관광 데이터 분석을 통한

해외 여행지 추천 시스템

어해림 이세윤 최민주

목차

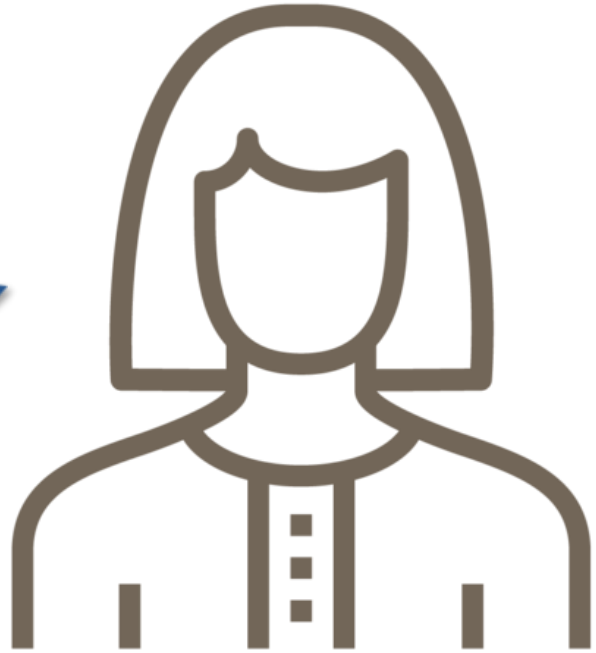
01

주제 선정 동기

시간도 많고 돈도 있는데.. 어디로 여행가지?

아무리 포털 사이트 검색을 해보아도..

01 주제 선정 동기

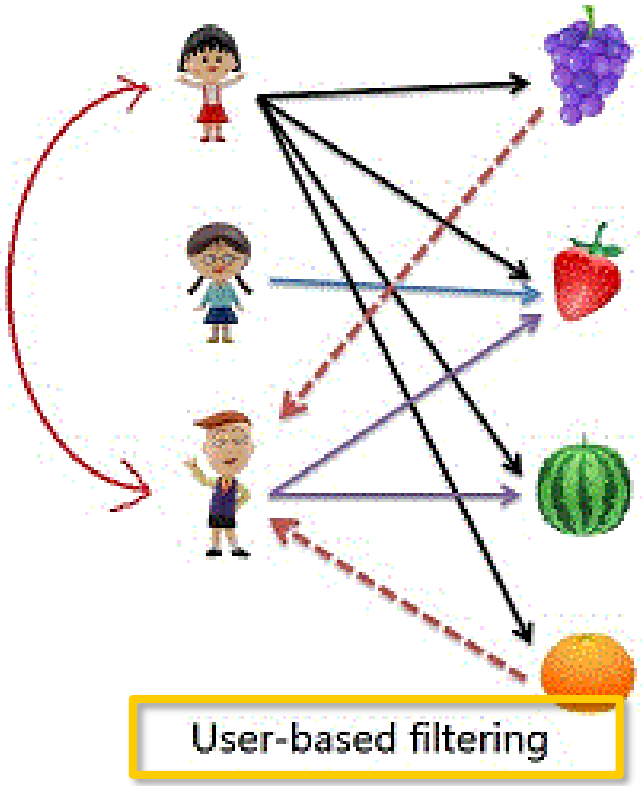


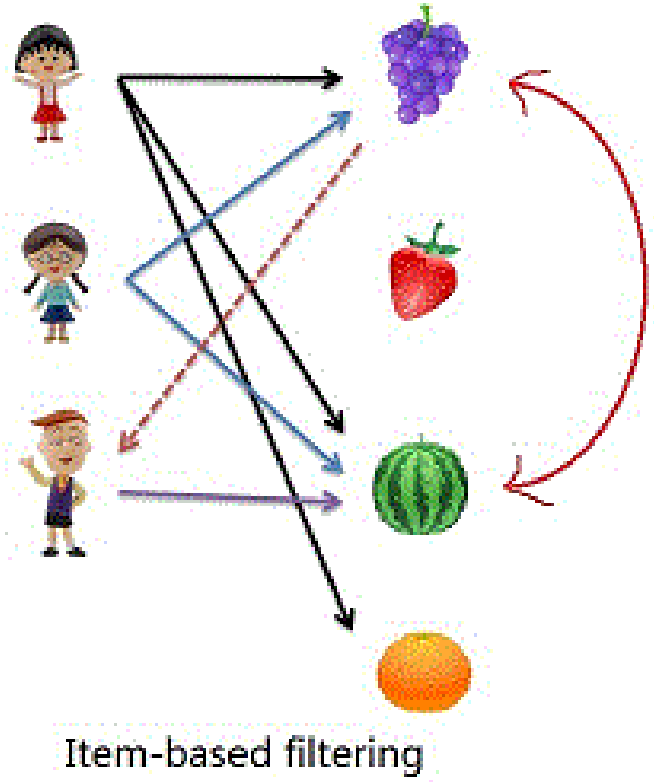
그렇다면 사용자의 특성에 맞는 맞춤 여행지를 추천해주자!

