숭실대학교

인공지능특론 1
보고서

Assignment #2 – Differential Privacy

1102040007 오혁진
1102340003 유자연
1102378004 서영재
1102378010 강민정

• 목차

Ⅰ 서론

■ Differential Privacy

: 데이터 분석을 하면서 개인정보 보호를 보장할 수 있는 기술로, 데이터 분석 결과가 개인정보를 보호하고, 그 결과를 이용하여 유추할 수 없도록 한다.

Differential Privacy의 핵심은 데이터 분석의 결과가 특정 개인의 데이터에 대한 정보를 제공하지 않는 것으로, 즉 분석 결과를 이용하여 특정 개인의 데이터를 유추할 수 없도록 하는 것을 목표로 한다. 이를 위해 Differential Privacy는 개인정보 왜곡과 노이즈 추가를 통해 개인정보를 보호한다.

개인정보 왜곡은 데이터에서 무작위로 일부 값을 변경하여 데이터 분석 결과가 일정 수준 이상으로 보호되도록 하며, 노이즈 추가는 원본 데이터에 무작위로 일부 값을 추가하여 개인정보 보호를 제공한다.

■ Meaning of ϵ and δ

$$Pr[M(d) \in S] \leq e^{\epsilon} Pr[M(d') \in S] + \delta$$

- Epsilon(ϵ)
  : 개인정보 왜곡의 강도를 나타내는 매개변수로, epsilon 값이 작을수록 왜곡 정도가 작아지며, epsilon 값이 커질수록 왜곡 정도가 커진다. 즉, epsilon 값이 작아질수록 보호 수준이 높아진다.

- Delta(δ)
  : 개인정보 유출의 확률을 나타내는 매개변수로, delta 값이 작을수록 개인정보 유출 확률이 작아지며, delta 값이 커질수록 개인정보 유출 확률이 커진다. 즉, delta 값이 작을수록 보호 수준이 높아지지만, 데이터 분석 결과의 정확도가 낮아진다.

- (ϵ, δ)-Differential Privacy는 개인정보 손실의 절대값이 최소 $1-\delta$의 확률로 ϵ에 의해 제한되도록 한다. 즉 epsilon과 delta 값은 서로 상충되는 요소이므로, 보호 수준과 확률을 동시에 고려하여 적절한 값을 선택해야 한다.

■ Gaussian Mechanism

: 노이즈가 가우시안 분포를 따르는 임의의 값을 데이터에 추가하는 방식으로, 일반적으로 쿼리(데이터베이스에서 정보를 추출하는 작업을 말함)의 민감도가 낮은 경우에 적용된다. 즉, 쿼리의 결과가 데이터 포인트의 작은 변화에 크게 영향을 받지 않을 때 적용된다. 가우시안 분포는 노이즈의 크기를 조절하는 매개변수인 epsilon 값을 사용하여 노이즈의 강도를 조절할 수 있다.

$$\sigma \geq \sqrt{2\log\left(\frac{5/4}{\delta}\right)} \frac{\Delta f}{\epsilon}$$

■ Laplace Mechanism

: 노이즈가 라플라스 분포를 따르는 임의의 값을 데이터에 추가하는 방식으로, 일반적으로 쿼리의 민감도가 높은 경우에 적용된다. Gaussian Mechanism과 마찬가지로 epsilon 값을 이용하여 노이즈의 강도를 조절할 수 있다.

$$b = \frac{\Delta f}{\epsilon}$$

(*b = scale of Laplace noise **f = sensitivity)

두 메커니즘 모두 Differential Privacy를 보장하기 위해 사용되며, 적절한 메커니즘은 쿼리의 민감도와 데이터의 특성에 따라 결정된다.

