경기도 거주 및 방문 외국인을 위한 음식점 추천 서비스

새로운 경기

공정한 세상

경기도 데이터 산업인력 양성사업 3조
TEAM '6인분'

목차

연구 배경 01

02 데이터 전처리

이미지 분류 03

04 지도 시각화

한계점 및 발전방향 05

06 참고문헌

1. 연구 배경

여행계획에 있어 음식은 중요한 요소 중 하나이며 SNS와 같은 온라인 검색 서비스를 활용하여 다양한 정보를 확인한다. 하지만 수많은 검색결과가 존재함으로 양질의 정보를 선별하기 많은 어려움이 따른다.

이러한 문제점을 바탕으로 대한민국을 방문하는 **외국인을 대상으로** 음식을 선정하는 과정에 있어 **정확한 정보 제공을 목적으로** 한다.

A.

국내 외국인
증가 현황

B.

‘한국 여행
방문지
만족도 순위’

C.

‘한국 방문
선택 시
고려 요인’

D.

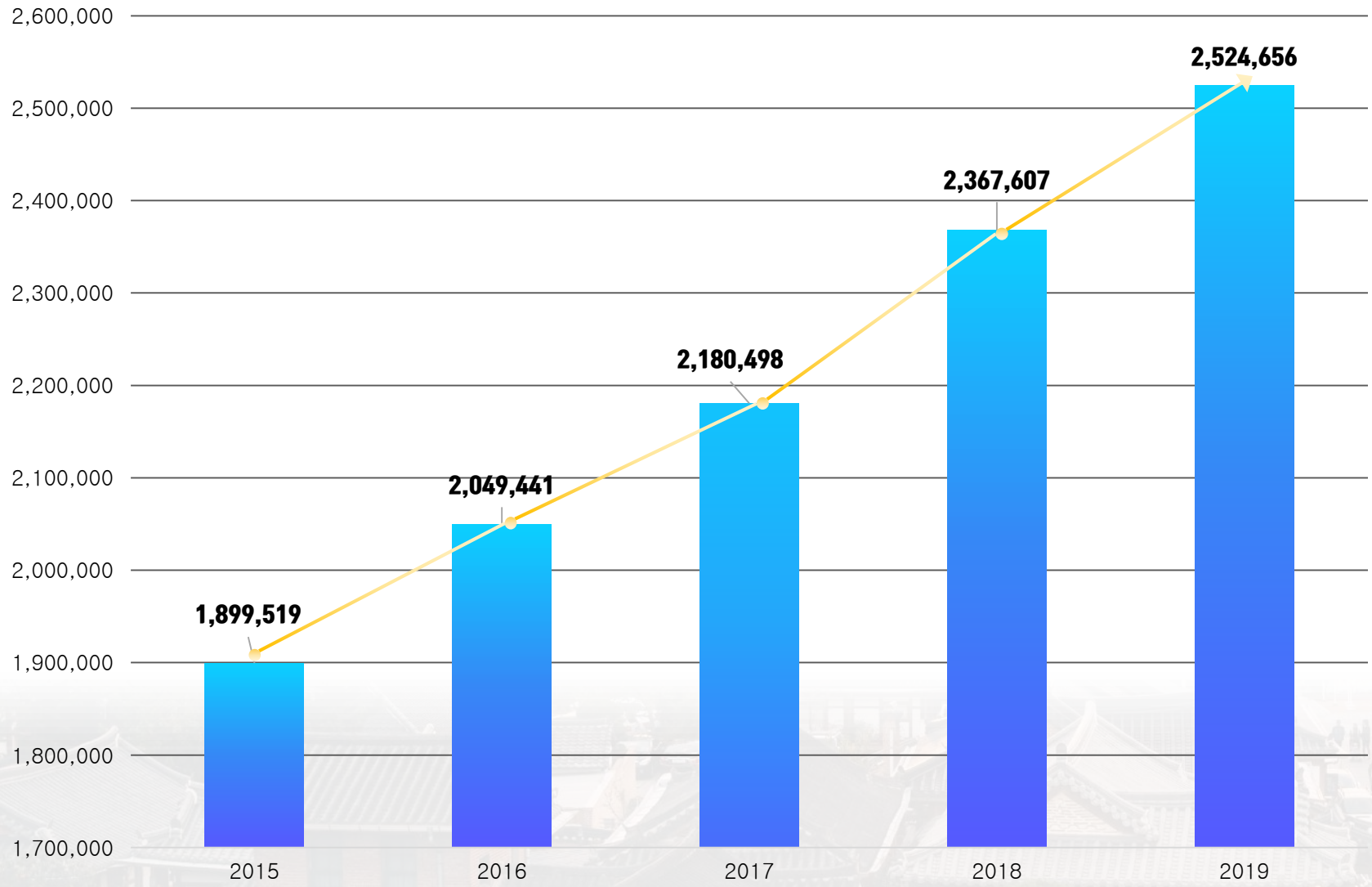
‘한국 방문 후
만족한 활동’

E.

‘한국 여행시
정보 출처’

외국인 음식 정보 검색

주제를 선정하게 된 주요요인을 살펴보고 연관관계를
파악하고자 데이터 시각화를 위한 전처리 과정 진행

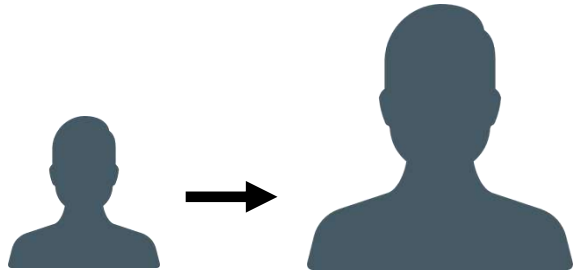


[단위 : 명]

	2015	2016	2017	2018	2019
체류외국인	1,899,519	2,049,441	2,180,498	2,367,607	2,524,656

A.

