협업필터링 기반 제주도 여행지 추천 서비스

김예지, 최혜미

[선정이유]
코로나 종식이 가까워지면서 여행산업이 활발해지고 있다. 팀원들 역시 여행을 좋아하고 자주 가기 때문에 여행지 추천 서비스를 만들면 좋겠다는 의견을 제시하였고 토의 끝에 주제로 선정하게 되었다. 그 중에서도 사람들이 자주 방문하는 여행지인 제주도를 기반으로 하는 여행지 추천 서비스를 만들기로 결정하였다.

[협업필터링]
많은 사용자로부터 얻은 취향 정보를 활용하는 기법이다. 협업 필터링 중에서도 유사한 사용자나 아이템을 이용할 것이므로, Memory-Based Approach 을 사용하였다. 협업필터링의 Memory-Based Approach 는 사용자기반(user-based)와 아이템기반(item-based)으로 다시 나눌 수 있다.

[사용언어]
Python, jupyter notebook

[협업 툴]
github

[github 주소]
https://github.com/Jeju-Travel-AITeam/jeju_recommendation_system

[레퍼런스 github 주소]
https://github.com/data-say/Datamining-Tour_Recommendation

[input data]

사용자에게 설문값을 받아 가장 만족할만한 여행지를 추천해준다. 추후 사용자에게 추천 받은 여행지에 대한 만족도 조사를 실행해 인공지능을 학습시켜 나갈 예정이다.

(사용자 설문지)

1. 성별을 입력해주세요.(남자 1/여자 2 입력)
2. 나이를 입력해주세요.(15~20=1, 21~30=2, 31~40=3, 41~50=4, 51~60=5, 61 이상=6)
3. 혼인 여부를 입력해주세요. (미혼 1/ 기혼 2 입력)
4. 몇번째 제주도 방문인가요?(1,2,3,4)
5. 여행의 주 목적이 무엇입니까? (1-11 까지 해당되는 번호 하나를 입력해주세요.)

1) 자연경관감상 2) 식도락(맛집여행) 3) 산/오름/올레길 트래픽
4) 쇼핑 5) 박물관/테마공원 방문 6) 영화/드라마촬영지 방문
7) 해변활동 8) 레포츠(승마, 골프, ATV 등) 9) 전통문화체험
10) 역사/문화유적지 방문 11) 종교/순례활동

5. 혼자 여행 하시나요?(아니면 1 맞으면 2 를 입력해주세요.)
->값이 1 인 경우 6 번 문항으로 이동 값이 2 인 경우 8 번으로 이동
6. 본인을 제외한 함께 여행할 일행의 수는 몇 명입니까?
7. 일행과의 관계를 골라주세요.(1-6 까지 해당되는 번호 하나를 입력해주세요.)

1) 친구/연인 2) (직장)동료
3) 단체모임 4) 비동거가족
5) 친척 6) 기타

(사용자 만족도 설문지)

1. 자연경관감상에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
2. 식도락(맛집여행)에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
3. 산/오름/올레길 트래픽에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
4. 쇼핑에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
5. 박물관/테마공원 방문에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
6. 영화/드라마촬영지 방문에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
7. 해변활동에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
8. 레포츠(승마, 골프, ATV 등)에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
9. 전통문화체험에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
10. 역사/문화유적지 방문에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
11. 종교/순례활동에 대한 만족도를 입력해주세요.(1 아주불만족-5 매우만족)
12. 여행추천 서비스에 대한 전반적 만족도에 대해 입력해주세요.
13. 재방문 의향이 있으신가요?(1 매우아님-5 아주많음)
14. 다른사람에게 추천할 의향이 있으신가요?(1 매우아님-5 아주많음)

[학습데이터]

(jeju_personal)

pid	sex	age	marry
100101	2	3	1
100102	1	4	1
100103	1	2	1
100104	2	2	2
100105	2	4	2

(jeju_place)

pid	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	q3_9	q3_10	q3_11
100101	1	3	3	3	2	1	1	2	2	2	2
100102	3	2	1	3	3	4	2	1	1	1	3
100103	5	1	3	3	1	3	2	2	1	2	2
100104	2	2	3	3	2	3	3	3	1	2	1
100105		2	1	2	3	3	4	2	3	1	2

(jeju_travel)

pid	count	q1	q2	q2_1	q2_2_1	q2_2_2	q2_2_3	q2_2_4	q2_2_5	q2_2_6
100101	4	60	1	1	1					
100102	1	174	2	0						
100103	1	77	1	4	1					
100104	1	70	1	4	1					
100105	1	37	1	42			3			

응답자 특성데이터인 jeju_personal, 방문지 만족 데이터인 jeju_place, 여행특성 데이터인 jeju_travel 세 데이터를 병합해 학습시켰다(6133 개의 데이터 학습)

+자세한 데이터 정보는 github 참고

데이터출처 : 제주특별자치도,「제주특별자치도방문관광객실태조사 - 2019 년 데이터 사용

https://kosis.kr/statHtml/statHtml.do?orgId=218&tblId=DT_21807N_A013&conn_path=I3

[유사도 측정]

유사도 파악은 추천해줄 대상의 선호도를 바탕으로 계산이 된다. 평균 제곱차이 유사도, 피어슨 유사도, 코사인 유사도를 이용해 사용자간의 유사도를 파악하였다.