02

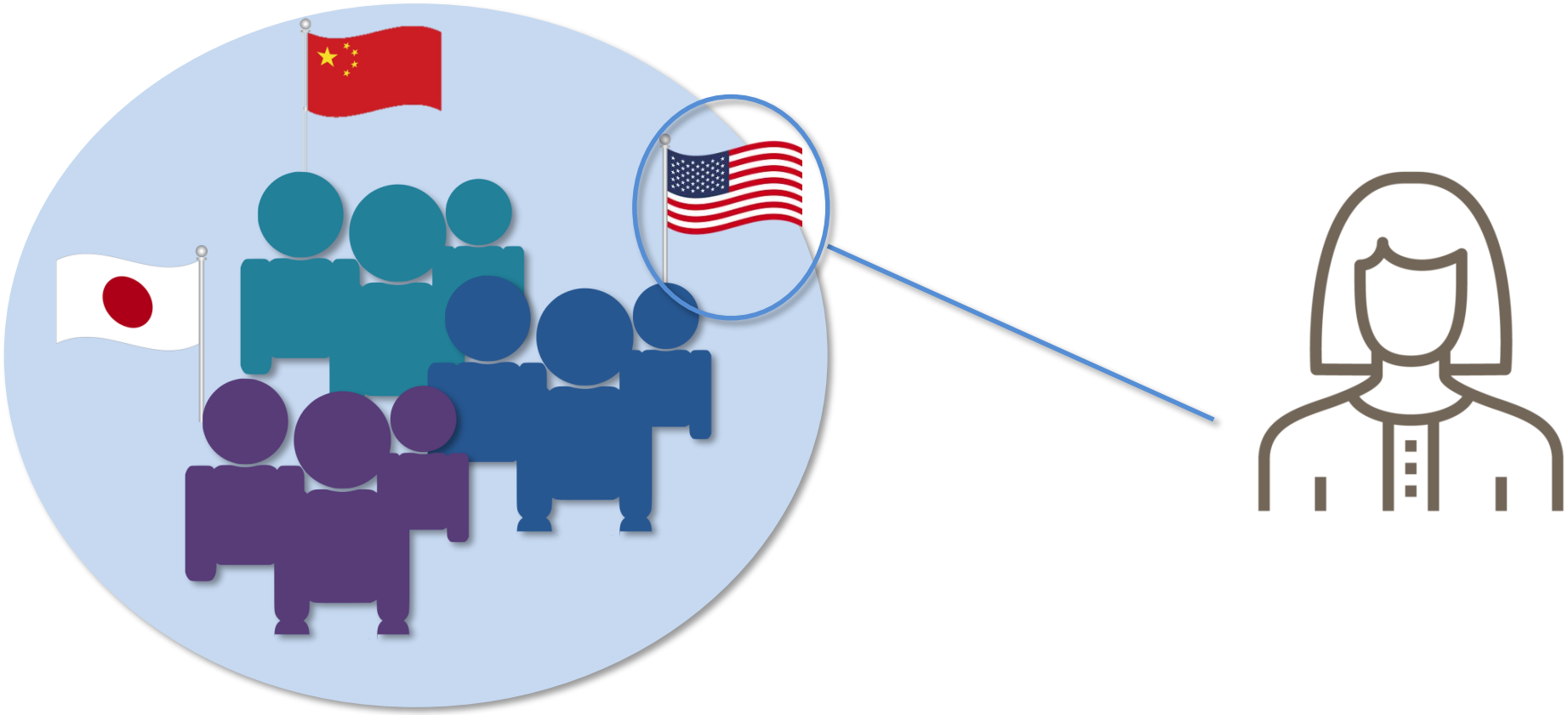
분석 전략

Collaborative Filtering Algorithm





여행지에서의 활동 만족도가 비슷한 사람이 간 해외 여행지 추천



03

데이터 전처리

관광지식정보시스템에서 제공하는 내국인 국내여행 실태 데이터 자료

응답자 특성 데이터

2015, 2016, 2017년 응답자 개인 특성 기록 6170건

HID	PID	sex	age	sr_type	sido	ara_size	school1	school2	occ0	occ1	occ2	fac	marry
10001	1000101	1	40	2	1	1	5	5		2	8	4	2
10001	1000102	2	40	2	1	1	4	1		11	9	4	2

속성: 성별, 나이(연령대), 혼인 상태, 학력(교육 정도), 직업, 가구번호, 가구소득 등

여행 데이터

2015, 2016, 2017년 설문지 조사 내용 6170건

hid	PID	type1	type2	Month	M_ID	q1	q2_a	q2_a_1	q2_a_2	q2_b	q2_b_1	q2_b_2	q2_c_1	q2_c_2
10001	1000101	1	2	8	115439	2	2017	8	4	2017	8	5	1	2
10001	1000102	1	2	12	115441	1	2017	12	23	2017	12	23	0	1

속성: 구매한 여행상품, 동행한 사람과의 관계, 숙박비, 교통, 관광지 물가 등

방문지 데이터

2015, 2016, 2017년 설문지 조사 내용 6170건

hid	PID	type1	type2	Month	M_ID	q6_1	q6_1_1	q6_2	q6_3_a	q6_3_b	q6_3_c	q6_4	q6_5_1
10001	1000101	1	2	8	115439	931	010	2	1	5	7	1	
10001	1000102	1	2	12	115441	931	260	1	1	3	7	1	1

속성: 여행장소, 여행 월, 여행지 선택 이유, 여행지에서의 활동 등

응답자 특성 데이터

```
personal17 = pd.read_excel('personal_2017.xlsx')
personal16 = pd.read_excel('personal_2016.xlsx')
personal15 = pd.read_excel('personal_2015.xlsx')
```

여행 데이터

```
travel17 = pd.read_excel('travel_2017.xlsx')
travel16 = pd.read_excel('travel_2016.xlsx')
travel15 = pd.read_excel('travel_2015.xlsx')
```

방문지 데이터

```
place17 = pd.read_excel('place_2017.xlsx')
place16 = pd.read_excel('place_2016.xlsx')
place15 = pd.read_excel('place_2015.xlsx')
```

MERGE

병합 데이터

```