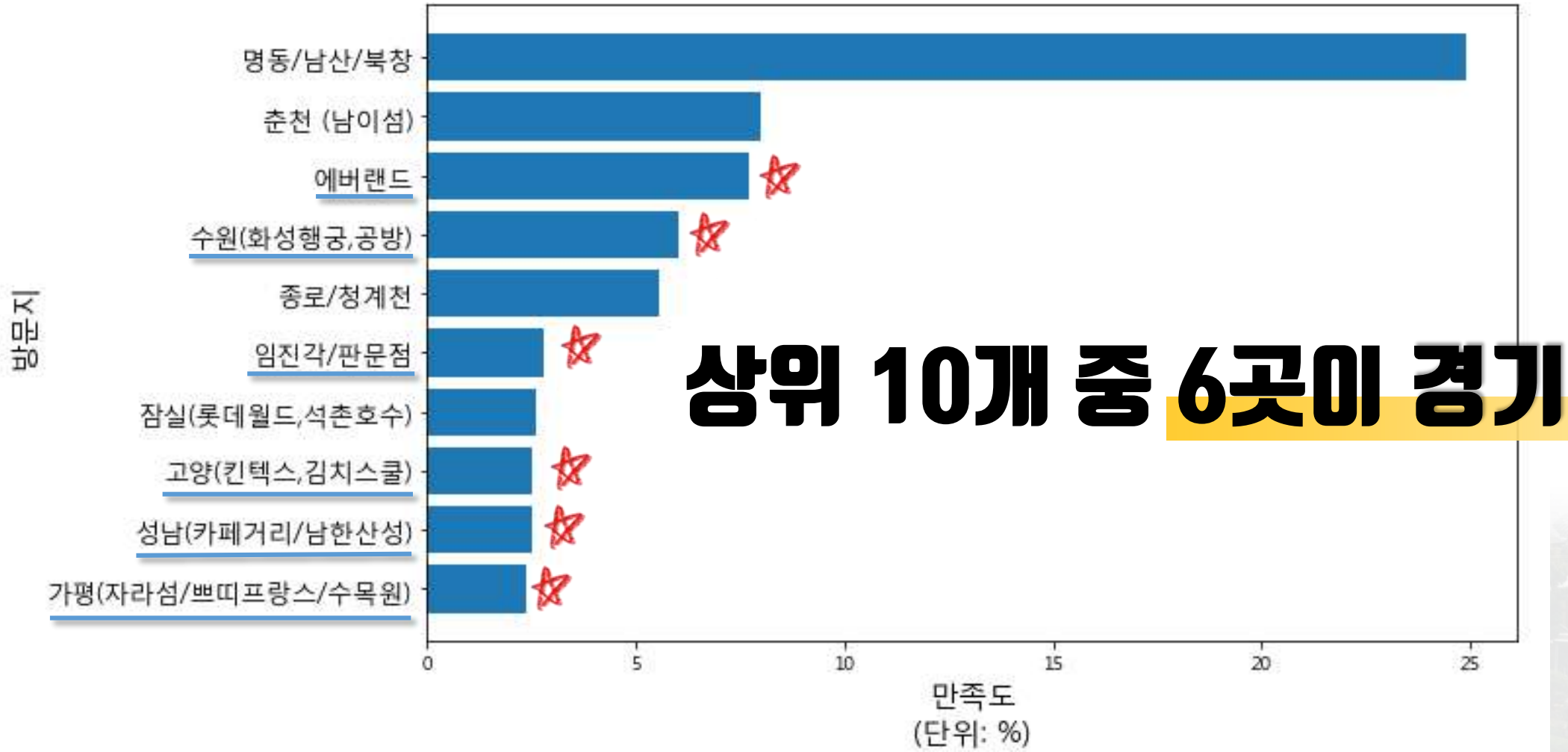
국내 외국인 증가 현황



대한민국에 체류하는 외국인들이
점점 증가하고 있는 것을 확인

원본 데이터 정보 출처 : - 나라지표

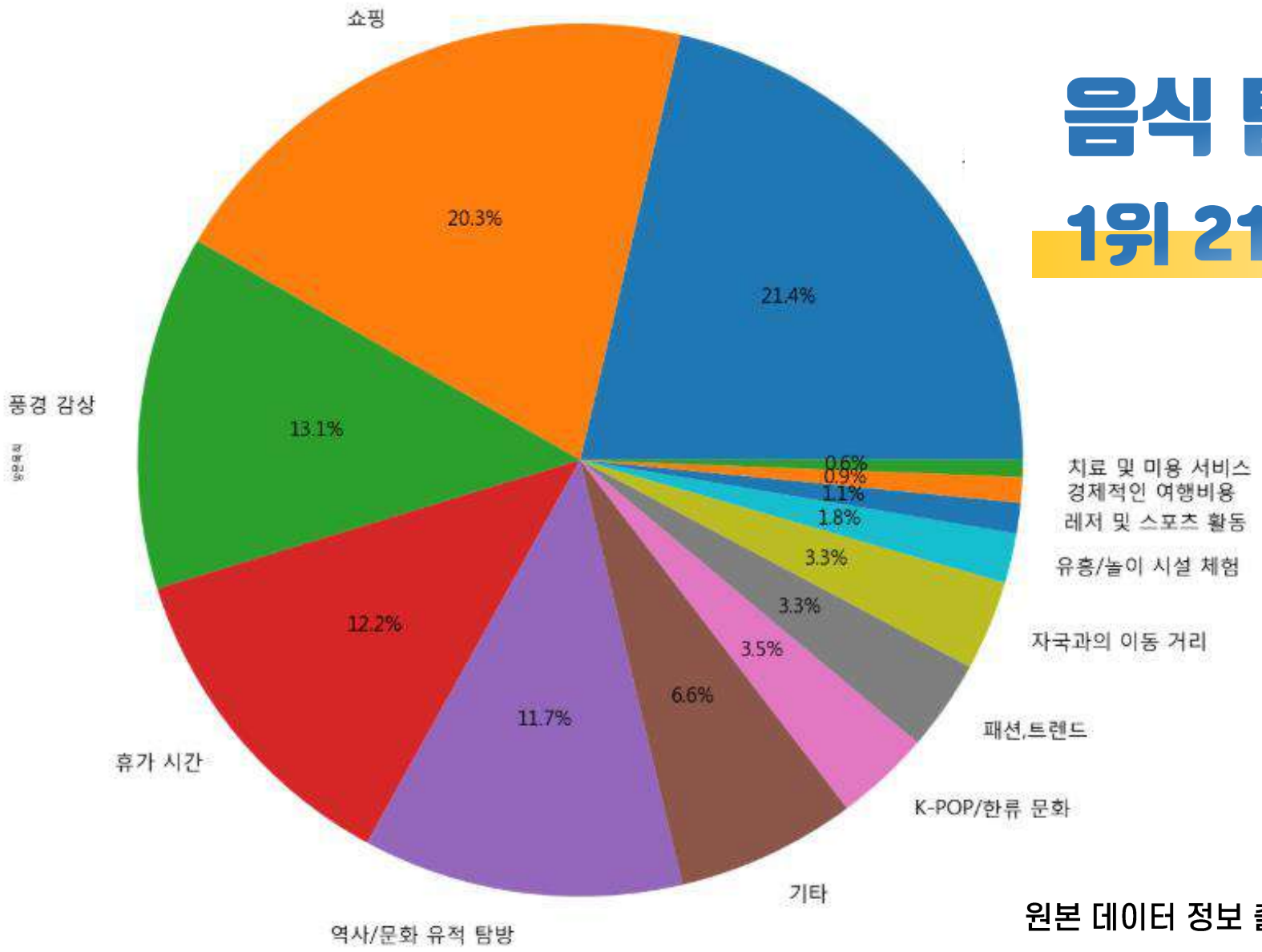
B. ‘한국 여행 방문지 만족도 순위’



상위 10개 중 6곳이 경기도

원본 데이터 정보 출처 : 한국관광공사

c. '한국 방문 선택 시 고려 요인'

