



Day50; 20221116

날짜	@2022년 11월 16일
유형	@2022년 11월 16일
태그	

GitHub - u8yes/AI

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/u8yes/ai>

u8yes/AI



1 Contributor 0 Issues 0 Stars 0 Forks

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/791b0f55-4281-4fd4-95b4-e1d52e3d8fc3/03_SVM_20221114.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1c48a9ee-1d7c-491f-8905-40edfa0dc36a/04_decision-tree_teacher_version.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d88891d8-c8f5-4610-ad46-1e18e7e2f072/04_decision-tree_20221116.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b5ba36a6-a735-4735-a1cf-5af7434ad6b9/district_dict_list.txt

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5ccf0e13-f267-4bb2-8a53-671d10369c47/seoul.pdf>

이진 분류인 이유는 2개로 나누어지니까.

빅분기 실기 - 머신러닝)

```
# [23] 다양한 모델을 만들고 성능을 출력하는 함수 작성
def make_models(xtrain, xtest, ytrain, ytest):
    model1 = LogisticRegression(max_iter=5000).fit(xtrain, ytrain)
    print('model1', get_scores(model1, xtrain, xtest, ytrain, ytest))

    for k in range(1, 10):
        model2 = KNeighborsClassifier(k).fit(xtrain, ytrain)
        print('model2', k, get_scores(model2, xtrain, xtest, ytrain, ytest))

    # overfitting ??
    model3 = DecisionTreeClassifier().fit(xtrain, ytrain)
    for d in range(3, 8):
        model3 = DecisionTreeClassifier(max_depth=d).fit(xtrain, ytrain)
        print('model3', d, get_scores(model3, xtrain, xtest, ytrain, ytest))
```

```
# overfitting ??
model4 = RandomForestClassifier().fit(xtrain, ytrain)
for d in range(3, 8):
    model4 = RandomForestClassifier(500, max_depth=d).fit(xtrain, ytrain)
    print('model4', d, get_scores(model4, xtrain, xtest, ytrain, ytest))


model5 = XGBClassifier().fit(xtrain, ytrain)
print('model5', get_scores(model5, xtrain, xtest, ytrain, ytest))
```

```
# [24] X를 train 용도, submission 용도로 나누고, Y를 1차원으로 바꿈
# (StandardScaler 적용)
def get_data(dfX, Y):
    X = dfX.drop(columns=['cust_id'])
    X_use = X.iloc[:3500, :]
    X_submission = X.iloc[3500:, :]
    Y1 = Y['gender']
    scaler = StandardScaler()
    X1_use = scaler.fit_transform(X_use)
    X1_submission = scaler.transform(X_submission)
    print(X1_use.shape, X1_submission.shape, Y1.shape)
    return X1_use, X1_submission, Y1
```

$2^0 = 1$ 이다.

2의 0승이 1인 이유

자 오늘은 2의 0승이 1이 되는 이유를 간단히 보여 드리겠습니다. 우선 아래에서 보는 것과 같이 2의 3승 나누기 2의 2승은 2의 (3-2)승, 즉 2의 1승입니다. 이번엔 4 나누기 4를 한 번 해 보겠습니다. 일단 분모와 분자가 같으니 답은 1이고 분모와 분자를 위의 식처럼 거듭제곱근

 <https://johnleedu.tistory.com/22>

$$2^1 = 2^{(1+0)} = 2^1 \times 2^0$$

$$\frac{2^2}{2^2} = 2^{(2-2)} = 2^0$$

RBF 알고리즘에 C와 gamma를 넣어서 GridSearchCV에 넣어서 볼 수 있음

지니계수는 이진분류일 때만 적용 가능.

2진 분류에 적용과 상관없이 엔트로피 사용한다.

지니 계수(Gini coefficient)

- 특징에 의한 분리가 이진 분류로 나타날 경우 지니 계수를 사용할 수 있음.
 - 사이킷런의 의사결정 트리는 CART(classification and regression tree) 타입의 의사결정 트리.
 - CART는 트리의 노드마다 특징을 이진 분류하는 특징이 있기에 사이킷런은 트리를 구성할 때 기본적으로 지니 계수를 사용.
 - 지니 계수의 특징
 - 특징이 항상 이진 분류로 나뉠 때 사용됨.
 - 지니 계수가 높을수록 순도가 높음.
 - 순도가 높다는 뜻: 한 그룹에 모여있는 데이터들의 속성들이 많이 일치한다는 뜻.
 - 불순도가 높다는 뜻: 한 그룹에 여러 속성의 데이터가 많이 섞여 있다는 뜻.
 - 의사결정 트리 알고리즘은 지니 계수가 높은 특징으로 의사결정 트리 노드를 결정.
-

optimization

영어사

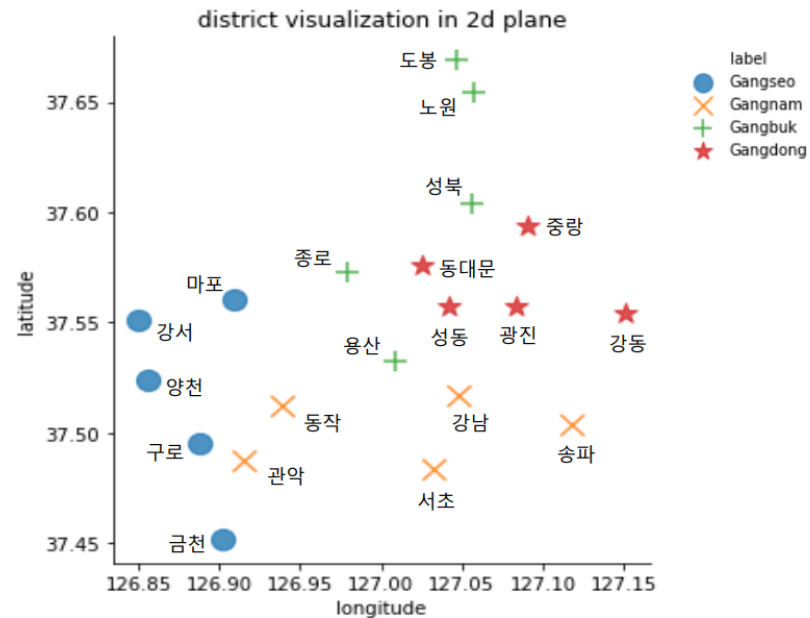
- 1 최대한의 이용[활용].
- 2 최적화(最適化), 가장 적합한 조건, 가장 효과적인 상태.
- 3 (함수의) 최적화.