(유사도 측정 기준)

```
act_data_pid.loc[0] = [1714257,4,5,0,1,2,3,2,3,1,0,0,60004]
act_data_pid.loc[1] = [1613413,4,0,1,0,3,0,1,1,0,2,0,77001]
act_data_pid.loc[2] = [1712377,1,1,5,2,2,3,2,0,2,4,0,57003]
```

	pid	act_nature	act_food	act_sports	act_shop	act_museum	act_movie	act_bitch	act_report	act_culture	act_history	act_religion	purpose_jplace
0	1714257	4	5	0	1	2	3	2	3	1	0	0	60004
1	1613413	4	0	1	0	3	0	1	1	0	2	0	77001
2	1712377	1	1	5	2	2	3	2	0	2	4	0	57003
3	100104	4	4	2	0	0	0	0	0	0	0	0	70008
4	100105	0	1	0	0	0	0	0	0	0	0	0	37005

pid	purpose_jplace
1714257	60004(식도락(맛집여행))
1613413	77001(자연경관감상)
1712377	57003(산/오름/올레길 트래픽)

+act_nature~act_religion 은 각 활동에 대한 만족도이다.

새로운 행에 pid(개개인의 일련번호)가 1714257, 1613413, 1712377 인 데이터를 생성해 임의의 만족도 값을 넣고, 해당 pid 의 purpose_jplace(여행 다녀온 장소) 데이터를 넣는다. 이 데이터들을 이용해 유사도 측정을 하였을 때 결과값이 purpose_jplace 와 유사하게 나오면 유사도가 높다고 판단하였다.

유사도 측정을 위해 평균 제곱차이 유사도, 피어슨 유사도, 코사인 유사도를 이용하였다. 위의 측정기준 값들을 넣어 각각 실행해 얻은 결과를 분석해봤을 때, 제주여행 추천 시스템은 사용자간의 유사도가 높다고 판단되었다.

평균제곱차이 유사도 (Mean Squared Difference, MSD)

: 유클리드 공간에서의 거리 제곱에 비례하는 값

$$\text{msd\_sim}(u, v) = \frac{1}{\text{msd}(u, v) + 1}$$
$$\text{msd\_sim}(i, j) = \frac{1}{\text{msd}(i, j) + 1}$$

- 사용자 u와 사용자 v간의 msd

$$\text{msd}(u, v) = \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r(u, i) - r_(v, i))^2$$

위 식에서 I_{uv}는 사용자 u와 사용자 v 모두에 의해 평가된 상품의 집합이고 $|I_{uv}|$는 사용자 u와 사용자 v 모두에 의해 평가된 상품의 수

- 상품 i와 상품 j간의 msd

$$\text{msd}(i, j) = \frac{1}{|U_{ij}|} \cdot \sum_{u \in U_{ij}} (r(u, i) - r(u, j))^2$$

위 식에서 U_{ij}는 상품 i와 상품 j 모두를 평가한 사용자의 집합이고 $|U_{ij}|$는 상품 i와 상품 j 모두를 평가한 사용자의 수

```
from collections import OrderedDict

def sim_msd(data, name1, name2):
    s = 0
    count = 0
    for i in data[data.pid == name1]:
        if i == "pid":
            continue
        s += pow(int(data[data.pid == name1][i]) - int(data[data.pid == name2][i]), 2)
        count += 1

    return 1 / (1 + (s / count))

def top_match(data, name, index = 10, sim_function = sim_msd):
    li = []
    for i in data["pid"]:
        if (name != i):
            li.append((sim_function(data, name, i), i))
    li.sort()
    li.reverse()

    c = []
    for i in range(index):
        if int(data.loc[data.pid == li[i][1], "purpose_jplace"]) is not
int(data.loc[data.pid == name, "purpose_jplace"]):
            c.append(int(data.loc[data.pid == li[i][1], "purpose_jplace"]))
            c = list(OrderedDict.fromkeys(c))

    return c


top_match(act_data_pid, 1714257))
top_match(act_data_pid, 1613413))
top_match(act_data_pid, 1712377))
```

```
out(1714257) : [60004]

out(1712377) : [57003]

out(1613413) : [77001]
```