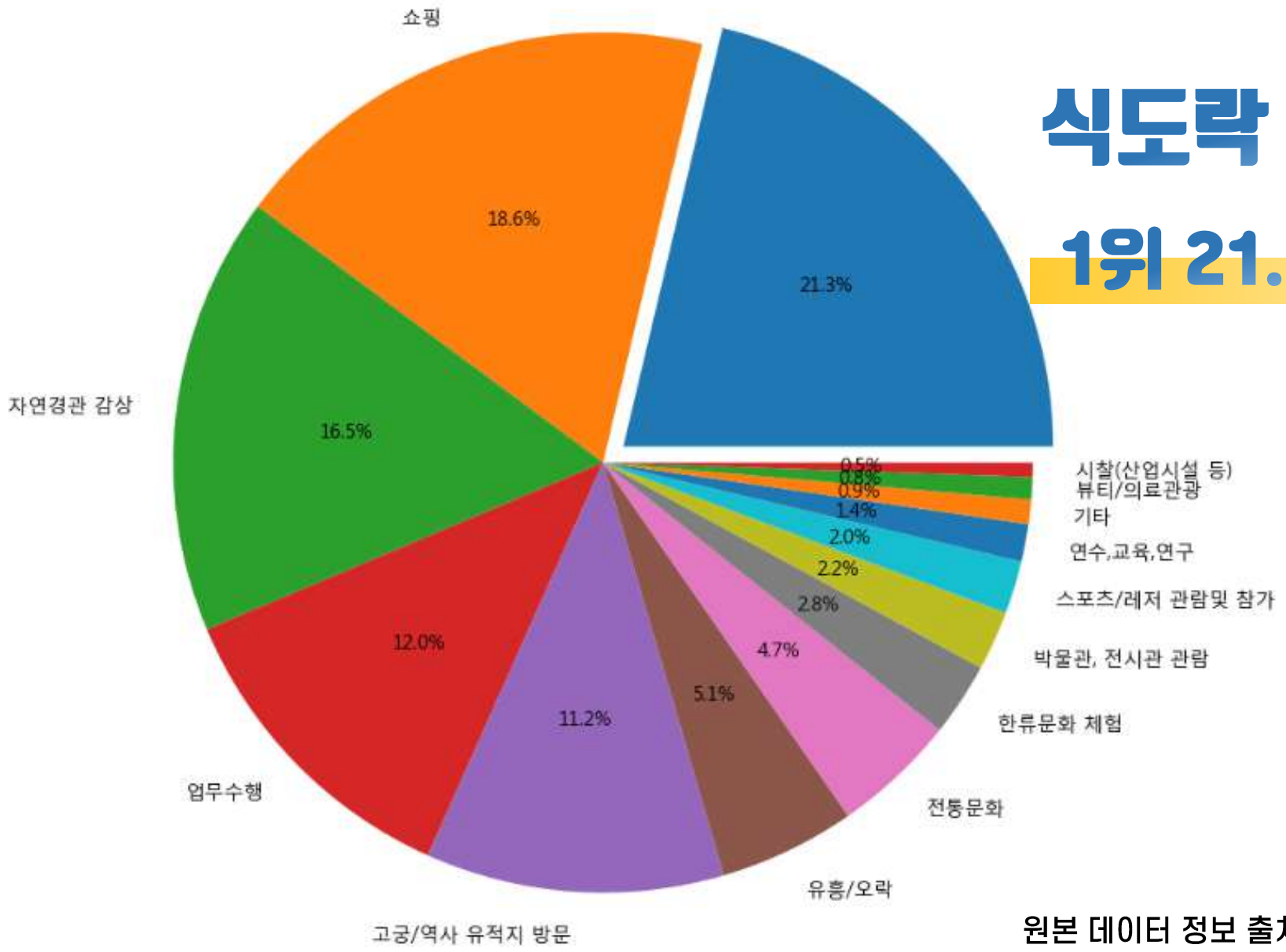


음식 탐방

1위 21.3%

원본 데이터 정보 출처 : 한국관광공사

D. ‘한국 방문 후 만족한 활동’



식도락 관광

1위 21.3%

원본 데이터 정보 출처 : 한국관광공사

F. ‘한국 여행시 정보 출처’

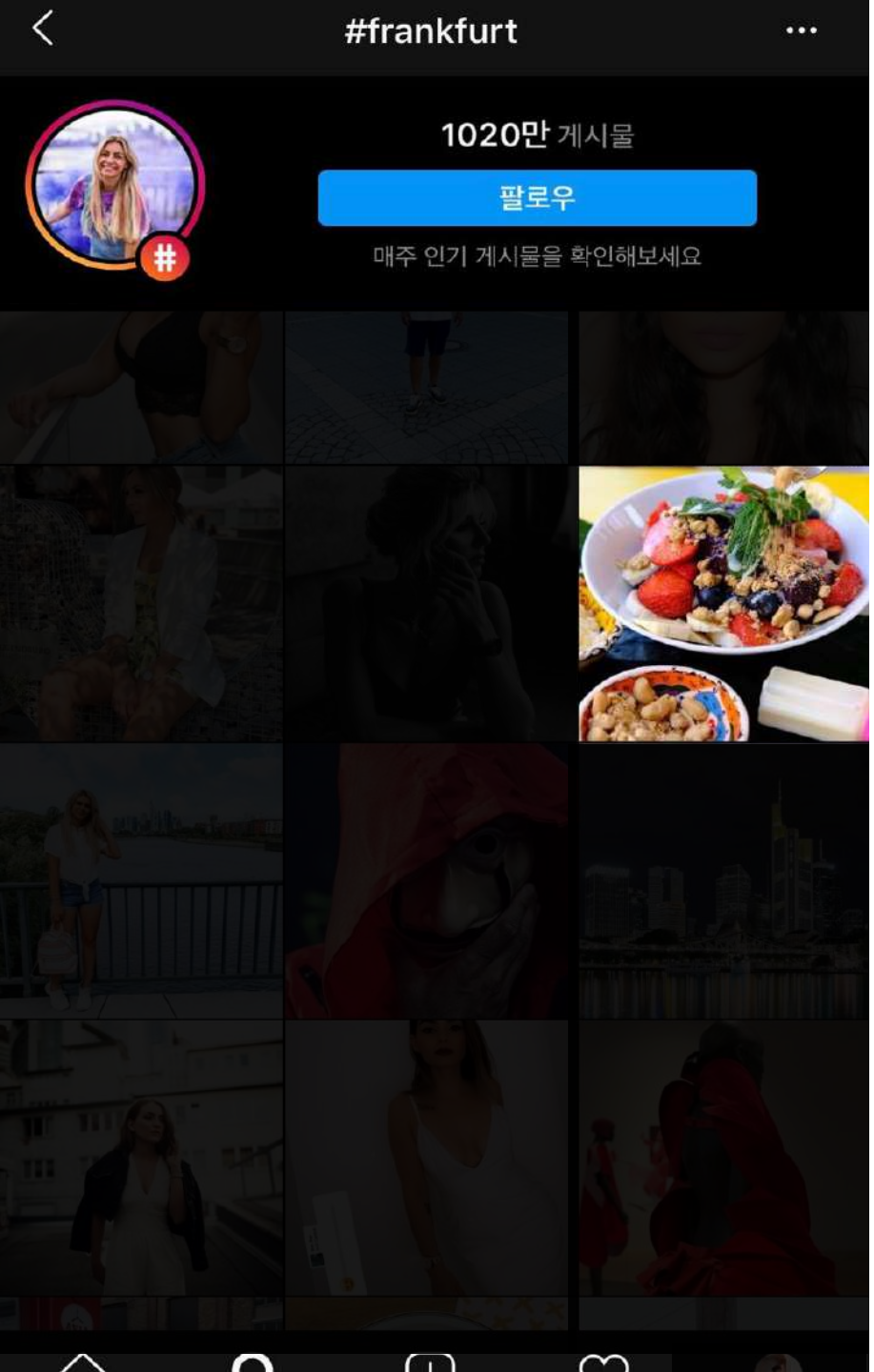
인터넷(스마트폰/ 노트북/ 테블릿 PC): 64 %

출처	비율(%)
인터넷(스마트폰)	59.8076
지인	15.947
여행가이드	10.0925
인터넷(노트북/테블릿 PC)	5.16325
여행 가이드북	2.62845
지역주민	2.39453
다른 여행자	1.49681
없음	1.10973
기타	0.577989
공항/항만 내 안내소	0.441689
공항/항만 이외 안내소	0.440371

여행 정보 제공 시스템이 충분한가?
한국 여행을 계획하는데 불편한 점은 없는가?

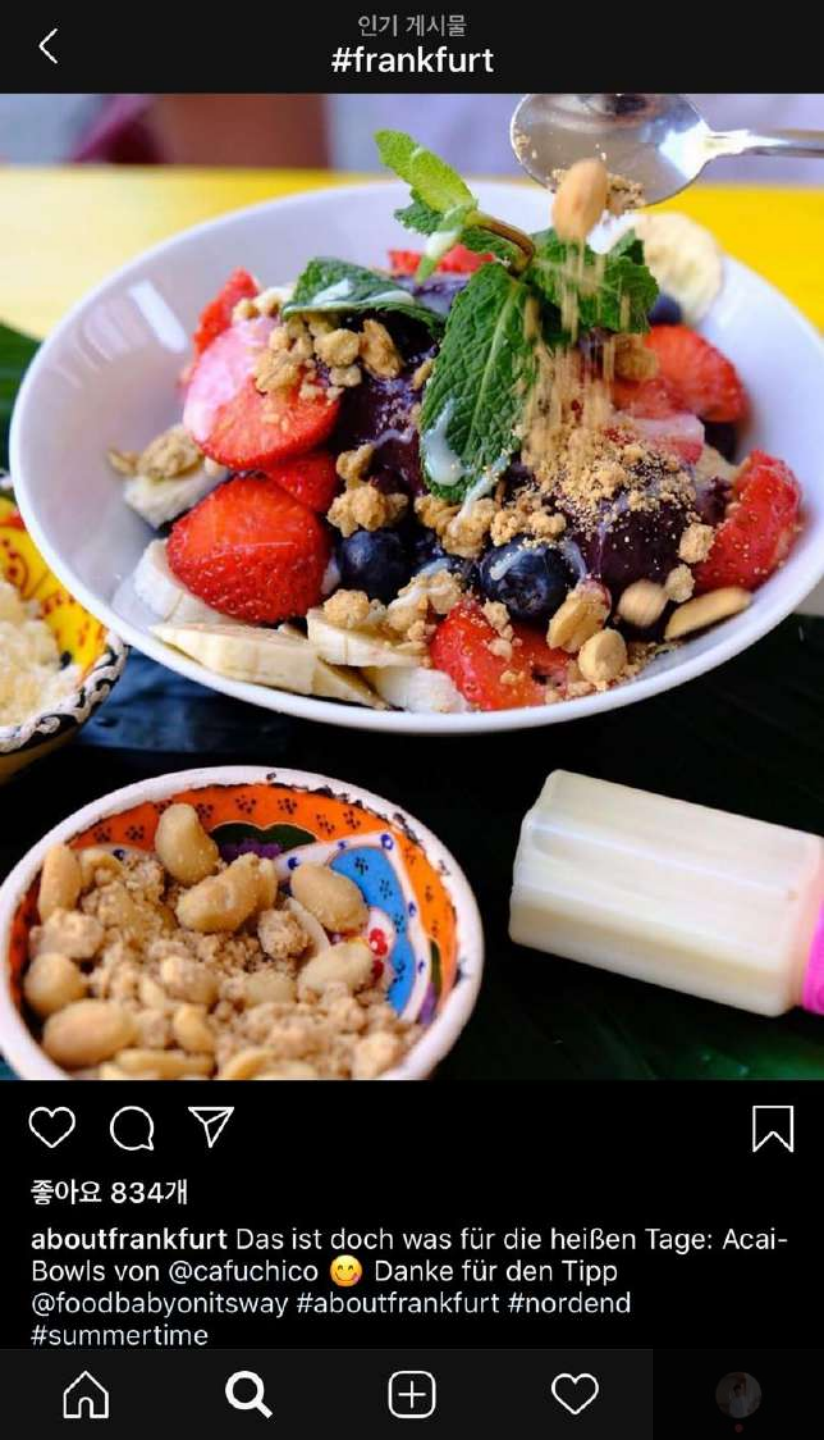
원본 데이터 정보 출처 : 한국관광공사

독일의 프랑크푸르트로
여행을 떠나는 계획을
세운다고 가정해봅시다.



#Frankfurt 검색

현지 분위기 등을 보기 위해 SNS를 검색하다가, 정말 맛있어 보이는 **음식사진을 발견**하게 됩니다.



#Frankfurt 검색 🇩🇪

정말 안타깝게도,
게시물의 설명이 **독일어로**
써져 있었고, 음식이름에 대한
정보는 이해할 수 없었습니다. ☹

그렇다면 대한민국 경기도에 여행 오는 외국인들은?

#Suwon 검색 🇰🇷

우리와 마찬가지로 게시물이 한국어로 써져 있어, 음식이름에 대한 **정보를 찾기 힘들 것**이라 생각했습니다. ☹

Needs

?

“ Yugsasimi ”
“ Yughoe ”

‘사진을 통해 한식의 이름을 알아내는 기술’

기본 모델

음식 이미지

음식 이미지
분석
(딥러닝)

위치 기반
모범 음식점 추천

데이터 출처: 경기데이터드림

2. 데이터 전처리 과정

a. 한국 방문 선택시 고려 요인
b. 한국 여행 계획 정보 출처
c. 한국 방문지 만족도 순위
d. 한국 여행시 만족한 활동

4가지 데이터를 바탕으로 음식 추천에 대한 필요성을 확인하였다.