data17 = pd.merge(personal17, travel17, on = 'pid')
data17 = pd.merge(data17, place17, on = 'pid')
```

각각의 레코드: 개인의 3년 동안의 여행 정보
6170명에 대한 1년 동안의 여행 정보 데이터

병합 데이터

↓

해외 여행 데이터

↓

여행지 활동
만족도 데이터

```
data["income"] = data["income"].map(lambda i: i/1000 if i > 0 else 0).astype(int)
data["cost"] = data["cost"].map(lambda i: i/100000 if i > 0 else 0).astype(int)
```

```
data17 = data17[data17.type == 2]
```

```
act_data = data[["act_nature", "act_food", "act_sports", "act_history", "act_play",
                 "act_rest", "act_spa", "act_shop", "act_culture", "act_watchgame",
                 "act_festival", "act_program", "act_religion", "act_gambling",
                 "act_citytour", "act_film", "act_visit", "act_meeting",
                 "act_edu", "act_entertain", "act_etc"]]
```

```
import random
random.seed(1)
for i in range(721):
    for j in range(0,21):
        if act_data.values[i][j] != 0:
            act_data.values[i][j] = random.randrange(1,6)

act_data_pid = pd.concat([data['pid'], act_data, data['country_city']], axis=1)
```

	pid	act_nature	act_food	act_sports	act_history	act_play	act_rest	act_spa	act_shop	act_culture
9	1000601	0	0	0	0	0	0	0	0	0
34	1002203	5	1	3	1	4	0	0	0	0
35	1002204	4	4	4	2	1	0	0	0	0
56	1003501	0	0	0	0	0	0	0	0	0
96	1005603	1	4	0	0	0	0	4	0	0
107	1006203	0	0	0	0	0	0	0	0	0
207	1012201	0	0	0	0	0	0	0	0	0
230	1013602	0	0	0	2	5	1	0	0	0
231	1013603	3	0	0	1	1	0	0	0	0
232	1013604	1	0	0	5	0	1	0		
236	1013704	4	2	0	0	0	4	0		