코사인 유사도 (Cosine Similarity)

: 두 특성 벡터의 각도에 대한 코사인 값, 각도 θ 가 0 도이면 코사인 유사도는 1 이다. 반대로 각도 θ 가 90 도이면 코사인 유사도는 0 이다.

$$x \cdot y = |x||y| \cos\theta$$

$$\cos\theta = \frac{x \cdot y}{|x||y|}$$

- 사용자 u와 사용자 v간의 코사인 유사도

$$\text{cosine\_sim}(u, v) = \frac{\sum_{i \in I_{uv}} r(u,i) \cdot r(v,i)}{\sqrt{\sum_{i \in I_{uv}} r(u,i)^2} \cdot \sqrt{\sum_{i \in I_{uv}} r(v,i)^2}}$$

- 상품 i와 상품 j간의 코사인 유사도

$$\text{cosine\_sim}(i, j) = \frac{\sum_{u \in U_{ij}} r(u,i) \cdot r(u,j)}{\sqrt{\sum_{u \in U_{ij}} r(u,i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} r(u,j)^2}}$$

```
from math import sqrt
from collections import OrderedDict

def sim_cosine(data, name1, name2):
    sum_name1 = 0
    sum_name2 = 0
    sum_name1_name2 = 0

    for i in data[data.pid == name1]:
        if i == "pid":
            continue
        sum_name1 += pow(int(data[data.pid == name1][i]), 2)
        sum_name2 += pow(int(data[data.pid == name2][i]), 2)
        sum_name1_name2 += int(data[data.pid == name1][i]) * int(data[data.pid ==
name2][i])

    return sum_name1_name2 / (sqrt(sum_name1)*sqrt(sum_name2))

def top_match(data, name, index = 10, sim_function = sim_cosine):
    li = []
    for i in data["pid"]:
        if (name != i):
            li.append((sim_function(data, name, i), i))
    li.sort()
    li.reverse()

    c = []
    for i in range(index):
        if int(data.loc[data.pid == li[i][1], "purpose_jplace"]) is not
```

```
int(data.loc[data.pid == name, "purpose_jplace"]):
            c.append(int(data.loc[data.pid == li[i][1], "purpose_jplace"]))
            c = list(OrderedDict.fromkeys(c))

    return c

top_match(act_data_pid, 1714257))
top_match(act_data_pid, 1613413))
top_match(act_data_pid, 1712377))
```

```
out(1714257) : [60004, 37005, 7008]

out(1712377) : [57003, 70008]

out(1613413) : [77001]
```

피어슨 유사도 (Pearson Similarity)

: 두 벡터의 상관계수 (Pearson correlation coefficient)를 말하며 다음과 같이 정의

- 사용자 u와 사용자 v간의 msd

$$\text{pearson\_baseline\_sim}(u,v) = \hat{\rho}_{uv} = \frac{\sum_{i \in I_{uv}} (r(u,i) - b(u,i)) \cdot (r(v,i) - b(v,i))}{\sqrt{\sum_{i \in I_{uv}} (r(u,i) - b(u,i))^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r(v,i) - b(v,i))^2}}$$

- 상품 i와 상품 j간의 msd

$$\text{pearson\_baseline\_sim}(i,j) = \hat{\rho}_{ij} = \frac{\sum_{u \in U_{ij}} (r(u,i) - b(u,i)) \cdot (r(u,j) - b(u,j))}{\sqrt{\sum_{u \in U_{ij}} (r(u,i) - b(u,i))^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r(u,j) - b(u,j))^2}}$$

```
from collections import OrderedDict

def sim_pearson(data, name1, name2):
    avg_name1 = 0
    avg_name2 = 0
    count = 0
    for i in data[data.pid == name1]:
        if i == "pid":
            continue
        avg_name1 = int(data[data.pid == name1][i])
        avg_name2 = int(data[data.pid == name2][i])
        count += 1

    avg_name1 = avg_name1 / count
    avg_name2 = avg_name2 / count
```

```
    sum_name1 = 0
    sum_name2 = 0
    sum_name1_name2 = 0
    count = 0
    for i in data[data.pid == name1]:
        if i == "pid":
            continue
        sum_name1 += pow(int(data[data.pid == name1][i]) - avg_name1, 2)
        sum_name2 += pow(int(data[data.pid == name2][i]) - avg_name2, 2)
        sum_name1_name2 += (int(data[data.pid == name1][i]) - avg_name1) *
(int(data[data.pid == name2][i]) - avg_name2)

    return sum_name1_name2 / (sqrt(sum_name1)*sqrt(sum_name2))

def top_match(data, name, index = 10, sim_function = sim_pearson):
    li = []
    for i in data["pid"]:
        if (name != i):
            li.append((sim_function(data, name, i), i))
    li.sort()
    li.reverse()

    c = []
    for i in range(index):
        if int(data.loc[data.pid == li[i][1], "purpose_jplace"]) is not
int(data.loc[data.pid == name, "purpose_jplace"]):
            c.append(int(data.loc[data.pid == li[i][1], "purpose_jplace"]))
            c = list(OrderedDict.fromkeys(c))

    return c

top_match(act_data_pid, 1714257))
top_match(act_data_pid, 1613413))
top_match(act_data_pid, 1712377))
```

```
out(1714257) : [60004, 37005, 70008]

out(1722377) : [57003,70008]

out(1613413) : [77001]
```