주요 데이터 전처리 과정

a.‘한국 방문 **선택시 고려 요인**’

c.한국 ‘방문지 **만족도 순위**’

b.‘한국 여행 계획 **정보 출처**’

d.한국 여행시 ‘**만족한 활동**’

출처 : 한국관광공사 ‘2019 외래관광객조사표-경기도’

한국관광공사

a. '한국 방문 선택시 고려 요인'

초기 데이터 값을 가져와,
원하는 정보 추출을 위해 전처리 진행

한국방문 고려요인

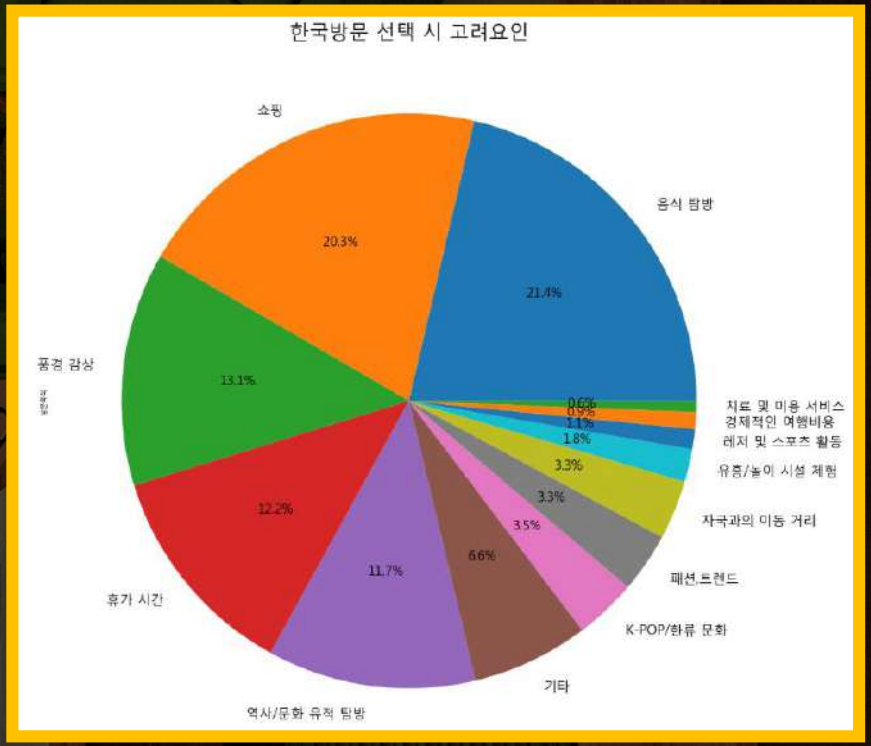
```
visit_s = pd.read_excel('한국방문 선택 시 고려요인.xlsx',sep=',',encoding='utf8')
```

```
visit_s.head(5)
```

	BASE: 여가, 위락, 휴식, 친구/친지 방문 목적	Unnamed:	사례	음식/미식	쇼핑	자연 풍경	휴가 시간	역사/문화	기타	K-POP/ 한류스타 관련 팬	패션, 유행 등세	자국과의 이동 거	유흥/놀이 시설	레저 및 스포츠	경제적인
0	▨전체▨	Na													
1	▨월별▨	1													
2	NaN	2													
3	NaN	3													
4	NaN	4													

```
In [59]: visit_s = visit_s.drop('BASE:여가,위락, 휴식, 친구/친지 방문 목적',axis=1)
         visit_s1 = visit_s.rename({'Unnamed: 1':'방문목적'},axis=1).fillna('비율')
         visit_s1 = visit_s1.rename({'음식/미식 탐방':'음식 탐방','자연 풍경 감상':'풍경 감상',
                                     'K-POP/한류스타 관련 팬 미팅, 촬영지 등 경험':'K-POP/한류 문화',
                                     '패션, 유행 등세련된 현대 문화':'패션,트렌드'},axis=1)
         visit_s1
```

	방문목적	사례수	음식 탐방	쇼핑	풍경 감상	휴가 시간	역사/문화 유적 탐방	기타	K-POP/한류 문화	패션,트렌드	자국과의 이동 거리	유흥/놀이 시설 체험	레저 및 스포츠 활동	경제적인 여행비용	치료 및 미용 서비스	계
0	비율	(1650)	21.353858	20.276021	13.131696	12.241142	11.714732	6.585169	3.488149	3.348062	3.324149	1.836393	1.133897	0.946078	0.620653	100
1	1월	(91)	13.480903	27.498510	10.308175	11.826332	5.252967	9.112053	3.816827	3.051359	4.726662	2.660316	3.608537	3.105967	1.551391	100
2	2월	(88)	26.664469	24.015920	10.518929	8.963075	10.678036	6.485019	1.905757	0.211728	2.480446	2.156722	2.931475	0.000000	2.988424	100
3	3월	(128)	19.610709	19.812585	17.232830	15.091362	9.152485	1.817646	3.473969	3.357530	6.074295	0.666132	0.000000	1.625901	2.084558	100
4	4월	(191)	22.583952	13.596584	11.664814	14.848375	17.012873	6.353546	3.728637	5.210060	1.561838	2.636020	0.206386	0.456648	0.140266	100
5	5월	(130)	23.128014	19.078157	13.908537	13.863807	17.407337	3.936860	1.760044	2.490142	0.659563	0.397583	0.000000	2.721028	0.648929	100



필요 없는 칼럼 삭제 및 칼럼명 변경

a. '한국 방문 선택시 고려 요인'

그래프를 그리기 위한 값을 설정

```
labels = list(visit_s1.iloc[0:1,2:15].columns)
labels
```

```
['음식 탐방',
 '쇼핑',
 '풍경 감상',
 '휴가 시간',
 '역사/문화 유적 탐방',
 '기타',
 'K-POP/한류 문화',
 '패션,트렌드',
 '자국과의 이동 거리',
 '유흥/놀이 시설 체험',
 '레저 및 스포츠 활동',
 '경제적인 여행비용',
 '치료 및 미용 서비스']
```

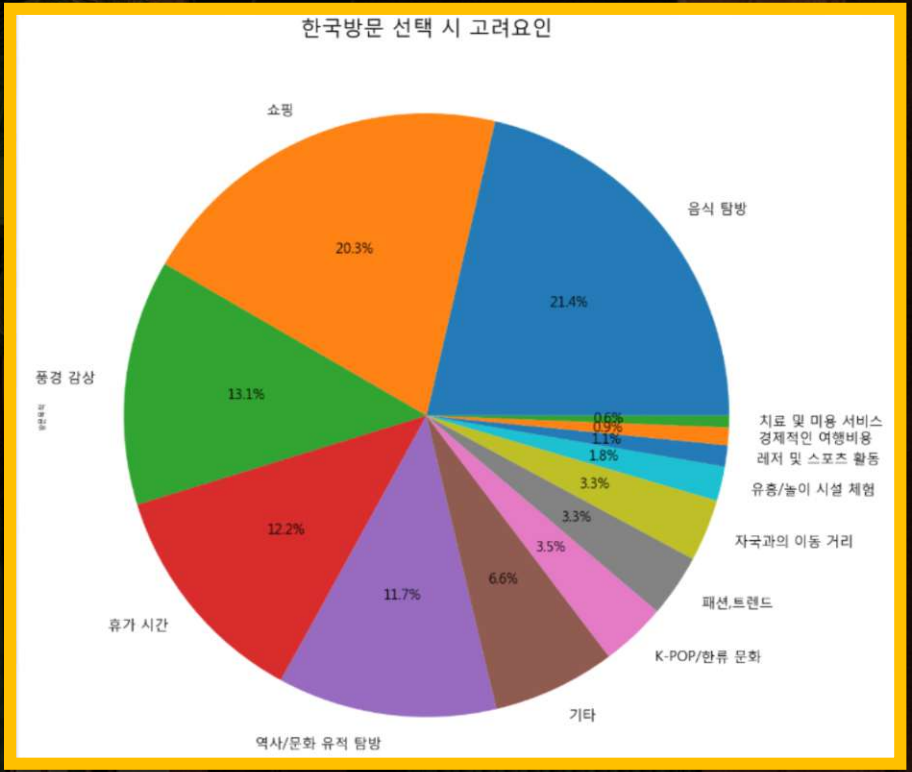
```
values = list(visit_s1.iloc[0,2:15])
values
```

```
[21.353857937884808,
 20.27602059996278,
 13.131696326634811,
 12.241142430515835,
 11.714731878704209,
 6.585169033497097,
 3.4881487624603915,
 3.348062384836469,
 3.324149072025791,
 1.8363929146860456,
 1.133897455974594,
 0.946077924386526,
 0.6206532784305188]
```

```
pie = pd.DataFrame({'방문목적':values},
                   index = labels)
```

```
pie['방문목적'].plot(kind='pie',figsize=(20,20),
                     autopct = '%1.1f%%',fontsize = 20)
plt.title('한국방문 선택 시 고려요인', fontsize=30)
plt.show()
```

설정 labels, value 값 활용 그래프 그리기



b.'한국 여행시 정보 출처'

정보출처

```
i_source = pd.read_excel('정보출처순위.xlsx', sep=',', encoding='utf8', skiprows=1)

i_source2 = i_source.rename({'Unnamed: 1':'출처','한국 친구, 친지에게 문의':'지인',
                             '한국 지역 주민에게 문의':'지역주민','다른 여행자에게 문의':'다른 여행자',
                             '공항/항만 내 관광 안내소':'공항/항만 내 안내소',
                             '공항/항만 이외 관광 안내소':'공항/항만 이외 안내소'}, axis=1).fillna('비율(%)')

i_source2 = i_source2.drop('BASE:전체',axis=1)

i_source3 = i_source2.iloc[0:1,:].drop('사례수',axis=1).T  # T: 행렬 전치

i_source3
```

정보출처

```
i_source.head(5)
```

	BASE: 전체	Unnamed: 1	사례수	인터넷(스마트폰)	한국 친구, 친지에게 문의	여행가이드	인터넷(노트북/테블릿 PC)	여행 가이드북	한국 지역 주민에게 문의	다른 여행자에게 문의
0	전체	NaN	(2402)	59.807625	15.947027	10.092527	5.063249	2.628449	2.394529	1.[illegible]6812
1	월별	1 월	(148)	58.139355	20.034829	3.423321	10.028890	1.437491	4.986406	1.687216
2	NaN	2 월	(153)	54.068679	21.415532	8.350656	4.611392	2.123506	4.824698	0.745962
3	NaN	3 월	(172)	61.271268	15.463879	11.520904	5.530424	1.345833	1.858442	0.844932
4	NaN	4 월	(254)	58.665247	11.778880	13.789756	5.677012	2.836255	3.716246	1.874135

출처	비율(%)
인터넷(스마트폰)	59.8076
지인	15.947
여행가이드	10.0925
인터넷(노트북/테블릿 PC)	5.0[illegible]325
여행 가이드북	2.62845
지역주민	2.39453
다른 여행자	1.49681
없음	1.10973
기타	0.577989
공항/항만 내 안내소	0.441689
공항/항만 이외 안내소	0.440371

'인터넷(스마트폰)' 정보의 활용도가 높다는 것을 알 수 있다.