영어사전 다른 뜻 1

다중 분류

➤ 서울시 행정구역에 대한 다중 분류 데이터 시각화

- 데이터를, 특징을 바탕으로 한 공간에 데이터 특징을 시각화함으로써, 우리는 머신러닝 학습에 필요한 특징과 불필요한 특징을 쉽게 구분 지을 수 있고, 데이터의 패턴을 눈으로 쉽게 파악할 수 있음.



빅분기 실기 - 머신러닝)

DataFrame에 저장된 값을 치환하기 위해서 **replace**를 사용합니다.

replace를 사용해 문자열을 치환하는 경우 검색 문자열과 완전히 일치하는 문자만 치환이 됩니다.

일부분 일치하는 문자열은 변환되지 않습니다.

문자열 일부만 치환하고 싶은 경우는 **regex=True**를 설정해 **정규 표현식**으로 문자열 치환을 원하는 부분만 할 수 있습니다.

정규 표현식을 사용하지 않고 문자열을 치환한 결과를 먼저 보겠습니다.

```
import pandas as pd

df = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie', 'Dave', 'Ellen', 'Frank'],
                   'age': [24, 42, 18, 68, 24, 30],
                   'state': ['NV', 'CA', 'CA', 'TX', 'CA', 'NV'],
                   'point': [64, 24, 70, 70, 88, 57]})

print(df)
#      name  age state point
# 0   Alice   24   NV    64
# 1    Bob   42   CA    24
# 2  Charlie   18   CA    70
# 3    Dave   68   TX    70
# 4   Ellen   24   CA    88
# 5   Frank   30   NV    57

print(df.replace('li', 'LI'))
#      name  age state point
```

```
print(df.replace('li', 'LI'))
#      name  age state point
# 0   Alice   24   NY    64
# 1    Bob    42   CA    24
# 2  Charlie   18   CA    70
# 3    Dave   68   TX    70
# 4   Ellen   24   CA    68
# 5   Frank   30   NY    57
```

DataFrame에 저장된 **li**라는 문자를 **LI**로 치환하려고 했습니다.

하지만 문자열은 치환되지 않았습니다.

정규 표현식을 사용해 치환하겠습니다.

```
import pandas as pd

df = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie', 'Dave', 'Ellen', 'Frank'],
                  'age': [24, 42, 18, 68, 24, 30],
                  'state': ['NY', 'CA', 'CA', 'TX', 'CA', 'NY'],
                  'point': [64, 24, 70, 70, 68, 57]})

print(df.replace('(.*?)li(.*?)', r'##1LI##2', regex=True))
#      name  age state point
# 0   ALice   24   NY    64
# 1    Bob    42   CA    24
# 2 CharLi   18   CA    70
# 3    Dave   68   TX    70
# 4   Ellen   24   CA    68
# 5   Frank   30   NY    57
```

문자 **li**가 **LI**로 치환되었습니다.

replace를 정규 표현식으로 사용하는 경우 작성 방법은 **re.sub()** 정규 표현식 모듈과 동일합니다.

What is regex true in python?



Regular expressions (regex) are essentially **text patterns that you can use to automate searching through and replacing elements within strings of text**. 2020. 1. 7.

<https://www.dataquest.io/blog/regular-expressions-data...>

Decision Tree

클라이언트에게 의뢰를 받았을 때 보통 대학 연구실에서 프로젝트를 맡아 '문제 정의'를 한다.

라이브러리 импорт_20221116

- 실습에 필요한 라이브러리를 импорт.

```
2]: import numpy as np
import pandas as pd

#sklearn 모델의 동일한 결과 출력을 위해 선언합니다.
np.random.seed(5)
```

문제 정의

- 서울 지역(구)의 경도와 위도 정보를 사용하여, 임의로 입력된 지역(등)을 강동, 강서, 강남, 강북으로 분류해보는 예제.

데이터 수집

- 아래는 서울의 대표적인 구(district) 위치 데이터임.
- 구(district) 정보는 학습에 사용할.

컬럼 주석

- district : 행정구역(서초구, 송파구, 용산구 등, 서울의 단위 지역 분류)
- dong : 구(district)보다 작은 행정구역(개치동, 도곡동, 암사동 등, 서울의 소단위 분류)
- latitude : 위도
- longitude : 경도
- label : 한강 기준으로 동/서/남/북으로 구분한 지역 명칭

문제 정의 → 데이터 수집 → 컬럼 주석

latitude

미국식[ˈlætɪtuːd] <영국식[ˈlætɪtjuːd]>

(영사)

1 (Abbr.) lat.. 위도 (→longitude)

2 (위도상으로 본) 지역[지방]

the northern latitudes <영>

북반구 지역

3 (선택,행동 방식의) 자유 (=liberty)

[영어사전 결과 더보기](#)

longitude

미국식[ˈlɒndʒətuːd] <영국식[ˈlɒŋɡɪtjuːd; ˈlɒndʒɪtjuːd]>

(영사)

(Abbr.) long.. 경도 (→latitude)

the longitude of the island <영>

그 섬의 경도

[영어사전 결과 더보기](#)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c616df37-c642-40ed-afb6-fab9a2235834/district_dict_list.txt

