444
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 7.4489108045421135
User ID 8f44237ddad84f2184cfe4182fbf1105 score 7.18180857486745
User ID aa53710fec054a8fb2e6eab6c605c571 score 7.148524120660465
User ID dd3fe8a029dd49e9b59ae3f5f57198bc score 7.11357282217305
User ID 690e7e01008b472d839fb2ceb1823860 score 6.982386600963063
User ID b1e7bc0aaf204a498954922e51bbd95e score 6.971030636641553
User ID 3a3c6bcaffbc4d70a7166fc8091cc6c7 score 6.731843297622519

Item ID college top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.522055804534007
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 7.47842385124483
User ID b1e7bc0aaf204a498954922e51bbd95e score 6.997163709380776
User ID 87b78a895bfa4b099b816c42babbffaa score 6.759657306552603
User ID 3a3c6bcaffbc4d70a7166fc8091cc6c7 score 6.736071417200574
User ID 57a2e98e3b324c19aca1db63ee607c2a score 6.597131130692749
User ID 4dd5601d3d36403a8bf9d125207fbc6b score 6.529068034190792
User ID 29e16b94e8a441308b6fe5e6e5682a77 score 6.471190303382659
User ID a5a46cdc1fe04d3c9a6677e7f09b319d score 6.277279232465109
User ID 87c3a9df81054cae888283ac313aeb76 score 6.264594980593602

Item ID medicine top-10 results
User ID d2172296954d41369ac4e563f379cdc6 score 8.318903712593807
User ID b3bc740ed50849f1bb58afc0860106d4 score 7.945881673401732
User ID d8676c764b0f4f4fab4f2df45b950daf score 7.565153291785524
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 7.435213336971222
User ID 0eefd053f1a34cf3a6bb8e7180c19ae3 score 7.21511963939948
User ID de99f2c4d1314fc88bd2f972c6b8edfa score 7.210328306694787
User ID b1e7bc0aaf204a498954922e51bbd95e score 7.081734954933375
User ID eb2b5b36876b4658b302e345eb25c356 score 7.07092321647229
User ID 690e7e01008b472d839fb2ceb1823860 score 7.0698972371718005
User ID 8f44237ddad84f2184cfe4182fbf1105 score 6.991241164187791

Item ID teaching top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.469894208087906
User ID d2172296954d41369ac4e563f379cdc6 score 8.341515198910539
User ID b3bc740ed50849f1bb58afc0860106d4 score 8.016725214731274
User ID d8676c764b0f4f4fab4f2df45b950daf score 7.496864765087047
User ID aa53710fec054a8fb2e6eab6c605c571 score 7.226875496606871
User ID de99f2c4d1314fc88bd2f972c6b8edfa score 7.200957178597401
User ID 0eefd053f1a34cf3a6bb8e7180c19ae3 score 7.166936842081041
User ID 690e7e01008b472d839fb2ceb1823860 score 7.05828217306328
User ID eb2b5b36876b4658b302e345eb25c356 score 7.032005126026193
User ID 0d459058e7924332a623f39cfe37dad8 score 6.983063134822716

두번째 surprise.KNNWithMeans를 이용하였다. 이때 유사도는 pearson으로 k=40, min_k=1, sim_options={'name': 'pearson','user_based': True}, verbose=False로 지정하였다. 이를 통해서 출력된 추천 결과 top-10 item은 다음과 같다.

Item ID art top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.71497492069681
User ID 690e7e01008b472d839fb2ceb1823860 score 8.02893010678337
User ID aa53710fec054a8fb2e6eab6c605c571 score 7.761727731631334
User ID b0650c915d0246c0b55c5c6cbfdbb8b8 score 7.735386367488095
User ID d8676c764b0f4f4fab4f2df45b950daf score 7.5902002128926505
User ID b3bc740ed50849f1bb58afc0860106d4 score 7.330927508539462
User ID e6b4e36dacc94795a5be9c3c3d4ce4e3 score 7.25228369242369
User ID 362344cfbd954ae99a2d07a8767ad4a7 score 7.22983268312217
User ID dd3fe8a029dd49e9b59ae3f5f57198bc score 7.1286728373455475
User ID 1c98c8de1766419581beedf0a9ff6d82 score 7.117536724334267