↓

확인할 수 없는 정보가 발생할 수 있다.

↓

그만큼 확실한 정보 제공 시스템이 필요하다.

c. '방문지 만족도 순위'

초기 데이터 값을 가져와, 원하는 정보 추출을 위해 전처리 진행

방문지 만족도 순위

```
visit_sa = pd.read_excel('방문지만족도순위.xlsx', sep=',', encoding='utf8')
visit_sa.head(5)
```

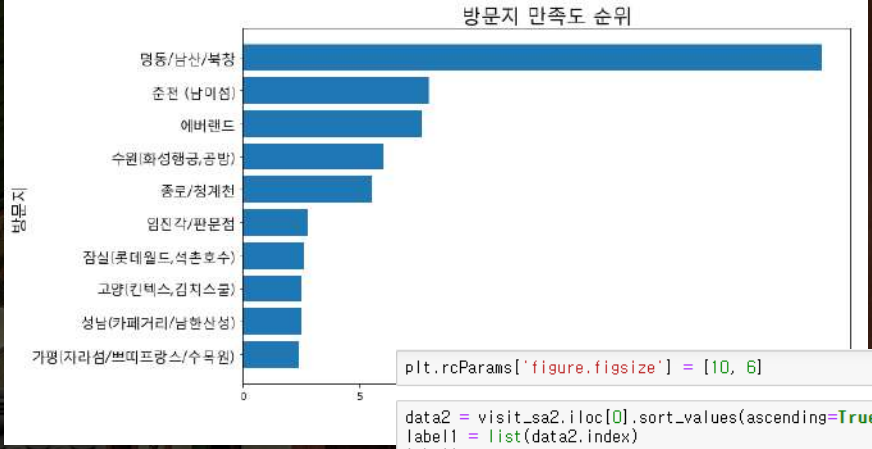
BASE: 전체	Unnamed: 1	사례수	명동/남대문/북창 (숭례문(남대문), 덕수궁, 명동, 한옥마을(남산), 남산/서울타워, 남대문 시장)	춘천 (남이섬)	에버랜드	수원 화성 (수원화성/화성행궁, 공방거리, 수원남문시장)	종로/청계 (청계천, 고궁, 박물관/기념관, 북촌, 삼청동, 인사동)	임진각, 제3땅굴/통일전망대/판문점	잠실 (잠실롯데월드/롯데월드몰/롯데월드타워, 석촌호수, 올림픽공원)	...	홍천	군산	여주	마곡	무주구천동 (무주구천동/리조트, 덕유산, 태권도원, 반디랜드)
전체	NaN	(2402)	24.888979	8.015580	7.709922	6.040829	5.541614	2.795561	2.595033	...	0.011143	0.011112	0.01024	0.009821	0.007779
월별	1 월	(148)	23.268708	4.352043	3.794252	14.116626	5.064088	1.320231	2.264216	...	0.000000	0.000000	0.00000	0.000000	0.000000
NaN	2 월	(153)	22.113245	5.555579	2.993210	8.417999	3.652036	2.847838	0.846843	...	0.000000	0.000000	0.00000	0.000000	0.000000
NaN	3 월	(172)	28.012882	11.016880	7.665919	3.121596	4.676800	5.348704	0.666907	...	0.000000	0.000000	0.00000	0.000000	0.000000
NaN	4 월	(254)	28.224746	10.680398	6.988924	2.261630	8.130095	2.954901	2.678497	...	0.000000	0.000000	0.00000	0.092802	0.000000

원하는 정보 추출

```
visit_sa = visit_sa.drop('BASE:전체',axis=1)
visit_sa1 = visit_sa.rename({'Unnamed: 1':'만족도',
                             '명동/남대문/북창 (숭례문(남대문), 덕수궁, 명동, 한옥마을(남산), 남산/서울타워, 남대문 시장)':
                             '명동/남산/북창',
                             '수원 화성 (수원화성/화성행궁, 공방거리, 수원남문시장)':'수원(화성행궁,공방)',
                             '종로/청계 (청계천, 고궁, 박물관/기념관, 북촌, 삼청동, 인사동)':'종로/청계천',
                             '임진각, 제3땅굴/통일전망대/판문점':'임진각/판문점',
                             '잠실 (잠실롯데월드/롯데월드몰/롯데월드타워, 석촌호수, 올림픽공원)':'잠실(롯데월드,석촌호수)',
                             '고양 (고양 킨텍스, 김치스쿨, 원마운트)':'고양(킨텍스,김치스쿨)',
                             '성남 (판교/카페거리/남한산성)':'성남(카페거리/남한산성)',
                             '가평 (자라섬/쁘띠프랑스/아침고요수목원)':'가평(자라섬/쁘띠프랑스/수목원)'},axis=1)
visit_sa1 = visit_sa1.fillna('비율')
visit_sa1 = visit_sa1.drop('사례수',axis=1)
visit_sa2 = visit_sa1.iloc[0:1,1:11]
visit_sa2.head(5)
```

	명동/남산/북창	춘천 (남이섬)	에버랜드	수원(화성행궁,공방)	종로/청계천	임진각/판문점	잠실(롯데월드,석촌호수)	고양(킨텍스,김치스쿨)	성남(카페거리/남한산성)	가평(자라섬/쁘띠프랑스/수목원)
0	24.888979	8.01558	7.709922	6.040829	5.541614	2.795561	2.595033	2.520683	2.510284	2.353539

최종 데이터 값을 바 그래프로 표현




```
plt.rcParams['figure.figsize'] = [10, 6]
```

```
data2 = visit_sa2.iloc[0].sort_values(ascending=True)
label1 = list(data2.index)
label1
```

```
['가평(자라섬/쁘띠프랑스/수목원)',
 '성남(카페거리/남한산성)',
 '고양(킨텍스,김치스쿨)',
 '잠실(롯데월드,석촌호수)',
 '임진각/판문점',
 '종로/청계천',
 '수원(화성행궁,공방)',
 '에버랜드',
 '춘천 (남이섬)',
 '명동/남산/북창']
```

```
index1 = np.arange(len(label1))
index1
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

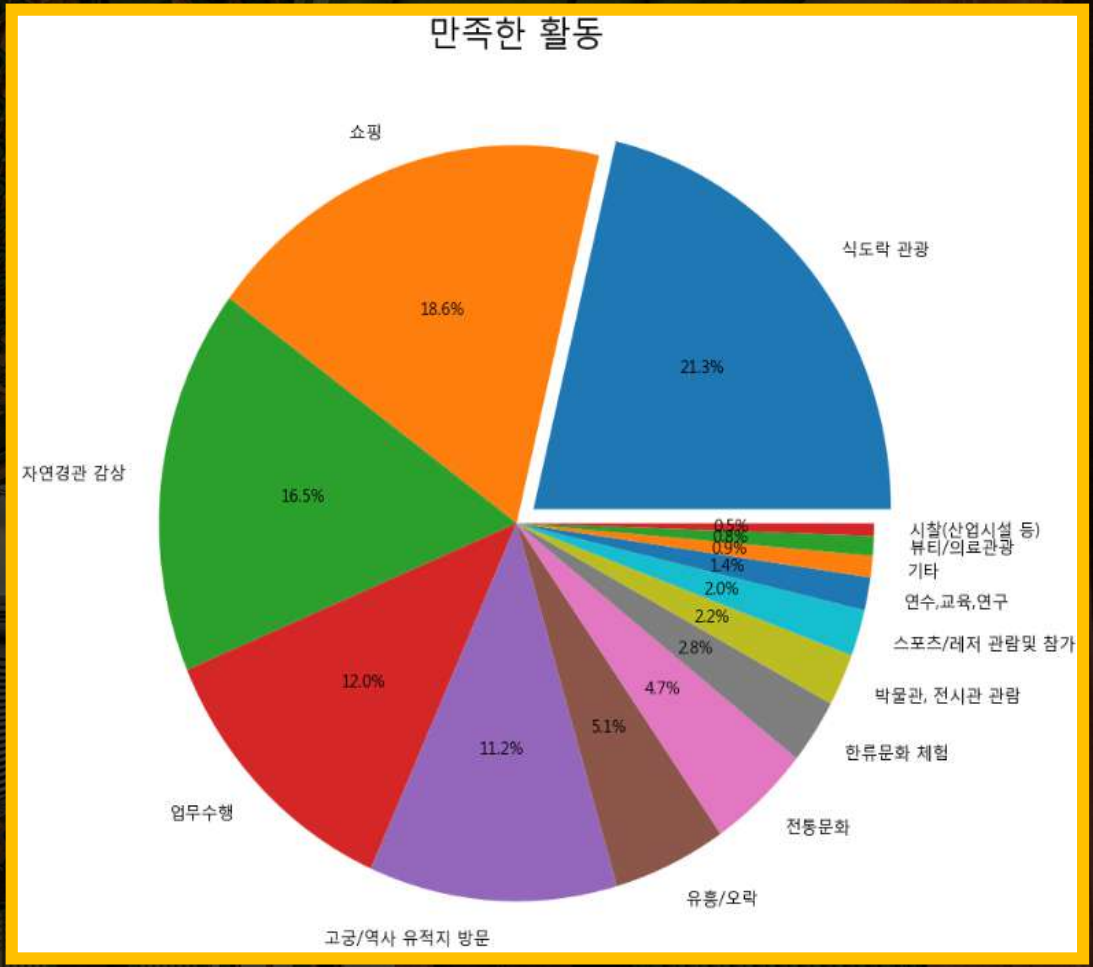
d.'만족한 활동'

만족한 활동