전처리 과정

```
district_dict_list = [
    {'district': 'Gangseo-gu', 'latitude': 37.551000, 'longitude': 126.849500, 'label': 'Gangseo'},
```



```

{'district': 'Yangcheon-gu', 'latitude': 37.52424, 'longitude': 126.855396, 'label': 'Gangseo'},
{'district': 'Guro-gu', 'latitude': 37.4954, 'longitude': 126.8874, 'label': 'Gangseo'},
{'district': 'Geumcheon-gu', 'latitude': 37.4519, 'longitude': 126.9020, 'label': 'Gangseo'},
{'district': 'Mapo-gu', 'latitude': 37.560229, 'longitude': 126.908728, 'label': 'Gangseo'},

{'district': 'Gwanak-gu', 'latitude': 37.487517, 'longitude': 126.915065, 'label': 'Gangnam'},
{'district': 'Dongjak-gu', 'latitude': 37.5124, 'longitude': 126.9393, 'label': 'Gangnam'},
{'district': 'Seocho-gu', 'latitude': 37.4837, 'longitude': 127.0324, 'label': 'Gangnam'},
{'district': 'Gangnam-gu', 'latitude': 37.5172, 'longitude': 127.0473, 'label': 'Gangnam'},
{'district': 'Songpa-gu', 'latitude': 37.503510, 'longitude': 127.117898, 'label': 'Gangnam'},

{'district': 'Yongsan-gu', 'latitude': 37.532561, 'longitude': 127.008605, 'label': 'Gangbuk'},
{'district': 'Jongro-gu', 'latitude': 37.5730, 'longitude': 126.9794, 'label': 'Gangbuk'},
{'district': 'Seongbuk-gu', 'latitude': 37.603979, 'longitude': 127.056344, 'label': 'Gangbuk'},
{'district': 'Nowon-gu', 'latitude': 37.6542, 'longitude': 127.0568, 'label': 'Gangbuk'},
{'district': 'Dobong-gu', 'latitude': 37.6688, 'longitude': 127.0471, 'label': 'Gangbuk'},

{'district': 'Seongdong-gu', 'latitude': 37.557340, 'longitude': 127.041667, 'label': 'Gangdong'},
{'district': 'Dongdaemun-gu', 'latitude': 37.575759, 'longitude': 127.025288, 'label': 'Gangdong'},
{'district': 'Gwangjin-gu', 'latitude': 37.557562, 'longitude': 127.083467, 'label': 'Gangdong'},
{'district': 'Gangdong-gu', 'latitude': 37.554194, 'longitude': 127.151405, 'label': 'Gangdong'},
{'district': 'Jungrang-gu', 'latitude': 37.593684, 'longitude': 127.090384, 'label': 'Gangdong'}
]

```

```

train_df = pd.DataFrame(district_dict_list) # 테이블 형으로 제공해주는 것이 가장 먼저다.
train_df = train_df[['district', 'longitude', 'latitude', 'label']]
train_df

```

```
]:
```

	district	longitude	latitude	label
0	Gangseo-gu	126.849500	37.551000	Gangseo
1	Yangcheon-gu	126.855396	37.524240	Gangseo
2	Guro-gu	126.887400	37.495400	Gangseo
3	Geumcheon-gu	126.902000	37.451900	Gangseo
4	Mapo-gu	126.908728	37.560229	Gangseo
5	Gwanak-gu	126.915065	37.487517	Gangnam
6	Dongjak-gu	126.939300	37.512400	Gangnam
7	Seocho-gu	127.032400	37.483700	Gangnam
8	Gangnam-gu	127.047300	37.517200	Gangnam
9	Songpa-gu	127.117898	37.503510	Gangnam
10	Yongsan-gu	127.008605	37.532561	Gangbuk
11	Jongro-gu	126.979400	37.573000	Gangbuk
12	Seongbuk-gu	127.056344	37.603979	Gangbuk
13	Nowon-gu	127.056800	37.654200	Gangbuk
14	Dobong-gu	127.047100	37.668800	Gangbuk
15	Seongdong-gu	127.041667	37.557340	Gangdong
16	Dongdaemun-gu	127.025288	37.575759	Gangdong
17	Gwangjin-gu	127.083467	37.557562	Gangdong
18	Gangdong-gu	127.151405	37.554194	Gangdong
19	Jungrang-gu	127.090384	37.593684	Gangdong

- 아래는 서울의 대표적인 동 위치 데이터.
- 동 정보는 테스트 시 사용하도록 함.

```

dong_dict_list = [
    # 강서구
    {'dong': 'Gaebong-dong', 'latitude': 37.489853, 'longitude': 126.854547, 'label': 'Gangseo'},
    {'dong': 'Gochuk-dong', 'latitude': 37.501394, 'longitude': 126.859245, 'label': 'Gangseo'},
    {'dong': 'Hwagok-dong', 'latitude': 37.537759, 'longitude': 126.847951, 'label': 'Gangseo'},
    {'dong': 'Banghwa-dong', 'latitude': 37.575817, 'longitude': 126.815719, 'label': 'Gangseo'},
    {'dong': 'Sangam-dong', 'latitude': 37.577039, 'longitude': 126.891620, 'label': 'Gangseo'},
    # 강남구
    {'dong': 'Nonhyun-dong', 'latitude': 37.508838, 'longitude': 127.030720, 'label': 'Gangnam'},
    {'dong': 'Daechi-dong', 'latitude': 37.501163, 'longitude': 127.057193, 'label': 'Gangnam'},
    {'dong': 'Seocho-dong', 'latitude': 37.486401, 'longitude': 127.018281, 'label': 'Gangnam'},
    {'dong': 'Bangbae-dong', 'latitude': 37.483279, 'longitude': 126.988194, 'label': 'Gangnam'},
    {'dong': 'Dogok-dong', 'latitude': 37.492896, 'longitude': 127.043159, 'label': 'Gangnam'},
    # 강북구
    {'dong': 'Pyeongchang-dong', 'latitude': 37.612129, 'longitude': 126.975724, 'label': 'Gangbuk'},
    {'dong': 'Sungbuk-dong', 'latitude': 37.597916, 'longitude': 126.998067, 'label': 'Gangbuk'},
    {'dong': 'Ssangmoon-dong', 'latitude': 37.648094, 'longitude': 127.030421, 'label': 'Gangbuk'},
    {'dong': 'Ui-dong', 'latitude': 37.648446, 'longitude': 127.011396, 'label': 'Gangbuk'},
    {'dong': 'Samcheong-dong', 'latitude': 37.591109, 'longitude': 126.980488, 'label': 'Gangbuk'},
    # 강동구
    {'dong': 'Hwayang-dong', 'latitude': 37.544234, 'longitude': 127.071648, 'label': 'Gangdong'},
    {'dong': 'Gui-dong', 'latitude': 37.543757, 'longitude': 127.086803, 'label': 'Gangdong'},
    {'dong': 'Neung-dong', 'latitude': 37.553102, 'longitude': 127.080248, 'label': 'Gangdong'},
    {'dong': 'Amsa-dong', 'latitude': 37.552370, 'longitude': 127.127124, 'label': 'Gangdong'},
    {'dong': 'Chunho-dong', 'latitude': 37.547436, 'longitude': 127.137382, 'label': 'Gangdong'}
]

test_df = pd.DataFrame(dong_dict_list) # 테이블 형으로 제공해주는 것이 가장 먼저다.
test_df = test_df[['dong', 'longitude', 'latitude', 'label']]
test_df