■ Sensitivity

: Differential Privacy에서 sensitivity는 데이터베이스에서 한 개인의 데이터를 삭제하거나 수정했을 때 쿼리 결과에 미치는 영향의 최대 크기를 나타내는 개념이다. 즉, input이 output에 얼마나 큰 영향을 미치는가에 관한 좌표이며, input과 output의 차이가 크면 노이즈를 강하게 추가하는 것이 민감한 정보의 노출을 최소화 시킬 수 있다.

$$S = \Delta f = \max_{d_1, d_2} \| f(d_1) - f(d_2) \|_1$$

Ⅱ Differential Privacy

ⅰ. 데이터 선정

■ 데이터

: 종관기상관측(23년 2월 1일 24시간 분당 데이터) 중 7개 지역 임의 선택

■ 데이터 상세

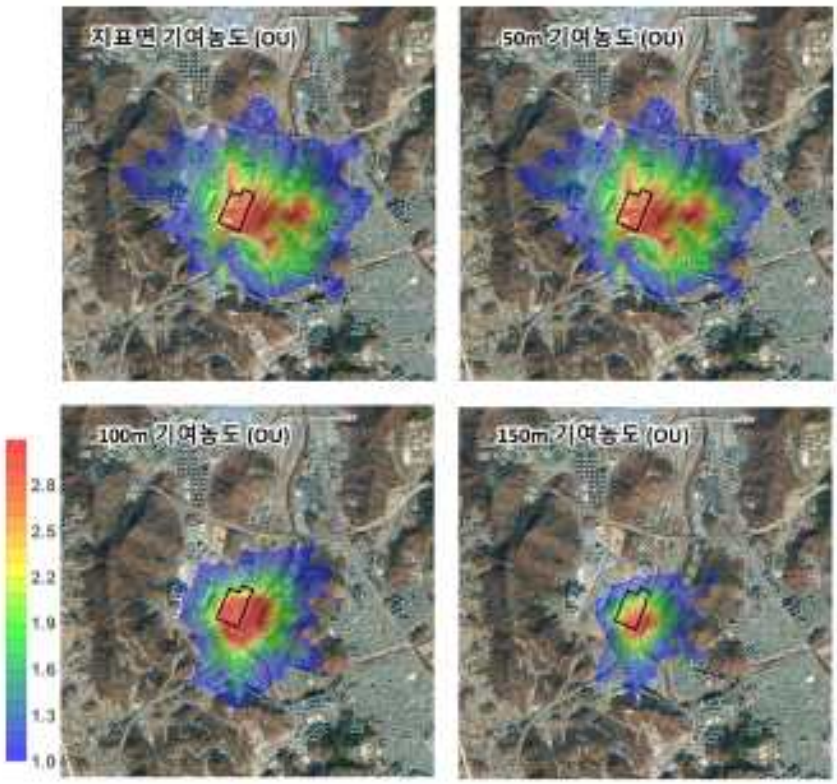
: 측정 지점, 일시, 기온(℃), 풍향(deg), 풍속(m/s), 현지기압(hPa), 습도(%) 등의 데이터를 customizing하여 사용

■ 데이터 수

: 약 1만 개

■ 데이터 사용 예시

: 시뮬레이션 프로그램을 이용한 특정 지역의 시간에 따른 대기오염물질 확산 추이 예상



[출처] 2023 악취 및 대기확산 모델링(CALPUFF) 교육, 안양대학교 기후·에너지·환경융합연구소, 한국냄새환경학회

ii. 코드 작성 및 결과

■ Assignment #1의 t-closeness

```
[ ]  def get_distribution(df, sensitive_column):
         return df[sensitive_column].value_counts(normalize=True)

     def apply_t_closeness(df, sensitive_column, columns, t):
         overall_dist = get_distribution(df, sensitive_column)
         grouped_data = df.groupby(columns)
         result_df = pd.DataFrame()

         for _, group in grouped_data:
             group_dist = get_distribution(group, sensitive_column)
             emd = wasserstein_distance(overall_dist, group_dist)

             if emd <= t:
                 result_df = result_df.append(group)

         if not result_df.empty:
             return result_df.reset_index(drop=True)
         else:
             print("No groups satisfy t-closeness.")
             return None

     # Load the CSV file
     file_path = "./original_data(2).csv"
     data = pd.read_csv(file_path)

     # Define the sensitive column and the columns to group by (e.g., anonymized columns)
     sensitive_column = '지점'
     columns_to_group_by = ['풍향(deg)','풍속(m/s)']

     # Apply k-anonymity and l-diversity (or any other anonymization techniques)
     # ...

     # Apply t-closeness
     t = 0.2
     tclosed_data = apply_t_closeness(data, sensitive_column, columns_to_group_by, t)

     if tclosed_data is not None:
         # Print the DataFrame
         print("Data after applying t-closeness:")
         print(tclosed_data)

         # Save the DataFrame to a CSV file
         output_file_path = "./t_closeness_data_output2.csv"
         tclosed_data.to_csv(output_file_path, index=False)
```