```
sa_act = pd.read_excel('만족한활동.xlsx', sep=',', encoding='utf8', skiprows=1)
sa_act.head(5)
```

	BASE: 전체	Unnamed: 1	사례수	식도락 관광	쇼핑	자연경관 감상	업무수행 (회의,학술대회,박람회등 참여)	고궁/역사 유적지 방문	유흥/오락	전통문화 체험(한복체험, 태권도, 한국음식 만들기 등)	K-POP/ 한류스타 관련 공연장및 드라마/ 영화 촬영지 방문	박물관, 전시관 관람	스포츠/ 레저 관람및 참가	연수,교육,연구	기타
0	전체	NaN	(2402)	21.291362	18.556565	16.490954	11.981527	11.211760	5.129335	4.653075	2.772031	2.227012	1.988288	1.407610	0.915050
1	월별	1 월	(148)	20.427184	22.582020	14.448921	12.228720	8.383563	2.111475	5.160659	4.268673	0.223310	7.412720	0.652164	1.037408
2	NaN	2 월	(153)	25.952319	22.853767	13.492904	14.615652	9.747668	3.733899	3.222236	1.509380	0.519844	2.079940	1.312004	0.366722
3	NaN														
4	NaN														

```
# 만족한 활동
pie2[' '].plot(kind='pie',figsize=(15,15),
                autopct = '%1.1f%%',fontsize = 15, explode=(0.06,0,0,0,0,0,0,0,0,0,0,0,0,0))
plt.title('만족한 활동', fontsize=30)
plt.show()
```



3. 이미지 분류

CNN
Alex Net
Mobile Net
Dense Net

모델 1. CNN

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65600
dense_1 (Dense)	(None, 2)	130

Total params: 122,050
Trainable params: 122,050
Non-trainable params: 0

"이진분류"

Accuracy 0.9631

모델 1. CNN

```
_____________________________________________________________
Layer (type) Output Shape Param #
=============================================================
conv2d (Conv2D) (None, 30, 30, 32) 896
_____________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32) 0
_____________________________________________________________
conv2d_1 (Conv2D) (None, 13, 13, 64) 18496
_____________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64) 0
_____________________________________________________________
conv2d_2 (Conv2D) (None, 4, 4, 64) 36928
_____________________________________________________________
flatten (Flatten) (None, 1024) 0
_____________________________________________________________
dense (Dense) (None, 64) 65600
_____________________________________________________________
dense_1 (Dense) (None, 30) 1950
=============================================================
Total params: 123,870
Trainable params: 123,870
Non-trainable params: 0
_____________________________________________________________
```

~~Accuracy 0.5157~~

모델 2. Alex Net

```
history = model.fit(x_train, y_train, epochs=2, validation_split=0.3)
result = model.evaluate(x_test, y_test)
print('loss (cross-entropy) :', result[0])
print('test accuracy :', result[1])
```

```
Train on 1120 samples, validate on 480 samples
Epoch 1/2
1120/1120 [==============================] - 75s 67ms/sample - loss: 4.6052 - acc: 0.7143 - val_loss: 16.1181
- val_acc: 0.0000e+00
Epoch 2/2
1120/1120 [==============================] - 74s 66ms/sample - loss: 4.6052 - acc: 0.7143 - val_loss: 16.1181
- val_acc: 0.0000e+00
407/407 [==============================] - 5s 14ms/sample - loss: 8.0789 - acc: 0.4988
loss (cross-entropy) : 8.078850650083542
test accuracy : 0.4987715
```

Accuracy : 0.498

모델 3. Mobile Net

```
history = model.fit(train_dataset,
                    epochs=initial_epochs,
                    validation_data=validation_dataset)
```

```
Epoch 1/10
750/750 [==============================] - 288s 384ms/step - loss: 3.6618 - accuracy: 0.0335 - val_loss: 3.47
79 - val_accuracy: 0.0392
Epoch 2/10
750/750 [==============================] - 287s 383ms/step - loss: 3.6737 - accuracy: 0.0451 - val_loss: 3.32
66 - val_accuracy: 0.0445
Epoch 3/10
750/750 [==============================] - 287s 382ms/step - loss: 3.5583 - accuracy: 0.0501 - val_loss: 3.15
36 - val_accuracy: 0.0464
Epoch 4/10
750/750 [==============================] - 287s 383ms/step - loss: 3.3684 - accuracy: 0.0545 - val_loss: 3.22
26 - val_accuracy: 0.0466
Epoch 5/10
750/750 [==============================] - 287s 383ms/step - loss: 3.3565 - accuracy: 0.0517 - val_loss: 3.18
35 - val_accuracy: 0.0472
Epoch 6/10
750/750 [==============================] - 286s 382ms/step - loss: 3.3611 - accuracy: 0.0665 - val_loss: 3.13
39 - val_accuracy: 0.0648
Epoch 7/10
750/750 [==============================] - 287
99 - val_accuracy: 0.4307
Epoch 8/10
750/750 [==============================
11 - val_accuracy: 0.4461
Epoch 9/10
750/750 [==============================] - 290s 386ms/step - loss: 3.8062 - accuracy: 0.4128 - val_loss: 3.58
32 - val_accuracy: 0.4784
Epoch 10/10
750/750 [==============================] - 290s 386ms/step - loss: 3.8862 - accuracy: 0.4411 - val_loss: 3.62
04 - val_accuracy: 0.4870
```

Accuracy : 0.4411

우리의 데이터는 위 모델과 맞지 않다.

모델 4.
Dense Net

이 모델의 장점으로는 이미지에서 저수준의 특징들이 잘 보존되고,
gradient가 수월하게 흘러 gradient vanishing 문제가 발생하지 않으며,
깊이에 비해 파라미터 수가 적기에 연산량이 절약됨과 동시에
적은 데이터셋에서도 비교적 잘 학습이 된다는 점이 있다.

모델 4.
Dense Net

Layers		Output Size	X	Kernel Size
InputLayer	-	(32, 32, 3)		
Convolution	Conv2D	(16, 16, 64)		7x7 Stride 2
	BatchNormalization	(16, 16, 64)		
	Activation - Relu	(16, 16, 64)		
MaxPooling2D	-	(8, 8, 64)		3x3
Dense Block	BatchNormalization	(8, 8, 64)	X 6	
	Activation - Relu	(8, 8, 64)		
	Conv2D	(8, 8, 128)		
	BatchNormalization	(8, 8, 128)		
	Activation - Relu	(8, 8, 128)		
	Conv2D	(8, 8, 32)		
	Concatenate	(8, 8, 96)		
		...		
Transition Block	BatchNormalization	(8, 8, 256)		
	Activation - Relu	(8, 8, 256)		
	Conv2D	(8, 8, 128)		1x1
	AveragePooling2D	(4, 4, 128)		2x2 stride 2
Dense Block	-	(4, 4, 896)	X 24	
Transition Block	-	(2, 2, 448)		
Dense Block	-	(2, 2, 960)	X 16	
Classification Layer	GlobalAveragePooling2D	(960)		
	Dense - Softmax	(10)		

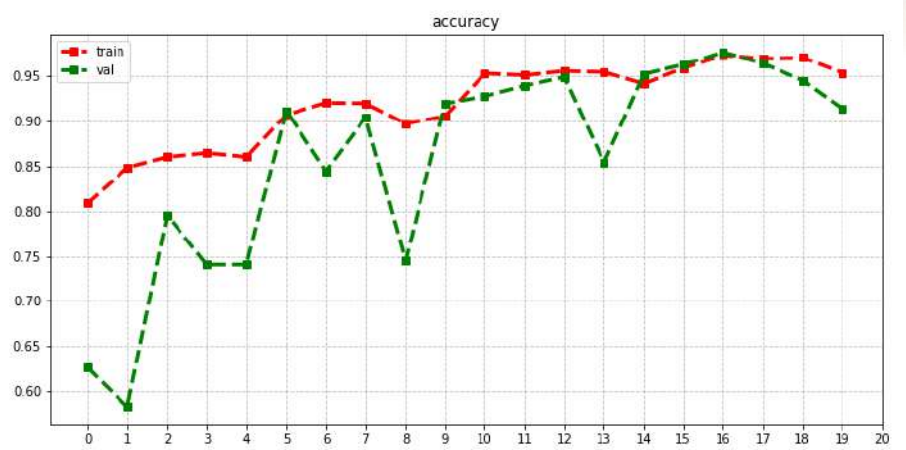
모델 4. Dense Net

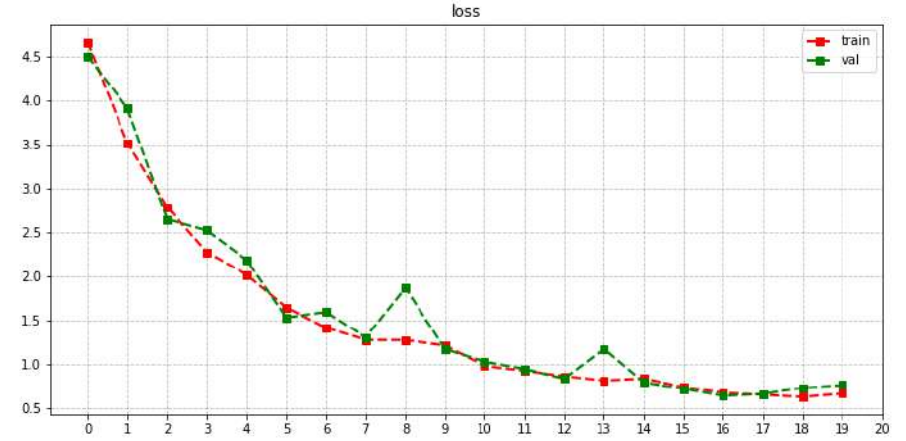
```
Epoch 1/20
107/107 [===] - 341s 3s/step - loss: 4.6685 - acc: 0.8087 - val_loss: 4.5097 - val_acc: 0.6262
Epoch 2/20
107/107 [===] - 326s 3s/step - loss: 3.5253 - acc: 0.8485 - val_loss: 3.9190 - val_acc: 0.5829
Epoch 3/20
107/107 [===] - 340s 3s/step - loss: 2.7878 - acc: 0.8602 - val_loss: 2.6501 - val_acc: 0.7956
.
.
Epoch 15/20
107/107 [===] - 321s 3s/step - loss: 0.8389 - acc: 0.9414 - val_loss: 0.7940 - val_acc: 0.9521
Epoch 16/20
107/107 [===] - 317s 3s/step - loss: 0.7416 - acc: 0.9587 - val_loss: 0.7194 - val_acc: 0.9626
Epoch 17/20
107/107 [===] - 314s 3s/step - loss: 0.6723 - acc: 0.9731 - val_loss: 0.6390 - val_acc: 0.9766
Epoch 18/20
107/107 [===] - 320s 3s/step - loss: 0.6521 - acc: 0.9694 - val_loss: 0.6617 - val_acc: 0.9638
Epoch 19/20
107/107 [===] - 331s 3s/step - loss: 0.6267 - acc: 0.9704 - val_loss: 0.7294 - val_acc: 0.9451
Epoch 20/20
107/107 [===] - 325s 3s/step - loss: 0.6643 - acc: 0.9534 - val_loss: 0.7638 - val_acc: 0.9136
```