```

	dong	longitude	latitude	label
0	Gaebong-dong	126.854547	37.489853	Gangseo
1	Gochuk-dong	126.859245	37.501394	Gangseo
2	Hwagok-dong	126.847951	37.537759	Gangseo
3	Banghwa-dong	126.815719	37.575817	Gangseo
4	Sangam-dong	126.891620	37.577039	Gangseo
5	Nonhyun-dong	127.030720	37.508838	Gangnam
6	Daechi-dong	127.057193	37.501163	Gangnam
7	Seocho-dong	127.018281	37.486401	Gangnam
8	Bangbae-dong	126.988194	37.483279	Gangnam
9	Dogok-dong	127.043159	37.492896	Gangnam
10	Pyeongchang-dong	126.975724	37.612129	Gangbuk
11	Sungbuk-dong	126.998067	37.597916	Gangbuk
12	Ssangmoon-dong	127.030421	37.648094	Gangbuk
13	Ui-dong	127.011396	37.648446	Gangbuk
14	Samcheong-dong	126.980488	37.591109	Gangbuk
15	Hwayang-dong	127.071648	37.544234	Gangdong
16	Gui-dong	127.086803	37.543757	Gangdong
17	Neung-dong	127.080248	37.553102	Gangdong
18	Amsa-dong	127.127124	37.552370	Gangdong
19	Chunho-dong	127.137382	37.547436	Gangdong

정답 or Target

iloc의 i는 index이다.

```
8]: # 현재 가지고 있는 데이터에서, 레이블의 갯수를 확인
    train_df.label.value_counts()
```

```
8]: Gangseo      5
     Gangnam     5
     Gangbuk     5
     Gangdong    5
     Name: label, dtype: int64
```

```
9]: test_df.label.value_counts()
```

```
9]: Gangseo      5
     Gangnam     5
     Gangbuk     5
     Gangdong    5
     Name: label, dtype: int64
```

데이터 전처리_20221116

- 이번 예제에서는 위도와 경도 정보만으로, 그 지역의 레이블(타겟)을 예측해 볼 수 있도록 데이터 전처리를 함.
- 경도와 위도의 평균과 편차를 보도록 함.

```
]: train_df.describe() # R의 summary와 같다
```

```
]:
```

	longitude	latitude
count	20.000000	20.000000
mean	126.999772	37.547909
std	0.089387	0.055086
min	126.849500	37.451900
25%	126.913481	37.510177
50%	127.028844	37.552597
75%	127.056458	37.573690
max	127.151405	37.668800

- 경도와 위도는 같은 단위를 사용하는 것을 확인.
- 이번 예제는 의사결정트리로 데이터를 분류 적용.
- 의사결정트리는 각 특징을 독립적으로 사용하기 때문에, 이번 예제에서는 별다른 전처리 과정이 필요없다.

```
] : train_df.head()
```

```
] :
```

	district	longitude	latitude	label
0	Gangseo-gu	126.849500	37.551000	Gangseo
1	Yangcheon-gu	126.855396	37.524240	Gangseo
2	Guro-gu	126.887400	37.495400	Gangseo
3	Geumcheon-gu	126.902000	37.451900	Gangseo
4	Mapo-gu	126.908728	37.560229	Gangseo

```
] : test_df.head()
```

```
] :
```

	dong	longitude	latitude	label
0	Gaebong-dong	126.854547	37.489853	Gangseo
1	Gochuk-dong	126.859245	37.501394	Gangseo
2	Hwagok-dong	126.847951	37.537759	Gangseo
3	Banghwa-dong	126.815719	37.575817	Gangseo
4	Sangam-dong	126.891620	37.577039	Gangseo

데이터 시각화

- 데이터를 특징을 바탕으로 한 공간에 데이터 특징을 시각화함으로써, 우리는 머신러닝 학습에 필요한 특징과 불필요한 특징을 쉽게 구분지을 수 있고, 데이터의 패턴을 눈으로 쉽게 파악할 수 있음.

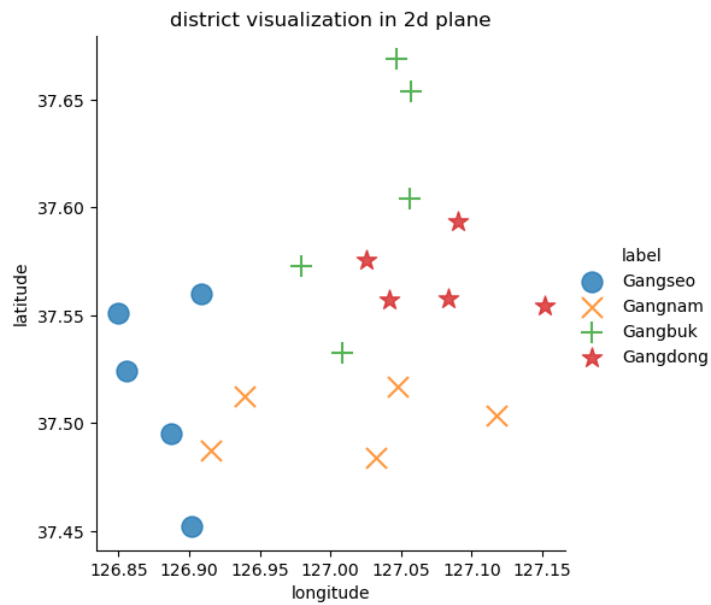
fit_reg=False

```
3]: import matplotlib.pyplot as plt
import seaborn as sns

# 경도, 위도에 따른 데이터 시각화
sns.lmplot(data=train_df, x='longitude', y='latitude', fit_reg=False, scatter_kws={'s':150}, markers=['o','x','+','*'], hue='label')

# title
plt.title('district visualization in 2d plane')

3]: Text(0.5, 1.0, 'district visualization in 2d plane')
```