■ 데이터 세트에 Differential Privacy 적용

: 데이터 세트의 각 속성에 대한 민감도 및 δ 값 계산/결정

Laplace 또는 Gaussian 노이즈 적용

• 민감도 계산

: 각 속성의 민감도는 해당 속성이 가질 수 있는 최댓값과 최솟값의 차이

```
import pandas as pd

# 데이터 불러오기
data = pd.read_csv('./t_closeness_data_output2.csv')

# 각 속성의 민감도 계산하기
sensitivity = []
for column in data.columns:
    max_value = data[column].max()
    min_value = data[column].min()
    sensitivity.append(max_value - min_value)

print(sensitivity)
```

```
[94, 360, 7, 23, 17, 81, 13]
```

• Delta 값 설정하기

- δ 값은 프라이버시 보장을 위한 매개변수
- δ 값이 작을수록 더 강한 보안이 적용
- δ 값은 보통 0.1, 0.01 또는 0.001과 같이 작은 값으로 설정

```
delta = 0.1
```

• Laplace 또는 Gaussian 노이즈 적용하기

- x : 민감한 데이터 값이 들어있는 numpy 배열
- Sensitivity : 민감도 값
- Epsilon : 프라이버시 보장을 위한 매개변수로, 값이 작을수록 더 강한 보안이 적용됨

```
[ ]  import numpy as np

     def add_laplace_noise(x, sensitivity, epsilon):
         beta = sensitivity / epsilon
         noise = np.random.laplace(0, beta, len(x))
         return x + noise

     def add_gaussian_noise(x, sensitivity, epsilon):
         sigma = sensitivity / epsilon
         noise = np.random.normal(0, sigma, len(x))
         return x + noise
```

```
[ ]  # 각 속성의 민감도 계산하기
     sensitivity = {}
     for column in data.columns:
         max_value = data[column].max()
         min_value = data[column].min()
         sensitivity[column] = max_value - min_value

     # Laplace 노이즈 적용
     noisy_data = data.apply(lambda x: add_laplace_noise(x, sensitivity[x.name], np.log(1.25/delta)), axis=0)
```

```
[ ]  data
```

	지점	풍향(deg)	풍속(m/s)	현지기압(hPa)	해면기압(hPa)	습도(%)	일사(MJ/m^2)
0	133	0	1	1007	1016	71	0
1	133	0	1	1016	1025	53	13
2	133	0	1	1017	1026	54	13
3	90	0	1	1013	1015	42	0
4	90	0	1	1013	1015	43	0
...	...	...	...	...	...	...	...
6910	106	360	2	1008	1013	20	0
6911	184	360	5	1021	1024	66	11
6912	184	360	5	1022	1025	63	11
6913	102	360	5	1024	1028	49	13
6914	106	360	5	1011	1016	20	0

6915 rows × 7 columns

```
[ ]  noisy_data
```

	지점	풍향(deg)	풍속(m/s)	현지기압(hPa)	해면기압(hPa)	습도(%)	일사(MJ/m^2)
0	158.082068	-378.104176	3.136110	1012.425212	1035.575133	80.039963	-1.418224
1	142.924561	22.580400	0.859183	1008.822847	1023.623847	74.720878	11.495471
2	64.768146	142.624190	-15.896671	1004.958909	1041.306258	17.615720	18.207857
3	123.437357	-112.871679	-0.349390	1002.937375	1017.356296	56.497874	-3.417613
4	129.931195	226.343668	1.397121	997.786277	1008.845710	22.309273	-6.967543
...	...	...	...	...	...	...	...
6910	-72.333633	311.814821	0.332820	988.302935	1017.383836	121.077417	9.353380
6911	206.638804	556.027513	-16.630744	1025.682820	1032.891322	55.917291	9.173469
6912	202.171478	328.610600	-9.721177	1018.282203	1042.468229	55.299192	16.605189
6913	107.113152	429.491090	5.582923	1023.010091	1040.390225	46.011816	14.901456
6914	98.271669	86.969830	7.528082	1017.971274	1015.761886	52.229086	8.325689