```
score = dense.evaluate(x_test, y_test_reshape)

print('test score:', score[0])
print('test accuracy:', score[1])
```

```
100/100 [==========================] - 2s 22ms/sample - loss: 3.1081 - acc: 0.5500
test score: 3.1080543422698974
test accuracy: 0.55
```





모델 4. Dense Net

DenseNet - Image size:32, Epoch:100

train acc

```
Epoch 96/100
108/108 [==============================] - 11s 98ms/step - loss: 0.2512 - acc: 0.9942
108/108 [==============================] - 83s 767ms/step - loss: 0.2532 - acc: 0.9942 - val_loss: 0.2512 - val_acc: 0.9942
Epoch 97/100
108/108 [==============================] - 10s 97ms/step - loss: 0.2344 - acc: 1.0000
108/108 [==============================] - 83s 769ms/step - loss: 0.2714 - acc: 0.9892 - val_loss: 0.2344 - val_acc: 1.0000
Epoch 98/100
108/108 [==============================] - 11s 100ms/step - loss: 0.3647 - acc: 0.9605
108/108 [==============================] - 88s 810ms/step - loss: 0.2562 - acc: 0.9939 - val_loss: 0.3647 - val_acc: 0.9605
Epoch 99/100
108/108 [==============================] - 10s 95ms/step - loss: 0.2742 - acc: 0.9860
108/108 [==============================] - 86s 794ms/step - loss: 0.2641 - acc: 0.99[illegible] - val_loss: 0.2742 - val_acc: 0.9860
Epoch 100/100
108/108 [==============================] - 10s 95ms/step - loss: 0.2536 - acc: 0.9930
108/108 [==============================] - 81s 746ms/step - loss: 0.2836 - acc: 0.9840 - val_loss: 0.2536 - val_acc: 0.9930
```

```
model_filename = "h_file/denseNet_top10_food.h5"
dense.load_weights(model_filename)

predicted = [np.argmax(i) for i in dense.predict(x_test)]
```

```
score = dense.evaluate(x_test, y_test_reshape)
```

```
100/100 [==============================] - 3s 29ms/sample - loss: 2.2571 - acc: 0.5900
```

모델 4. Dense Net

모델 4. Dense Net

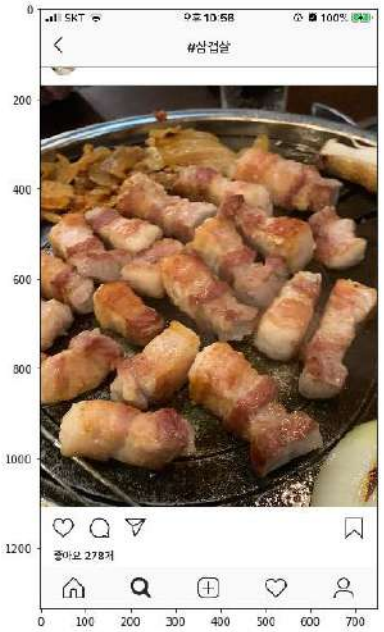
ok! good job!

```
view_img('predict_img/', 'img1.png')
data = get_img_array('predict_img/img1.png')
predict_result(dense, data)
```

('삼겹살', 'Sangyeobsal')

```
view_img('predict_img/', 'img2.png')
data = get_img_array('predict_img/img2.png')
predict_result(dense, data)
```

('삼겹살', 'Sangyeobsal')




```
view_img('predict_img/', 'img3.png')
data = get_img_array('predict_img/img3.png')
predict_result(dense, data)
```

('짬뽕', 'Jjamppong')

```
view_img('predict_img/', 'img4.png')
data = get_img_array('predict_img/img4.png')
predict_result(dense, data)
```

('짬뽕', 'Jjamppong')

```
view_img('predict_img/', 'img5.png')
data = get_img_array('predict_img/img5.png')
predict_result(dense, data)
```

('짜장면', 'Jjajangmyeon')