fit_reg=True

```

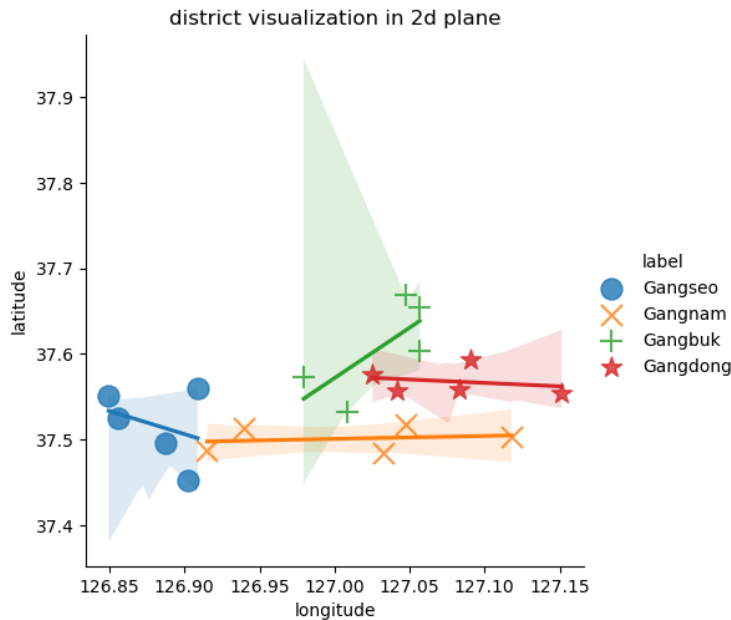
]: import matplotlib.pyplot as plt
import seaborn as sns

# 경도, 위도에 따른 데이터 시각화
sns.lmplot(data=train_df, x='longitude', y='latitude', fit_reg=True, scatter_kws={'s':150}, markers=['o', 'x', '+', '*'], hue='label')

# title
plt.title('district visualization in 2d plane')

]: Text(0.5, 1.0, 'district visualization in 2d plane')

```



공공데이터 포털

공공데이터 포털

국가에서 보유하고 있는 다양한 데이터를 『공공데이터의 제공 및 이용 활성화에 관한 법률 (제11956호)』에 따라 개방하여 국민들이 보다 쉽고 용이하게 공유·활용할 수 있도록 공공데이터(Dataset)와 Open API로 제공하는 사이트입니다.

<https://www.data.go.kr/tcs/puc/selectPublicUseCaseListView.do>



district 삭제됨 - 꼭 필요한 데이터만 가지고 결과를 내기 위해서

데이터 다듬기

- 학습 및 테스트에 필요 없는 특징(feature)을 데이터에서 제거.
- 구 이름 및 동 이름은 학습 및 테스트에 필요 없으므로 제거.

```
5]: train_df.drop(['district'], axis=1, inplace=True) # inplace=True 자체적으로 리턴되는 것을 업데이트해줌.  
train_df
```

5]:

	longitude	latitude	label
0	126.849500	37.551000	Gangseo
1	126.855396	37.524240	Gangseo
2	126.887400	37.495400	Gangseo
3	126.902000	37.451900	Gangseo
4	126.908728	37.560229	Gangseo
5	126.915065	37.487517	Gangnam
6	126.939300	37.512400	Gangnam
7	127.032400	37.483700	Gangnam
8	127.047300	37.517200	Gangnam
9	127.117898	37.503510	Gangnam
10	127.008605	37.532561	Gangbuk
11	126.979400	37.573000	Gangbuk
12	127.056344	37.603979	Gangbuk
13	127.056800	37.654200	Gangbuk
14	127.047100	37.668800	Gangbuk
15	127.041667	37.557340	Gangdong
16	127.025288	37.575759	Gangdong
17	127.083467	37.557562	Gangdong
18	127.151405	37.554194	Gangdong
19	127.090384	37.593684	Gangdong

```
7]: test_df.drop(['dong'], axis=1, inplace=True) # inplace=True 자체적으로 리턴되는 것을 업데이트해줌.  
test_df
```

```
7]:
```

	longitude	latitude	label
0	126.854547	37.489853	Gangseo
1	126.859245	37.501394	Gangseo
2	126.847951	37.537759	Gangseo
3	126.815719	37.575817	Gangseo
4	126.891620	37.577039	Gangseo
5	127.030720	37.508838	Gangnam
6	127.057193	37.501163	Gangnam
7	127.018281	37.486401	Gangnam
8	126.988194	37.483279	Gangnam
9	127.043159	37.492896	Gangnam
10	126.975724	37.612129	Gangbuk
11	126.998067	37.597916	Gangbuk
12	127.030421	37.648094	Gangbuk
13	127.011396	37.648446	Gangbuk
14	126.980488	37.591109	Gangbuk
15	127.071648	37.544234	Gangdong
16	127.086803	37.543757	Gangdong
17	127.080248	37.553102	Gangdong
18	127.127124	37.552370	Gangdong
19	127.137382	37.547436	Gangdong

```
]]: X_train = train_df[['longitude', 'latitude']]  
y_train = train_df[['label']]
```

```
]]: X_test = test_df[['longitude', 'latitude']]  
y_test = test_df[['label']]
```

```
]]: from sklearn import tree
```


label

미국·영국['leɪbl] ⇨ 영국식 ⇨

명사

- 1 (종이 등에 물건에 대한 정보를 적어 붙여 놓은) 표[라벨/상표] (=tag, ticket)

The washing instructions are on the **label**. ⇨

세탁법은 라벨에 나와 있다.

- 2 (사람물건의 성격 등을 묘사하는) 딱지[꼬리표]

I hated the **label** 'housewife'. ⇨

나는 '가정주부'라는 꼬리표가 싫었다.

동사

- 1 라벨[상표/표]을 붙이다, (표 같은 것에 필요한 정보를) 적다

We carefully **labelled** each item with the contents and the date. ⇨

우리는 각 품목에 내용물과 날짜를 조심해서 적어 넣었다[적은 표를 조심해서 붙였다].

- 2 (특히 부당하게) 딱지[꼬리표]를 붙이다

He was **labelled** (as) a traitor by his former colleagues. ⇨

그에게는 이전 동료들에 의해 배신자라는 딱지가 붙었다.