Item ID career top-10 results
User ID 8f44237ddad84f2184cfe4182fbf1105 score 8.882713452269229
User ID 71d0d37706f5407fbc235409639910fe score 8.811413998071057
User ID b3bc740ed50849f1bb58afc0860106d4 score 8.503353032364835
User ID d2172296954d41369ac4e563f379cdc6 score 8.302021315930968
User ID dd3fe8a029dd49e9b59ae3f5f57198bc score 7.89831726063156
User ID 362344cfbd954ae99a2d07a8767ad4a7 score 7.752041294126675
User ID 87b78a895bfa4b099b816c42babbffaa score 7.700549245471746
User ID 77c41bec124e43cfb9ca8322d22062b7 score 7.551401643177299
User ID 4400843986c34259a55b2c239492bccf score 7.161627057029432
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 7.008537483701229

Item ID college top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 9.189715201729857
User ID 87b78a895bfa4b099b816c42babbffaa score 8.993481591777208
User ID 362344cfbd954ae99a2d07a8767ad4a7 score 8.30383208798608
User ID 35b40b1bcefd4d9e86e11a97603a84e5 score 8.28740912212664
User ID a5a46cdc1fe04d3c9a6677e7f09b319d score 7.293244588993948
User ID 429bd50a797444e7b4129d29e3f94a6d score 7.19528453502237
User ID cfbe7406a1ad4409a04c5bae691282e1 score 7.156021317145546
User ID 4dd5601d3d36403a8bf9d125207fbc6b score 7.121014542294067
User ID b1e7bc0aaf204a498954922e51bbd95e score 7.078897962545173
User ID 4d15e9292c7a4cb485cd4b16908e8406 score 7.035115811343937

Item ID medicine top-10 results
User ID b1e7bc0aaf204a498954922e51bbd95e score 8.503517285334143
User ID d2172296954d41369ac4e563f379cdc6 score 7.816901542882395
User ID 690e7e01008b472d839fb2ceb1823860 score 7.738366712808995
User ID c02bb26642334f1bafba216c8eda5466 score 7.650489166909762
User ID b0650c915d0246c0b55c5c6cbfdbb8b8 score 7.606453173858298
User ID d8676c764b0f4f4fab4f2df45b950daf score 7.569931546057857
User ID 4400843986c34259a55b2c239492bccf score 7.26360437490752
User ID eb2b5b36876b4658b302e345eb25c356 score 7.239828789324742
User ID 362344cfbd954ae99a2d07a8767ad4a7 score 7.105119851123984
User ID b3bc740ed50849f1bb58afc0860106d4 score 7.095984997961974

Item ID teaching top-10 results
User ID 71d0d37706f5407fbc235409639910fe score 8.523371096209015
User ID d2172296954d41369ac4e563f379cdc6 score 8.15732036662246
User ID aa53710fec054a8fb2e6eab6c605c571 score 7.854121500319138
User ID eb2b5b36876b4658b302e345eb25c356 score 7.537124145278039
User ID b3bc740ed50849f1bb58afc0860106d4 score 7.480058590743466
User ID 2c192c95a76a4a54b09e438755990099 score 7.257659151533566
User ID 0d459058e7924332a623f39cfe37dad8 score 7.063504171856117
User ID 0eefd053f1a34cf3a6bb8e7180c19ae3 score 6.942372730283807
User ID de99f2c4d1314fc88bd2f972c6b8edfa score 6.9342374350818385
User ID 690e7e01008b472d839fb2ceb1823860 score 6.88114343685498