6915 rows × 7 columns

Ⅲ non-DP 및 DP Plot

■ 다른 매개변수 값 사용(예: ϵ, 클리핑 바운드 등)

: 속성 분포 plot을 그리기 위해 데이터의 분포를 알아야 하므로, 기초 통계량을 먼저 계산

```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     # 데이터 불러오기
     data = pd.read_csv('./t_closeness_data_output2.csv')

     # 각 속성의 기초 통계량 계산하기
     statistics = data.describe()
     print(statistics)
```

```
                지점       풍향(deg)      풍속(m/s)    현지기압(hPa)    해면기압(hPa)  ₩
count  6915.000000  6915.000000  6915.00000  6915.000000  6915.000000
mean    129.523644   270.232827     2.96081  1011.477946  1017.141432
std      32.079736    88.053818     1.36860     4.915505     3.768114
min      90.000000     0.000000     0.00000  1001.000000  1011.000000
25%     102.000000   270.000000     2.00000  1008.000000  1014.000000
50%     133.000000   300.000000     3.00000  1011.000000  1016.000000
75%     152.000000   320.000000     4.00000  1015.000000  1020.000000
max     184.000000   360.000000     7.00000  1024.000000  1028.000000

            습도(%)   일사(MJ/m^2)
count  6915.000000  6915.000000
mean     48.069993     3.710774
std      21.821534     5.278662
min      18.000000     0.000000
25%      29.000000     0.000000
50%      43.000000     0.000000
75%      66.000000    10.000000
max      99.000000    13.000000
```

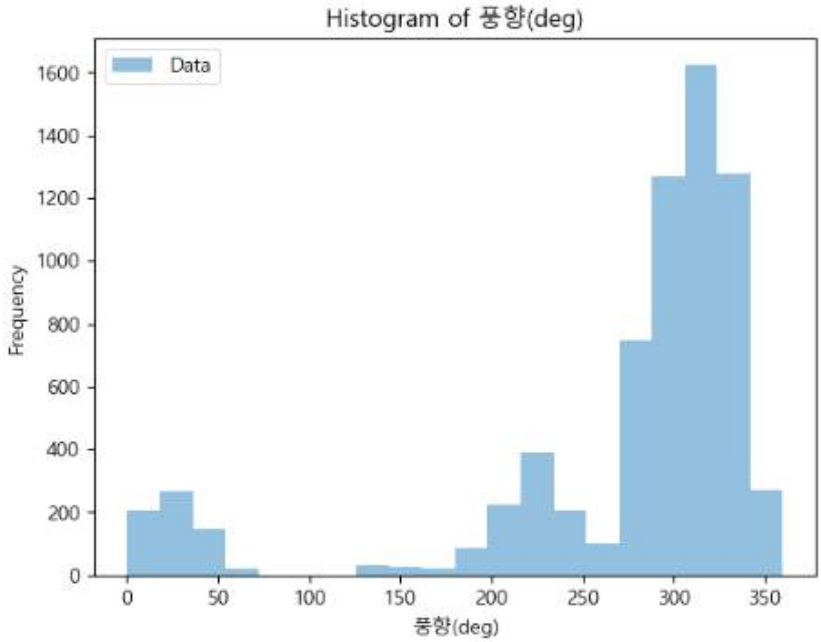
```
[ ]  def plot_histogram(data, column, bins=10):
         fig, ax = plt.subplots()
         ax.hist(data[column], bins=bins, alpha=0.5, label='Data')
         ax.set_xlabel(column)
         ax.set_ylabel('Frequency')
         ax.set_title('Histogram of {}'.format(column))
         plt.legend()
         plt.show()
```