영어사적 다른 뜻 3

```
: from sklearn import tree # 의사결정트리
from sklearn import preprocessing
```

```
: # LabelEncoder로 레이블을 숫자로 변경
le = preprocessing.LabelEncoder()
y_encoded = le.fit_transform(y_train) # 경합인 y_train를 fit_transform()을 통해 자동으로 '텍스트'를 '수치형'으로 변환해주는 기능
print(y_encoded)

clf = tree.DecisionTreeClassifier(random_state=35).fit(X_train, y_encoded) # .fit(X_train, y_encoded) 한줄의 코드로 수행되게끔 함.
```

```
"""
# LabelEncoder로 레이블을 숫자로 변경
le = preprocessing.LabelEncoder()
y_encoded = le.fit_transform(y_train)
print(y_encoded)

clf = tree.DecisionTreeClassifier(random_state=35).fit(X_train, y_encoded)
"""
```

```
[3 3 3 3 3 2 2 2 2 0 0 0 0 1 1 1 1]
```

```
C:\ProgramData\Anaconda3\envs\wt_f_cpu\lib\site-packages\sklearn\preprocessing\label.py:115: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
: '''\n# LabelEncoder로 레이블을 숫자로 변경\nle = preprocessing.LabelEncoder()\ny_encoded = le.fit_transform(y_train)\nprint(y_encoded)\n\nclf = tree.DecisionTreeClassifier(random_state=35).fit(X_train, y_encoded)\n'''
```

```

: x = np.array([1,2]) # np.array 바로 배열형으로 전환시켜줄,
  y = np.array([3,4])

print(np.r_[x,y]) # 1차원 # row로
print(np.r_[[x],[y]]) # 2차원으로 바꿔줄,

[1 2 3 4]
[[1 2]
 [3 4]]

: print(np.c_[x,y]) # column으로
  print(np.c_[[x],[y]])

[[1 3]
 [2 4]]
[[1 2 3 4]]

```

```

]: # pyplot은 숫자로 표현된 레이블을 시각화 할 수 있음
def display_decision_surface(clf, X, y): # 시각화를 할 때 입력으로 사용된 위도, 경도를 가지고 작업,
    x_min = X.longitude.min() - 0.01 # 최소에서 빼주고,
    x_max = X.longitude.max() + 0.01 # 최대에서 더해줄
    y_min = X.latitude.min() - 0.01
    y_max = X.latitude.max() + 0.01

    # 파라미터 설정
    n_classes = len(le.classes_) # 0 1 2 3 저장
    plot_colors = "rywb" # red yellow white black
    plot_step = 0.001

    xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step), np.arange(y_min, y_max, plot_step))

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()]) # predict() 최적의 모델을 가지고 예측하는 것. # c_ - column의 약자
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

    # 학습 데이터를 차트에 표시
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(X.loc[idx].longitude, X.loc[idx].latitude, c = color, label = le.classes_[i],
                    cmap=plt.cm.RdYlBu, edgecolor='black', s=200)

    # 차트 제목
    plt.title("Decision surface of a decision tree", fontsize=16)
    # 차트 기호 설명
    plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0, fontsize=14)
    # x축의 이름과 폰트 크기 설정
    plt.xlabel('longitude', fontsize=16)
    # y축의 이름과 폰트 크기 설정
    plt.ylabel('latitude', fontsize=16)
    # 차트 크기 설정
    plt.rcParams['figure.figsize'] = [7,5]
    # x축 좌표상의 폰트 크기 설정
    plt.rcParams['xtick.labelsize'] = 14
    # y축 좌표상의 폰트 크기 설정
    plt.rcParams['ytick.labelsize'] = 14

    # 차트 그리기
    plt.show()

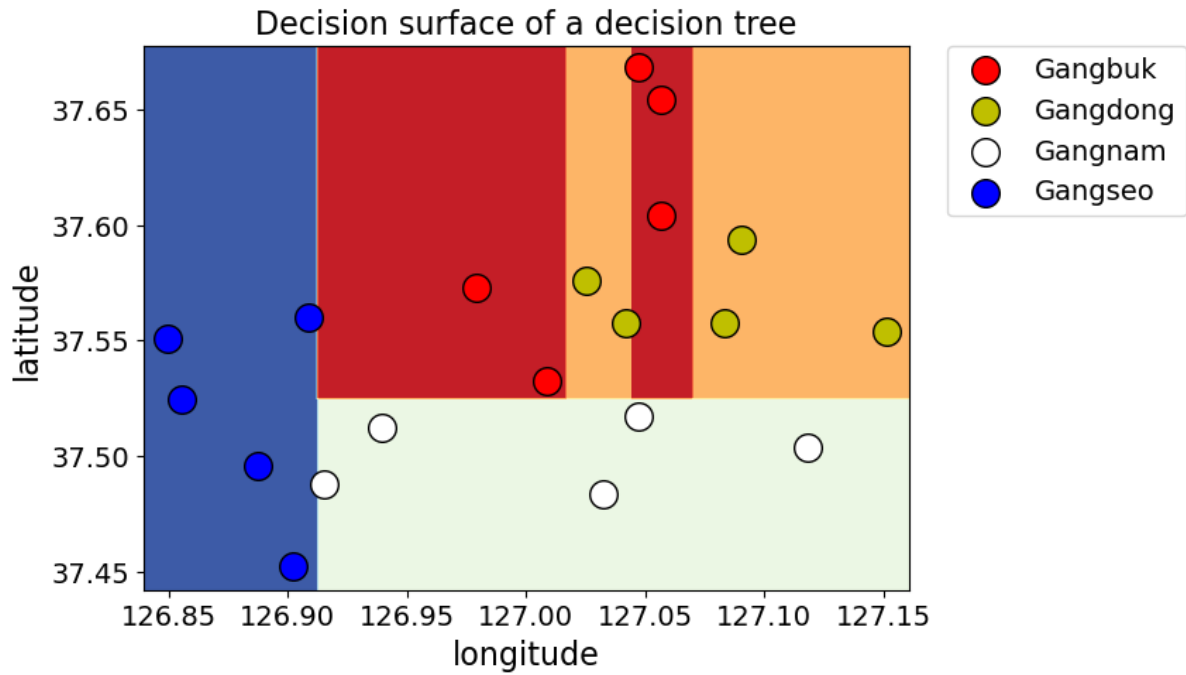
```