이외에도 다양한 알고리즘과 유사도를 적용해보며 parameters = {'k': [20, 30, 40, 50], 'min_k': [1], 'sim_options':{'name': ['pearson','cosine'], 'user_based': [False]}}로 다른 설정에서 Item-based 추천시스템을 적용시켜보았다. 다양한 조건속에서 진행해본 모델 중에서 cross validation(k=5,random_state=0)을 기준으로 가장 좋은 성능을 보이는 모델을 구하기 위해best_KNNBasic_score=grid_KNNBasic.best_score['RMSE'] 코드를 통해서 모델의 성능을 정량화 하였다. 가장 낮은 점수를 받은 모델을 선택하였더니 다음과 같은 결과가 나왔다.

```
The best IB algorithm is KNNWithMeans with {'k': 30,
'min_k': 1, 'sim_options': {'name': 'cosine',
'user_based': False}}
score: 2.1278176365400743
```

R3-4) Matrix-based 추천시스템

Matrix-based 추천시스템도 두가지 알고리즘을 통하여 추천 결과를 얻었다. 첫번째 surprise.SVD를 이용하였다. 이때 n_factors=100, n_epoch=50, biased=False로 설정한 후 나온 추천결과와 n_factors= 200, n_epoch=100, biased=True를 통해서 출력된 추천 결과 모두 구하였다. 각각의 추천 결과가 표시한 top-10 item은 다음과 같다.

```
Item ID management          score 3.0402763495652874
Item ID sales          score 3.0222411709630466
Item ID information-technology          score 2.8177554977057575
Item ID telecommunications    score 2.493882349273159
Item ID higher-education          score 2.4075602192369394
Item ID programming          score 2.390917000187266
Item ID business       score 2.3619236255606744
Item ID technology  score 2.1515575191341907
Item ID human-resources          score 2.137901899668723

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID marketing-and-advertising          score 1.8178554909581925
Item ID human-resources          score 1.6593227075189714
Item ID programming          score 1.6531906582542435
Item ID telecommunications    score 1.4602999281843834
Item ID technology  score 1.4356314025690198
Item ID research       score 1.335237757943399
Item ID art          score 1.3151690958253512
Item ID project-management   score 1.2925213712036725
Item ID computer-science          score 1.2350055235584412
Item ID design          score 1.2332286601959197

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID finance          score 3.2910941032336876
Item ID project-management   score 3.232088883188552
Item ID management          score 3.041376971932814
Item ID leadership    score 2.8630700083816754
Item ID information-technology          score 2.8233632723749835
Item ID sales          score 2.669830512929059
Item ID human-resources          score 2.5359029106199227
Item ID education-management          score 2.3389132671902075
Item ID law          score 2.1508244957165603
Item ID design          score 2.1162430939155317

User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID research       score 4.65196388349847
Item ID project-management   score 4.163762478241737
Item ID programming          score 4.057627071091741
Item ID computer-software          score 3.9547408868156126
Item ID telecommunications    score 3.562789864586408
Item ID leadership    score 3.5320856577901627
Item ID information-technology          score 3.280691357934974
Item ID technology  score 3.081374957158157
Item ID computer    score 3.0446043337821775
Item ID music          score 3.010578648051938

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID education-management          score 4.300637334941364
Item ID telecommunications    score 3.610109006957589
Item ID leadership    score 3.3928568821530605
Item ID research       score 3.3005601049214675
Item ID technology  score 3.1050191288652096
Item ID sales          score 3.0499379819709693
Item ID higher-education          score 2.9876510336879267
Item ID education   score 2.9304215369443063
Item ID tech          score 2.9263874970294825
Item ID engineering score 2.9130257932966637
```

```
User ID ffca7b070c9d41e98eba01d23a920d52 top-10 results
Item ID telecommunications    score 4.830357646736512
Item ID education-management          score 4.767323714467736
Item ID project-management   score 4.574277265600208
Item ID sales          score 4.27216217288093
Item ID management          score 3.95111411412979
Item ID financial-services          score 3.9088508778644515
Item ID leadership    score 3.832409779412614
Item ID human-resources          score 3.7688183744590797
Item ID marketing-and-advertising          score 3.6761744877137064
Item ID information-technology          score 3.5909657658138774

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID telecommunications    score 4.670382207676345
Item ID leadership    score 4.037096080022971
Item ID sales          score 3.538222116699015
Item ID project-management   score 3.4671391123193342
Item ID education-management          score 3.380231303172518
Item ID technology  score 2.9332579026070507
Item ID marketing-and-advertising          score 2.8806539928895143
Item ID financial-services          score 2.826255267774536
Item ID tech          score 2.7945576188558947
Item ID business       score 2.7723635786739944

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID telecommunications    score 4.850205469559218
Item ID project-management   score 4.351206914231233
Item ID leadership    score 4.291847743345121
Item ID information-technology          score 4.165734318272288
Item ID sales          score 4.148839885935377
Item ID marketing   score 4.134206706969189
Item ID management          score 4.034227937896933
Item ID education-management          score 3.900213588666389
Item ID human-resources          score 3.7565084811773497
Item ID higher-education          score 3.4637856558840867
```

```
User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID telecommunications    score 5.156653521568762
Item ID project-management   score 4.3564705748930095
Item ID education-management          score 4.290096785143732
Item ID technology  score 4.230422304930327
Item ID leadership    score 4.125643153895717
Item ID financial-services          score 4.034683068844983
Item ID research       score 3.941560333850104
Item ID marketing-and-advertising          score 3.9312782988990183
Item ID computer-software          score 3.901933536342248
Item ID information-technology          score 3.831929767255668

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID telecommunications    score 5.050158877331383
Item ID project-management   score 4.139110442125716
Item ID information-technology          score 4.040147979527708
Item ID sales          score 4.03310351536179
Item ID leadership    score 3.870855380631077
Item ID financial-services          score 3.787005640074392
Item ID engineering score 3.7656203893507234
Item ID computer-software          score 3.7235326449463337
Item ID education-management          score 3.6462249858566307
Item ID computer-science          score 3.5928963778832173
```