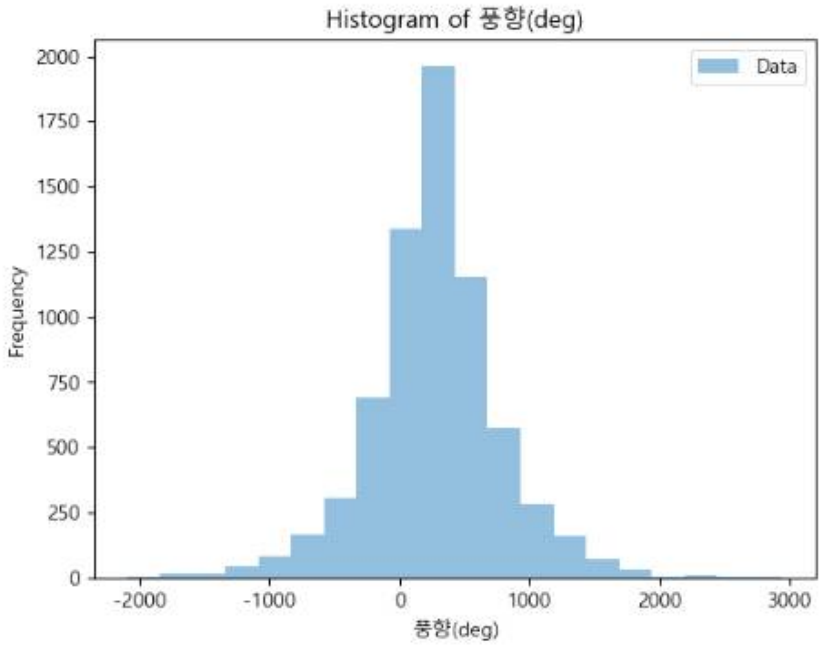
```
[ ]  ## 한글 깨짐 해결하기 위해 사용
     import matplotlib

     matplotlib.rcParams['font.family'] ='Malgun Gothic'
     matplotlib.rcParams['axes.unicode_minus'] =False
```

```
[ ]  # non-DP case
     plot_histogram(data, '풍향(deg)', bins=20)

     # DP case
     epsilon = 1.0
     noisy_data = data.apply(lambda x: add_laplace_noise(x, sensitivity[x.name], epsilon), axis=0)
     plot_histogram(noisy_data, '풍향(deg)', bins=20)
```





IV 결론

■ 각각의 파라미터가 의미하는 바를 알아보았다.

- Differential Privacy는 데이터 분석의 결과가 개인 데이터 대한 정보를 제공하지 않는 것을 핵심으로 한다
- Epsilon은 왜곡의 강도를 나타내며, Delta 값은 유출의 정도를 의미한다.
- Sensitivity는 개인의 데이터를 수정하였을 때 결과에 미치는 영향의 크기를 나타낸다.

■ 데이터 선정

- 과제 1과 같은 데이터를 이용하여 Differential Privacy에 적합하게 customizing을 하였다.
- 그 중 t-closeness 데이터를 이용하였다.

■ Differential Privacy

- 개인정보 왜곡과 노이즈 추가를 통해 개인정보를 보호하는 Differential Privacy를 데이터세트에 적용하였다.
- 그 중에서도 Laplace 노이즈를 적용하였다. non-DP와 DP Plot을 통해 확인한 결과가 다음과 같았다.

지점	풍향(deg)	풍속(m/s)	현지기압(hPa)	해면기압(hPa)	습도(%)	일사(MJ/m^2)
133	0	1	1007	1016	71	0
133	0	1	1016	1025	53	13
133	0	1	1017	1026	54	13
90	0	1	1013	1015	42	0
90	0	1	1013	1015	43	0
90	0	1	1013	1015	42	0
102	0	1	1010	1015	96	0
106	0	1	1007	1012	50	0
106	0	1	1007	1012	52	0

⇩

지점	풍향(deg)	풍속(m/s)	현지기압(hPa)	해면기압(hPa)	습도(%)	일사(MJ/m^2)
175.8112528	-722.2276277	0.87543681	939.4165094	1009.528457	68.74784703	1.110125846
42.56351894	-315.6540487	-8.294148151	1032.34894	998.2111951	-1.073336371	107.2659937
67.50773925	472.5720375	3.686919565	998.6534535	1001.725745	41.90278259	-15.67112793
0.649485827	241.9843706	16.56047103	1052.395476	1029.530099	101.4109688	2.534493792
168.1672332	436.6623395	6.602942543	992.6019588	1012.413744	-31.37765451	9.686733906
40.39840365	384.2503676	-2.416665832	995.5604405	1006.300346	12.2669464	1.557680434
48.54823818	89.23729687	-2.942966362	1002.971525	991.8959394	82.85861108	-0.323683833
44.87235448	-0.784280814	9.506852533	1006.935328	987.3050343	118.8172968	-11.13309271
156.0377474	-60.09949562	-3.720643619	1012.708043	999.9508618	38.88853359	-1.709406593