파라미터 없이 학습한 모델의 결정 표면 시각화

- 의사결정트리는 오버피팅(과대적합)되기 상당히 쉬운 모델임
- 별도의 파라미터를 설정해주지 않을 경우, 최대한 학습데이터에만 잘 맞게 모델이 형성.
- 아래 차트를 통해, 학습된 모델이 강남, 강북, 강동, 강서 지역 구분이라기 보다는, 단순히 학습 데이터 구분에만 집중된 모델임을 볼 수 있다.

```
] : display_decision_surface(clf, X_train, y_encoded)
```

C:\ProgramData\Anaconda3\envs\utf_cpu\lib\site-packages\sklearn\base.py:451: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
"X does not have valid feature names, but"



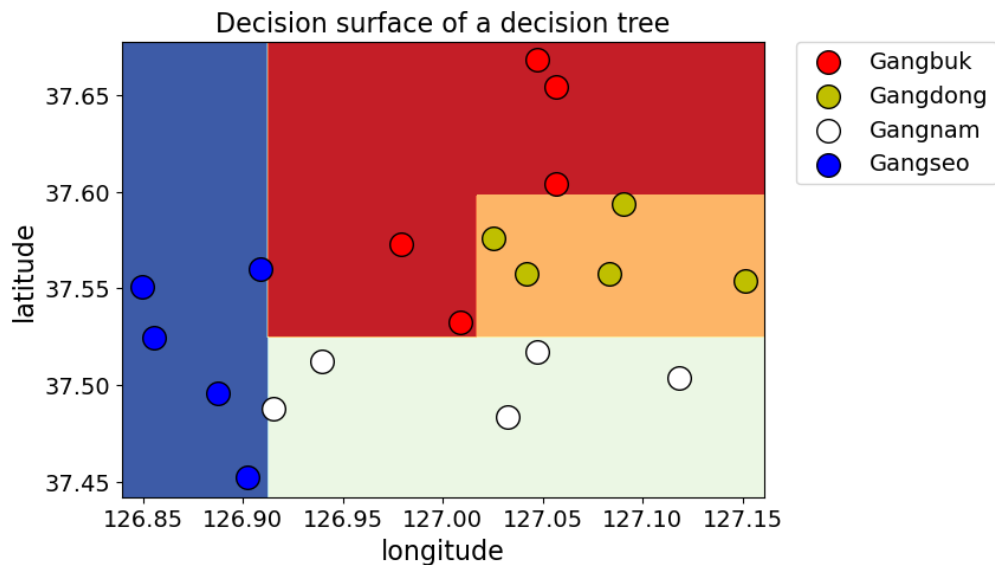
파라미터 설정한 모델의 결정 표면 시각화

- 오버피팅을 피하기 위해 별도의 파라미터를 설정한 의사결정트리의 의사결정표면 차트 생성.
- `max_depth`: 트리의 최대 한도 깊이
- `min_samples_split`: 자식 노드를 갖기 위한 최소한의 데이터 갯수
- `min_samples_leaf`: 맨 마지막 끝 노드의 최소 데이터 갯수
- `random_state`: 여러번 실행해도, 파라미터가 같은 경우, 결과가 항상 같게 만들어주는 파라미터

```
5) clf = tree.DecisionTreeClassifier(random_state=70,
                                   max_depth=4,
                                   min_samples_split=2,
                                   min_samples_leaf=2).fit(X_train, y_encoded)

display_decision_surface(clf, X_train, y_encoded)
```

C:\ProgramData\Anaconda3\envs\ft_cpu\lib\site-packages\sklearn\base.py:451: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
"X does not have valid feature names, but"



[실습]의사결정 트리(Decision Tree)

의사결정 트리를 사용하여 데이터 분류를 수행할 것이다. 데이터는 서울의 시군구, 읍면동을 사용하였고 강서, 강동, 강북, 강남으로 분류하는 작업을 실습해보았다. 데이터 불러오기 서울의 시군구 데이터는 시군구 데이터, 읍면동 데이터는 읍면동 데이터에서 가져올 수 있다.

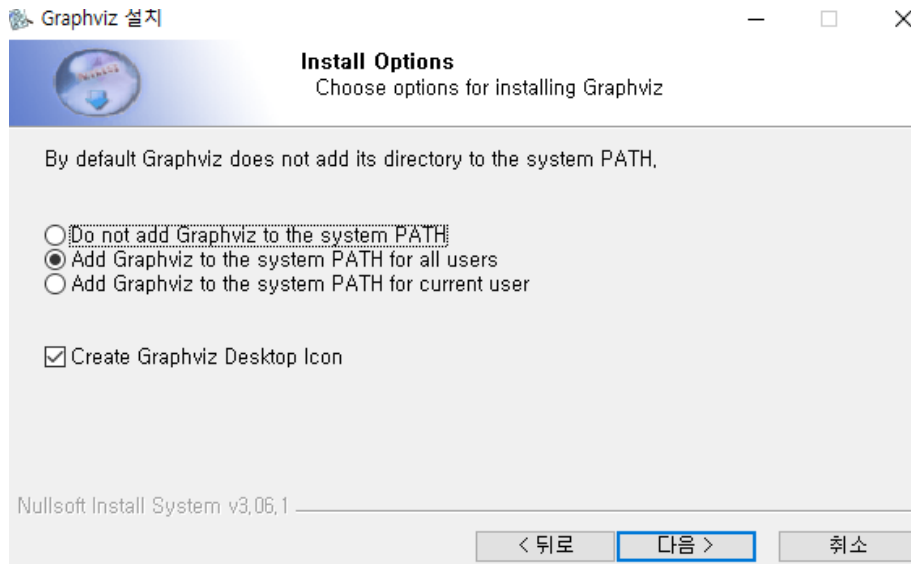
<https://kjoy042386.tistory.com/108>

	district	latitude	longitude	label
0	Dobong-gu	37.665861	127.031767	Gangbuk
1	Eunpyeong-gu	37.617612	126.922700	Gangseo
2	Dongdaemun-gu	37.583801	127.050700	Gangdong
3	Dongjak-gu	37.496504	126.944307	Gangnam
4	Geumcheon-gu	37.460097	126.900155	Gangseo

Download

Source code packages for the latest stable and development versions of Graphviz are available, along with instructions for anonymous access to the sources using Git. Packages marked with an asterisk(*) are provided by outside parties. We list them for convenience, but disclaim responsibility for the contents of these packages.