두번째 surprise.SVDpp를 이용하였다. 이때 n_factors=100, n_epoch=50, biased=False로 설정한 후 나온 추천결과와 n_factors= 100, n_epoch=100, biased=True를 통해서 출력된 추천 결과 모두 구하였다.

각각의 추천 결과가 표시한 top-10 item은 다음과 같다.

User ID ffca7b070c9d41e98eba01d23a920d52 top-10 results
Item ID project-management score 4.765698984746647
Item ID telecommunications score 4.701021912354204
Item ID information-technology score 4.002689803949669
Item ID financial-services score 3.9546122279137603
Item ID education-management score 3.8973304037039087
Item ID marketing-and-advertising score 3.8698575469672045
Item ID management score 3.7869600650516615
Item ID leadership score 3.7329578888978308
Item ID sales score 3.654584188279742
Item ID research score 3.529872602012066

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID telecommunications score 4.103577981403025
Item ID financial-services score 3.616987668043452
Item ID education-management score 3.260392569807097
Item ID leadership score 2.9379065708069376
Item ID higher-education score 2.8774236178499453
Item ID programming score 2.6889042444180395
Item ID project-management score 2.6485105221064096
Item ID information-technology score 2.645955569103244
Item ID sales score 2.610144838692127
Item ID hospital-and-health-care score 2.588490142190933

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID telecommunications score 4.744939776213147
Item ID sales score 4.374278469597332
Item ID finance score 4.250098031556359
Item ID information-technology score 4.10244225083715
Item ID leadership score 4.070989380545023
Item ID project-management score 3.9832338796403075
Item ID human-resources score 3.380499292853248
Item ID management score 3.2542484682561024
Item ID music score 3.233063309006486
Item ID computer-software score 3.1151547053959234

User ID ffca7b070c9d41e98eba01d23a920d52 top-10 results
Item ID telecommunications score 4.86379413560544
Item ID sales score 4.263346834146196
Item ID marketing-and-advertising score 4.23114907525848
Item ID project-management score 4.098756072301645
Item ID education-management score 3.970036695084488
Item ID management score 3.8795409321352095
Item ID financial-services score 3.668346067437175
Item ID leadership score 3.6430268845129357
Item ID human-resources score 3.5103902450027498
Item ID law score 3.4703156079158726