```
view_img('predict_img/', 'img6.png')
data = get_img_array('predict_img/img6.png')
predict_result(dense, data)
```

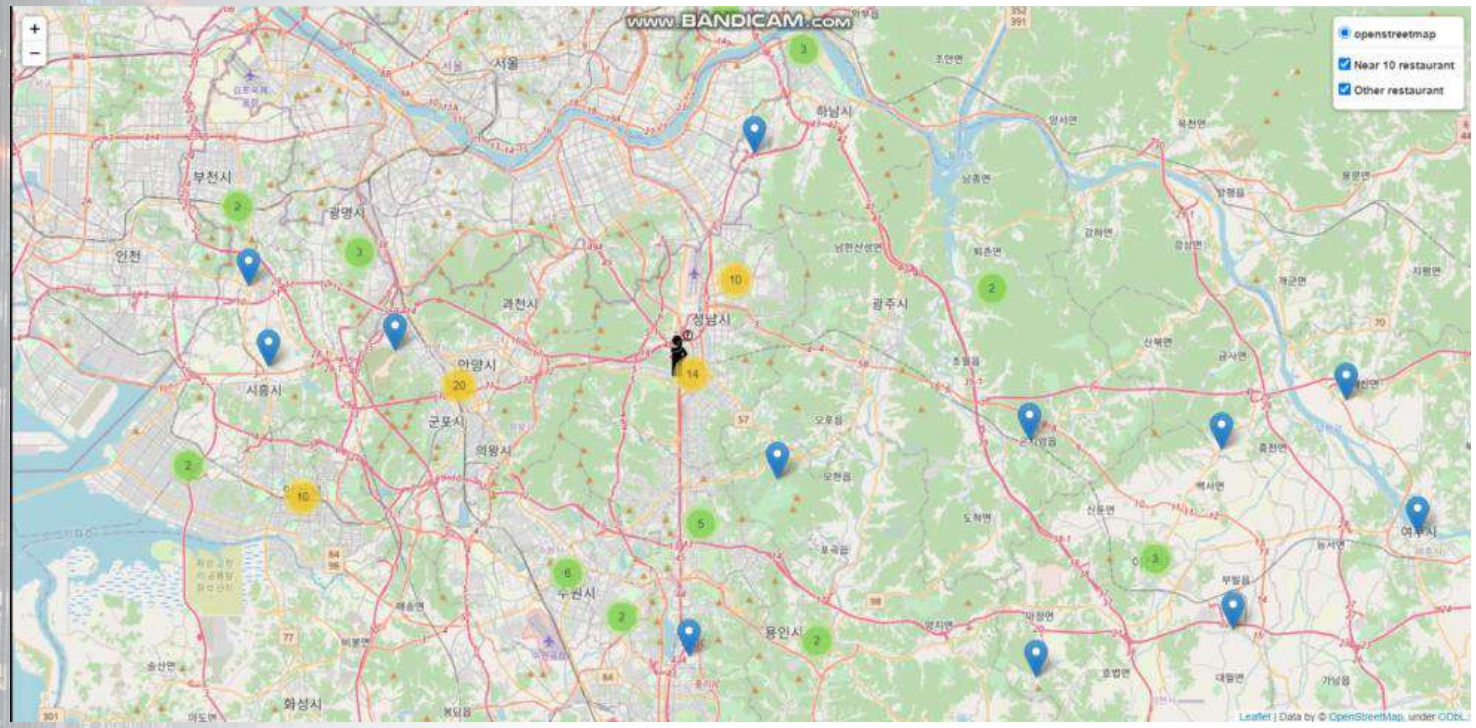
('곰탕_설렁탕', 'Gomtang_Seolleongtang')

4. 지도 시각화

경기도 공공 데이터를 통해 맛집, 모범음식점의 정보를 수집한다. 이미지 분류를 통해 얻어낸 음식명과 매치하여 음식점의 위치 정보 및 평점을 제공한다. 그리고 해당 음식점으로 길찾기를 할 수 있다.

구글 번역을 통해 식당명 영어 표기

	시군명	식당	식당_영문명	매칭 음식	위도	경도
0	가평군	가평(춘천방향)휴게소 한식당	gapyeong(chuncheonbanghyang)hyugeso hansigdang		37.70159	127.5465
1	가평군	해낙전 철판	haenagjeon cheolpan	닭갈비	37.7171	127.4585
2	가평군	일가가마솥설렁탕	ilgagamasotseolleongtang	곰탕_설렁탕	37.82276	127.34
3	가평군	이경숙할머니음식점	igyeongsughalmeonieumsigjeom	장어구이, 매운탕	37.70233	127.3965
4	가평군	늘 봄	neul bom	갈비구이	37.83476	127.51
5	가평군	미락무교동낚지	milag mugyodongnagji		37.81535	127.5016
6	가평군	가평축협 한우명가	gapyeongchughyeob han-umyeong-ga	갈비구이	37.81584	127.5161



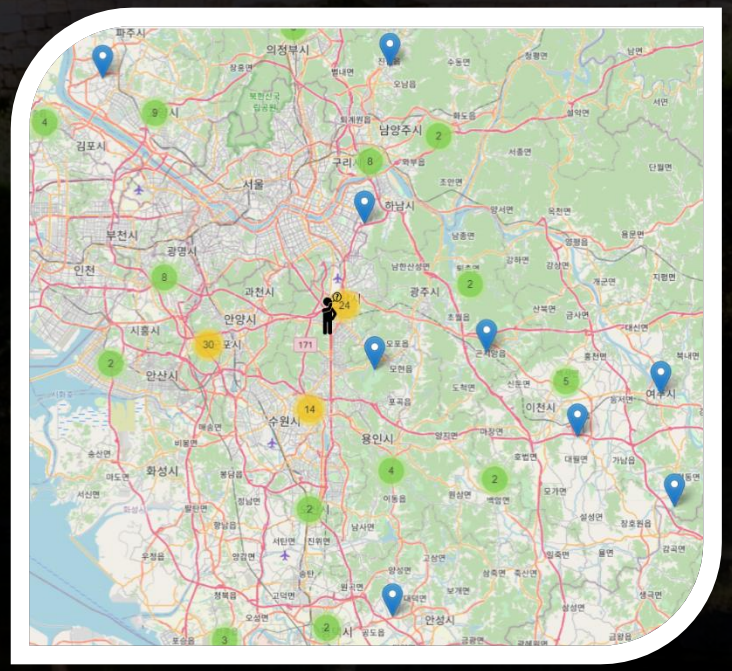
경기도 공공 데이터를 통해 맛집, 모범음식점의 정보를 수집한다. 이미지 분류를 통해 얻어낸 음식명과 매치하여 음식점의 위치 정보 및 평점을 제공한다. 그리고 해당 음식점으로 길찾기를 할 수 있다.

이미지를 통해 음식 이름 및 맛집 정보를 알 수 있다

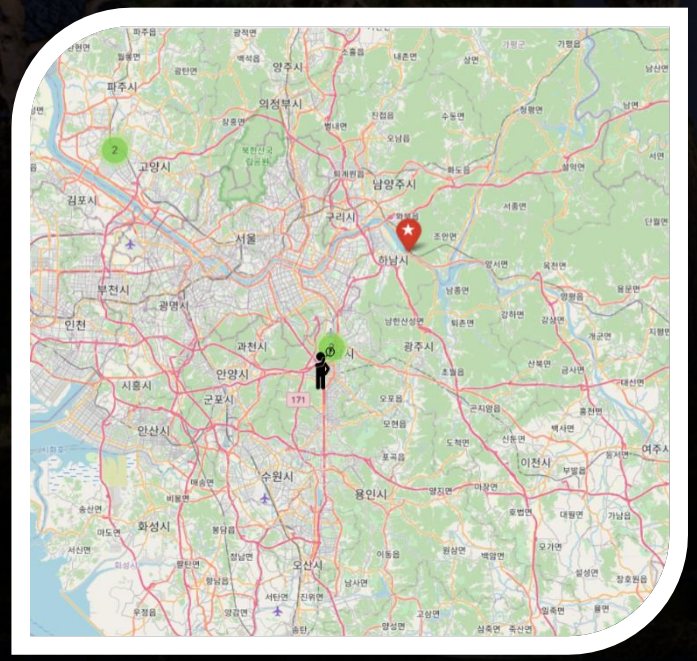
5. 한계점 및 발전방향

한계점

음식점 별 대표음식 종류 개수 데이터 차이
(대표음식 개수 차이)



“삼겹살”



“짬뽕”

한계점

공공데이터 – 맛집, 모범음식점 데이터의 한계점

- 폐점 데이터 포함 -> 업데이트 필요
- 음식점 이름에 특수기호(ex. ?) 포함

이미지 데이터의 한계점

- 이미지의 불명확성
- 중복된 이미지 다소 존재
- 다른 카테고리의 음식 사진이 포함

발전방향

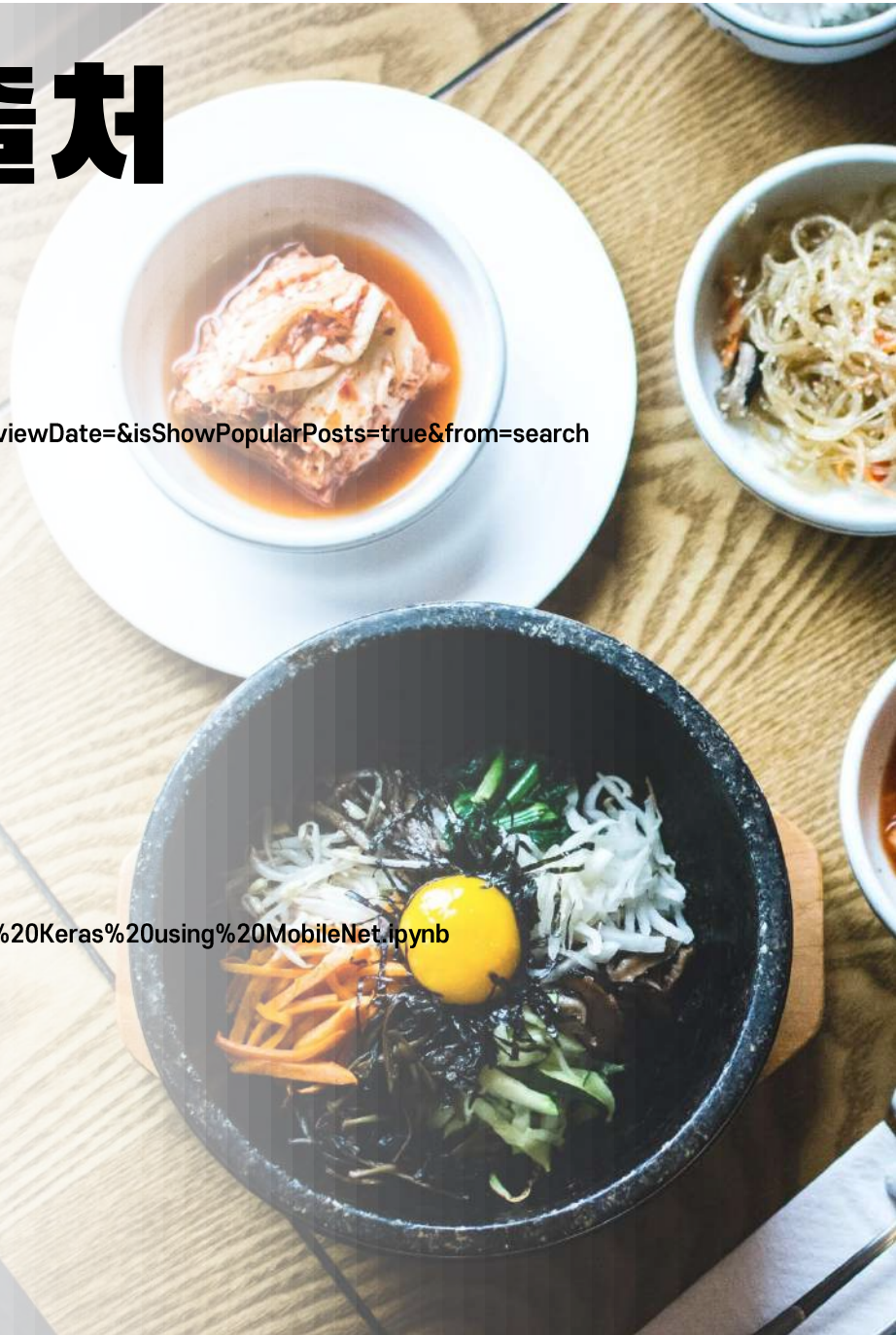
- 맛에 대한 세부화

- 어플, 웹사이트 구현

6. 참고 문헌 및 데이터 출처

논문

- http://blog.naver.com/PostView.nhn?blogId=chacagea&logNo=221692490366&parentCategoryNo=&categoryNo=32&viewDate=&isShowPopularPosts=true&from=search
- https://blog.naver.com/chacagea/221582912200
- Densely Connected Convolutional Networks (Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger)
- https://arxiv.org/abs/1608.06993

웹사이트

- https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299
- https://github.com/ferhat00/Deep-Learning/blob/master/Transfer%20Learning%20CNN/Transfer%20Learning%20in%20Keras%20using%20MobileNet.ipynb
- https://kau-deeperent.tistory.com/59
- https://www.tensorflow.org/tutorials/images/transfer_learning?hl=ko
- https://dailyheumsi.tistory.com/85
- https://blog.naver.com/jlmaj_sim/221358084087
- https://hoya012.github.io/blog/DenseNet-Tutorial-1/