속성: 풍경 감상, 음식 관광, 스포츠 활동, 역사 유적지 방문, 놀이시설, 휴양 등 총 21가지

여행지 활동
만족도 데이터

04

분석 과정

유클리디안 거리

$$\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\ldots+(p_n-q_n)^2}=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

평균 제곱 차이 유사도

$$\mathrm{msd}(u,v)=\frac{1}{|I_{uv}|}\cdot\sum_{i\in I_{uv}}(r_{ui}-r_{vi})^2 \qquad \mathrm{msd\_sim}(u,v)=\frac{1}{\mathrm{msd}(u,v)+1}$$

코사인 유사도

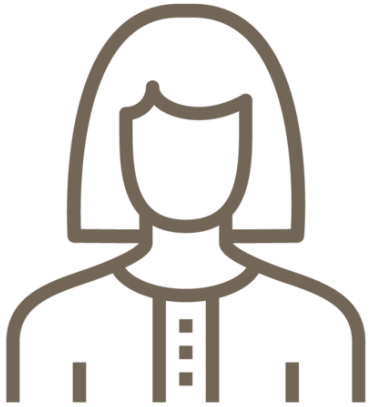
$$x\cdot y=|x||y|\cos\theta$$

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

$$\mathrm{cosine\_sim}(u,v)=\frac{\sum_{i\in I_{uv}} r_{ui}\cdot r_{vi}}{\sqrt{\sum_{i\in I_{uv}} r_{ui}^2}\cdot\sqrt{\sum_{i\in I_{uv}} r_{vi}^2}}$$

피어슨 유사도

$$\mathrm{pearson\_sim}(u,v)=\frac{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)\cdot(r_{vi}-\mu_v)}{\sqrt{\sum_{i\in I_{uv}}(r_{ui}-\mu_u)^2}\cdot\sqrt{\sum_{i\in I_{uv}}(r_{vi}-\mu_v)^2}}$$



pid: 1002203

해외 여행지

노르웨이(23) - 오슬로(001)

여행지에서의 활동 및 만족도

자연 및 풍경 감상	음식 관광	야외 위락 및 스포츠 활동	역사 유적지 방문	테마파크, 동/식물원 방문
5	1	3	1	4

유클리디안 거리

추천 해외 여행지

대만(30) - 타이베이(001)

일본(171) - 쿄토후(002)

필리핀(232) - 마닐라(001)

일본(171) - 도쿄토(004)

괌(8) - 아가냐(001)

중국(176) - 쑤저우(022)

```
# 유클리드 거리
# act 데이터

from math import sqrt
from collections import OrderedDict
import warnings
warnings.filterwarnings('ignore')

pid_count = act_data_pid[['pid']]
pid_count['count'] = 0

def sim_distance(data, name1, name2):
    s = 0
    for i in act_data:
        if (int(data[data.pid == name1][i]) is not 0) and (int(data[data.pid == name2][i]) is not 0):
            pid_count.loc[pid_count.pid == name2, 'count'] += 1
            s += pow(int(act_data_pid[act_data_pid.pid == name1][i]) - int(act_data_pid[act_data_pid.pid == name2][i]),2)

    return sqrt(s)

def top_match(data, name, index = 10, sim_function = sim_distance):
    li = []
    for i in data["pid"]:
        sim_function(data, name, i)
        if (name is not i) and (int(pid_count.loc[pid_count.pid == i, 'count']) is not 0):
            li.append((sim_function(data, name, i) / int(pid_count.loc[pid_count.pid == i, 'count']), i))
    li.sort()

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

top_match(act_data_pid, 1002203)
```

```
[23001, 30001, 171002, 232001, 171004, 8001, 176022]
```