User ID ffdb001850984ce69c5f91360ac16e9c top-10 results
Item ID telecommunications score 4.323037266866621
Item ID education-management score 3.9625762365565254
Item ID financial-services score 3.7321208584817613
Item ID leadership score 3.5021370910197867
Item ID accounting score 3.3372054043331736
Item ID project-management score 3.2115435492131796
Item ID college score 3.008677916622939
Item ID finance score 2.9970831036527015
Item ID marketing-and-advertising score 2.9681175273326095
Item ID higher-education score 2.6383829968007237

User ID ffdccaff893246519b64d76c3561d8c7 top-10 results
Item ID telecommunications score 4.83090029893996
Item ID information-technology score 4.610945367544074
Item ID project-management score 4.3533939780560775
Item ID sales score 4.2431288569912935
Item ID finance score 4.159432595537867
Item ID leadership score 4.0999976747730535
Item ID management score 3.764159940466835
Item ID marketing score 3.696285879316582
Item ID higher-education score 3.6315739980815835
Item ID human-resources score 3.629369857457773

User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID research score 6.325355218859508
Item ID project-management score 5.807166869637357
Item ID telecommunications score 4.986208604292946
Item ID leadership score 4.950912164207676
Item ID computer-software score 4.423320518523738
Item ID tech score 4.329735418296876
Item ID information-technology score 4.157817415038722
Item ID sales score 3.969894711170835
Item ID technology score 3.9098386763661077
Item ID programming score 3.8110409878505522

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID telecommunications score 4.833907250135994 5
Item ID project-management score 4.715100124306526
Item ID information-technology score 4.474923512683765
Item ID marketing score 4.016788104394517
Item ID leadership score 3.688308169020161
Item ID sales score 3.6599852275597358
Item ID hospital-and-health-care score 3.5414088160124404
Item ID career-choice score 3.515729976563308
Item ID psychology score 3.4105542045410067
Item ID higher-education score 3.3955170304109745

User ID ffe2f26d5c174e13b565d026e1d8c503 top-10 results
Item ID information-technology score 4.918937479095382
Item ID telecommunications score 4.617199665621844
Item ID project-management score 4.483432086202945
Item ID leadership score 4.453709459064759
Item ID research score 4.265228878412778
Item ID marketing score 4.2054758998558395
Item ID technology score 4.156616884355243
Item ID education-management score 4.075906972834643
Item ID programming score 4.034938819070734
Item ID sales score 3.855098541081275

User ID ffffbe8d854a4a5a8ab1a381224f5b80 top-10 results
Item ID telecommunications score 4.8752636996684595
Item ID project-management score 4.277673203962977
Item ID information-technology score 3.8953533793329242
Item ID financial-services score 3.8811043008494166
Item ID leadership score 3.8316498837849364
Item ID hospital-and-health-care score 3.6410989053395757
Item ID sales score 3.6213584452267025
Item ID computer-software score 3.5654598482814457
Item ID education-management score 3.5065595869711297
Item ID marketing score 3.3541601422365828

이외에도 다양한 알고리즘과 유사도를 적용해보며 parameters_SVD = {'n_factors': [50, 100, 200, 400], 'n_epochs':[10, 50, 100, 200], 'biased':[True, False]} 그리고 parameters_SVDpp = {'n_factors': [50, 100, 200, 400], 'n_epochs':[10, 50, 100, 200]}처럼 다른 설정에서 Matrix-based 추천시스템을 적용시켜보았다. SVD 와 SVDPP 각각의 알고리즘 중에서 cross validation(k=5, random_state=0)을 기준으로 가장 좋은 성능을 보이는 모델을 구하기 위해서 best_SVD_score = grid_SVD.best_score['RMSE'] 그리고 best_SVDpp_score = grid_SVDpp.best_score['RMSE'] 코드를 통해서 모델의 성능을 정량화 하였다. 가장 낮은 점수를 받은 SVD와 SVDPP 모델들을 선택하였더니 다음과 같은 결과가 나왔다.

```
The best MF algorithm is SVDpp with {'n_factors': 200,
'n_epochs': 50}
score: 2.012371384549619
```