<https://graphviz.org/download/>



all users로 다운로드

Download

Source code packages for the latest stable and development versions of Graphviz are available, along with instructions for anonymous access to the sources using Git. Packages marked with an asterisk(*) are provided by outside parties. We list them for convenience, but disclaim responsibility for the contents of these packages.

<https://graphviz.org/download/>

의사결정트리 시각화

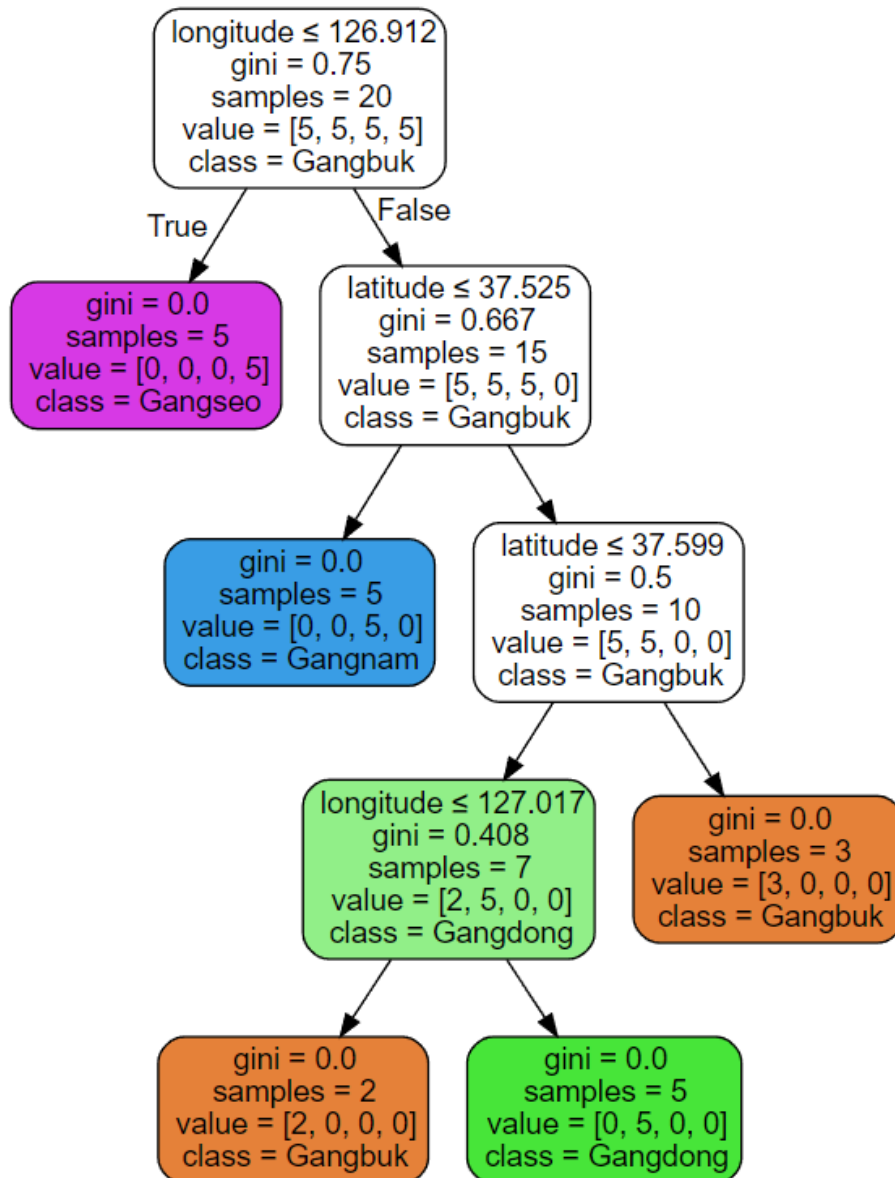
- 의사결정트리의 장점은 예측값이 어떤 식으로 판단되었는지 이해가 쉽다.
- 다른 모델들에 비해, 큰 수학적인 지식이 없어도, 시각화 그림을 보면, 어떻게 예측값이 도출되었는지 쉽게 알 수 있음.

```
: import graphviz

: dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("seoul")

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=['longitude', 'latitude'],
                                class_names=['Gangbuk', 'Gangdong', 'Gangnam', 'Gangseo'],
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

[51]:



gini : 불순도 척도. 0일 경우, 모든 샘플이 하나의 분류값을 갖고 있으며, 1에 가까울수록 여러 분류값이 한 노드에 존재.
samples : 노드 안에 들어있는 데이터의 갯수
value : 분류값 별 데이터의 갯수
class : 분류값

테스트

- 모델을 테스트.

```
from sklearn.metrics import accuracy_score
```

```
pred = clf.predict(X_test)
```

```
print('accuracy : ' + str(accuracy_score(y_test.values.ravel(), le.classes_[pred]))) # 인코딩된 값으로 넣어줄.
```


```
accuracy : 1.0
```

```
: comparison = pd.DataFrame({'prediction': le.classes_[pred],  
                             'ground_truth': y_test.values.ravel()})  
comparison
```

```
:  
      prediction ground_truth  
0    Gangseo    Gangseo  
1    Gangseo    Gangseo  
2    Gangseo    Gangseo  
3    Gangseo    Gangseo  
4    Gangseo    Gangseo  
5    Gangnam    Gangnam  
6    Gangnam    Gangnam  
7    Gangnam    Gangnam  
8    Gangnam    Gangnam  
9    Gangnam    Gangnam  
10   Gangbuk    Gangbuk  
11   Gangbuk    Gangbuk  
12   Gangbuk    Gangbuk  
13   Gangbuk    Gangbuk  
14   Gangbuk    Gangbuk  
15   Gangdong   Gangdong  
16   Gangdong   Gangdong  
17   Gangdong   Gangdong  
18   Gangdong   Gangdong  
19   Gangdong   Gangdong
```

seoul 파일

 seoul

 seoul.pdf