평균 제곱 차이 유사도

추천 해외 여행지

일본 (171) - 오사카후(007)

스위스(122) - 제네바(004)

베트남(82) - 다낭(004)

싱가포르(128) - 싱가포르(001)

일본(171) - 도쿄토(004)

오스트레일리아(153) - 시드니(003)

홍콩(236) - 침사추이(002)

```python
# 평균제곱차이 유사도
# act 데이터

from collections import OrderedDict

def sim_msd(data, name1, name2):
    s = 0
    count = 0
    for i in act_data[data.pid == name1]:
        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], "country_city"]) is not int(data.loc[data.pid == name, "country_city"]):
            c.append(int(data.loc[data.pid == li[i][1], "country_city"]))
            c = list(OrderedDict.fromkeys(c))

    return c

top_match(act_data_pid, 1002203)
```

```
[171007, 122004, 82004, 128001, 171004, 153003, 236002]
```

코사인 유사도

추천 해외 여행지

일본 (171) - 오사카후(007)

베트남(82) - 다낭(004)

싱가포르(128) - 싱가포르(001)

스위스(122) - 제네바(004)

중국(176) - 상하이(005)

일본(171) - 도쿄토(004)

태국(207) - 푸켓(002)

오스트레일리아(153) - 시드니(003)

인도네시아(170) - 발리(001)

```python
# 코사인 유사도
# act 데이터

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 act_data[data.pid == name1]:
        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], "country_city"]) is not int(data.loc[data.pid == name, "country_city"]):
            c.append(int(data.loc[data.pid == li[i][1], "country_city"]))
            c = list(OrderedDict.fromkeys(c))

    return c

top_match(act_data_pid, 1002203)
```

```
[171007, 82004, 128001, 122004, 176005, 171004, 207002, 153003, 170001]
```

피어슨 유사도

추천 해외 여행지

일본 (171) - 오사카후(007)

베트남(82) - 다낭(004)

싱가포르(128) - 싱가포르(001)

스위스(122) - 제네바(004)

중국(176) - 상하이(005)

일본(171) - 도쿄토(004)

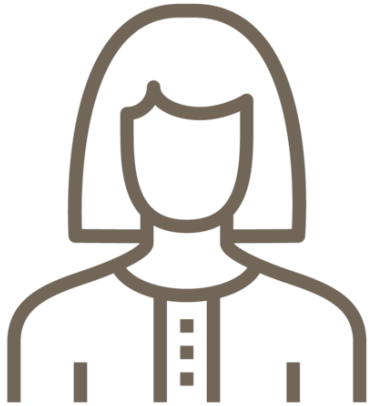
태국(207) - 푸켓(002)

오스트레일리아(153) - 시드니(003)

인도네시아(170) - 발리(001)

```
# 피어슨 유사도
# act 데이터

from collections import OrderedDict

def sim_pearson(data, name1, name2):
    avg_name1 = 0
    avg_name2 = 0
    count = 0
    for i in act_data[data.pid == name1]:
        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 act_data[data.pid == name1]:
        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], "country_city"]) is not int(data.loc[data.pid == name, "country_city"]):
            c.append(int(data.loc[data.pid == li[i][1], "country_city"]))
            c = list(OrderedDict.fromkeys(c))

    return c

top_match(act_data_pid, 1002203)
```

```
[171007, 82004, 128001, 122004, 176005, 171004, 207002, 153003, 170001]
```



pid: 1002203

자연 및 풍경 감상	테마파크, 동/식물원 방문
5	4

종합 추천 해외 여행지

일본 (171) - 오사카후(007)

일본(171) - 도쿄토(004)

베트남(82) - 다낭(004)

스위스(122) - 제네바(004)

싱가포르(128) - 싱가포르(001)

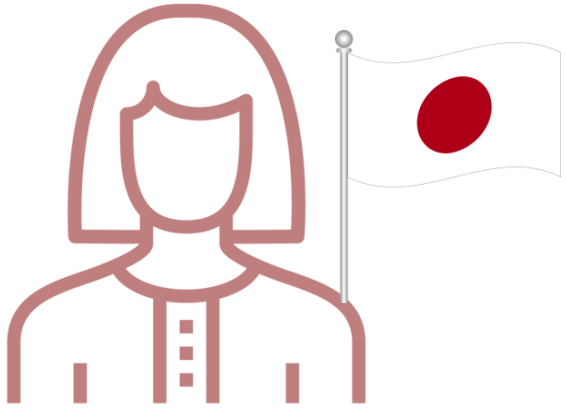
대만(30) - 타이베이(001)