[output data]

자연경관감상 (77)	우도(해양도립공원)(005) / 용두암(003) / 카멜리아힐 (001)
식도락(맛집여행) (60)	연돈(002) / 중문수원음식점(008) / 몽상드애월 (004)
산/오름/올레길 트래픽 (57)	성산일출봉(003) / 제주올레(006) / 새별오름 (009)
쇼핑 (80)	바이제주(011) / 선물고팡(020) / 모이소 (010)/지하상가(001)
박물관/테마공원 방문 (156)	제주김녕미로공원(015) / 아쿠아플라넷 제주(022) / 해녀박물관 (021)
영화/드라마촬영지 방문 (174)	오조포구(030) / 표선등대(004) / 물영아리오름 (034)
해변활동(해수욕 등) (37)	협재해수욕장(102) / 쇠소깍(802) / 함덕해수욕장 (005)
레포츠(승마, 골프, ATV 등) (70)	제주승마공원(008) / 나인브릿지(077) / 새별레져 ATV (079)
전통문화체험 (17)	제주민속촌(302) / 동문재래시장(202) / 김만덕객주 (006)
역사/문화유적지 방문 (104)	북촌국민학교(012) / 관덕정(017) / 항파두리항몽유적지 (008)
종교/순례활동 (90)	눈물의십자가(009) / 마라도기원정사(053) / 제주중앙성당 (050)

purpose	jplace	purpose_jplace
자연경관감상 (77)	우도(해양도립공원)(005)	77005
	용두암(003)	77003
	카멜리아힐 (001)	77001
식도락(맛집여행) (60)	연돈(002)	60002
	중문수원음식점(008)	60008
	몽상드애월 (004)	60004
산/오름/올레길 트래픽 (57)	성산일출봉(003)	57003
	제주올레(006)	57006
	새별오름 (009)	57009

쇼핑 (80)	바이제주(011)	80011
	선물고팡(020)	80020
	모이소 (010)	80010
	제주 지하상가 (001)	80001
박물관/테마공원 방문 (156)	제주김녕미로공원(015)	156015
	아쿠아플라넷 제주(022)	156022
	해녀박물관 (021)	156021
영화/드라마촬영지 방문 (174)	오조포구(030)	174030
	표선등대(004)	174004
	물영아리오름 (034)	174034
해변활동(해수욕 등) (37)	협재해수욕장(102)	37102
	쇠소깍(802)	37802
	함덕해수욕장 (005)	37005
레포츠(승마, 골프, ATV 등) (70)	제주승마공원(008)	70008
	나인브릿지(077)	70077
	새별레져 ATV (079)	70079
전통문화체험 (17)	제주민속촌(302)	17302
	동문재래시장(202)	17202
	김만덕객주 (006)	17006
역사/문화유적지 방문 (104)	북촌국민학교(012)	104012
	관덕정(017)	104017
	항파두리항몽유적지 (008)	104008
종교/순례활동 (90)	눈물의십자가(009)	90009
	마라도기원정사(053)	90053
	제주중앙성당 (050)	90050

```
data['purpose_jplace'] = data['purpose'].astype(int)*1000 +
data['jplace'].astype(int)
data.drop(labels = ['purpose'], axis = 1, inplace = True)
data.drop(labels = ['jplace'], axis = 1, inplace = True)
```

purpose 데이터 값과 jplace 의 데이터 값을 이용해 여행지인 purpose_jplace 를 출력한다. 출력된 purpose_jplace 의 값이 해당되는 장소를 최종 결과값으로 출력한다. 장소는 숫자로 표기되므로 위의 output data 표를 참고하여야한다.

(실제 출력결과)

카멜리아힐(77001)이 추천되었다는 것을 알 수 있다.