일본(171) - 쿄토후(002)

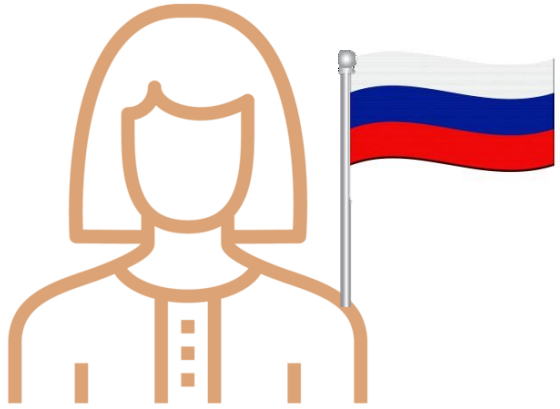
오스트레일리아(153) - 시드니(003)

05

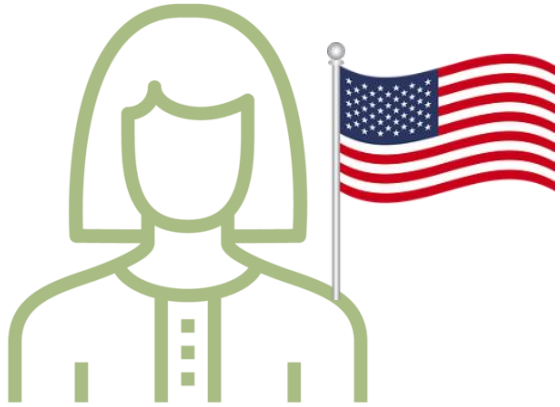
결과 및 해석



pid: 1714257



pid: 1613413



pid: 1712377

해외 여행지

일본(171) - 도쿄토(004)

러시아(39) - 블라디보스톡(002)

미국(70) - 샌프란시스코(018)

여행지에서의 활동 및 만족도

```
act_data_pid.loc[0] = [1714257,1,2,4,5,3,0,5,3,2,4,4,4,0,0,0,0,4,5,2,3,4,171004]
act_data_pid.loc[1] = [1613413,0,5,0,0,3,4,0,2,3,0,0,0,2,0,0,0,0,0,0,0,0,39002]
act_data_pid.loc[2] = [1712377,3,3,2,0,2,3,0,4,0,0,0,0,0,0,5,0,0,0,0,0,5,70018]
```

pid: 1714257

종합 추천 해외 여행지

일본(171) - 훗카이도(042)

베트남(82) - 다낭(004)

괌(80) - 아가냐(001)

오스트레일리아(153) - 시드니(003)

스페인(123) - 마드리드(001)

인도네시아(170) - 발리(001)

뉴질랜드(26) - 오클랜드(001)

미국(70) - 몬터레이(011)

역사 유적지 방문 | 온천 / 스파

pid: 1613413

종합 추천 해외 여행지

베트남(82) - 다낭(004)

일본(171) - 규슈(052)

일본(171) - 쿄토후(002)

터키(208) - 앙카라(007)

필리핀(232) - 보라카이(018)

영국(149) - 맨체스터(007)

미국(70) - 로스앤젤레스(008)

캄보디아(188) - 프놈펜(001)

음식 관광 | 휴식 / 휴양

pid: 1712377

종합 추천 해외 여행지

영국(149) - 맨체스터(007)

스페인(123) - 마드리드(001)

베트남(82) - 호치민(006)

스페인(123) - 바르셀로나(002)

인도네시아(170) - 발리(001)

태국(207) - 푸켓(002)

중국(176) - 광둥성(030)

일본(171) - 도쿄토(004)

시티 투어 | 기타

한계 및 보완점

데이터 상 아무도 가지 않았던 나라는 추천되지 않음

사람들이 많이 갔던 여행지가 많이 추천되는 경향

여행 날짜에 맞는 나라를 고려하지 않음

감사합니다