AI/ML

수도미터 동파경보 AI모델 고도화

2023. 7. 19

K-water연구원 AI연구센터

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 추진경위

□ ’22. 4 : 국가상수도정보시스템 고도화 방침 수립(유역수도지원처)

* AI기반 동파예측을 위한 연구관리처 협업 추진 계획

□ ‘22. 9 : (연구관리처) 온도 예측 AI 모델 개발 및 검증 완료

□ ‘22. 11 : (유역수도지원처)국가상수도정보시스템(https://waternow.go.kr)

- [동파 위험 정보] 서비스 개시

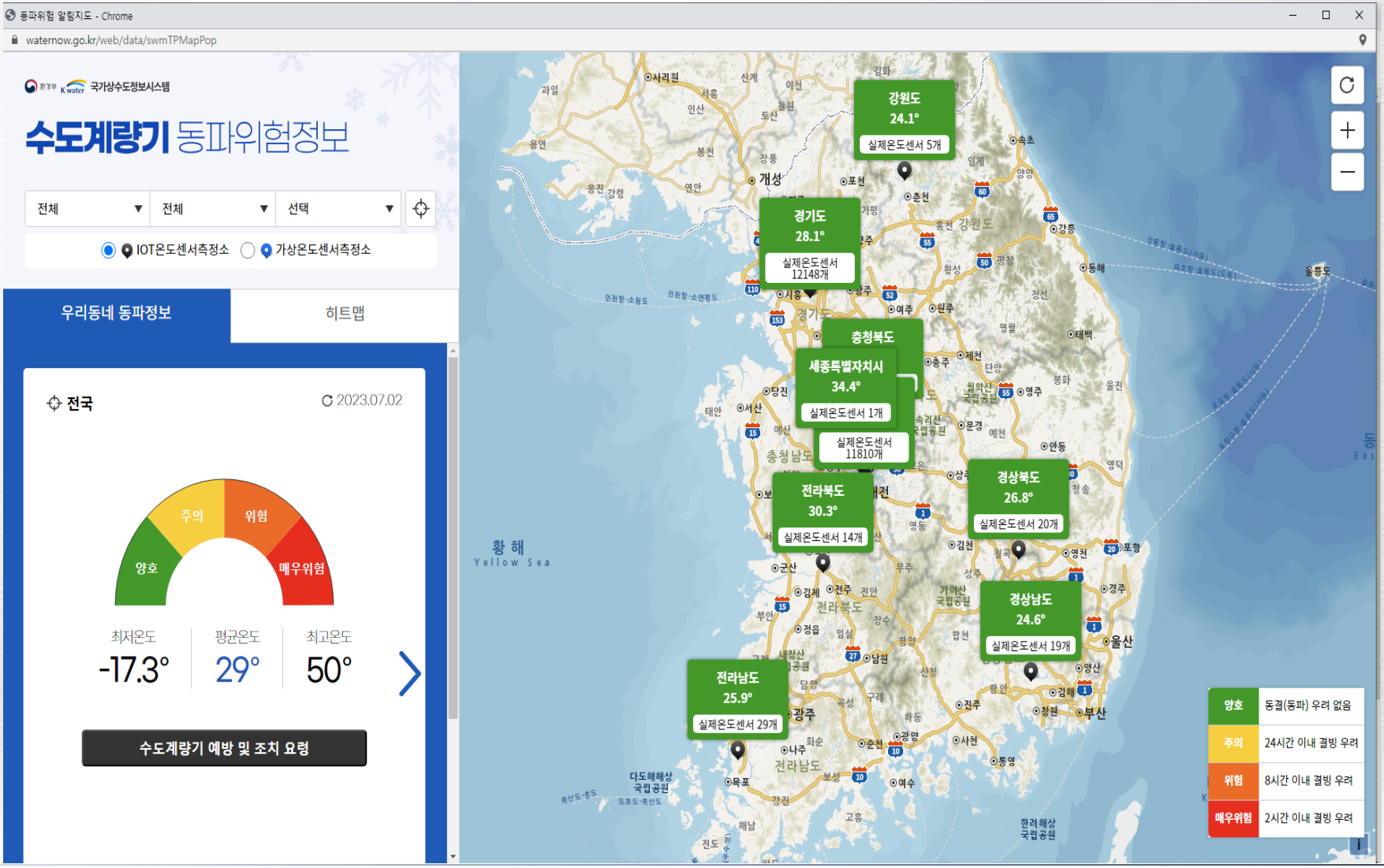
□ ‘23. 5 : (연구관리처) 온도예측 시각화 데모용 모듈(Streamlit) 고도화

□ ‘23. 6 : 온도예측 AI 모델 고도화 협의(유역수도지원처)

* AI 모델 재 학습 및 수온 등 추가인자를 통한 예측 검토

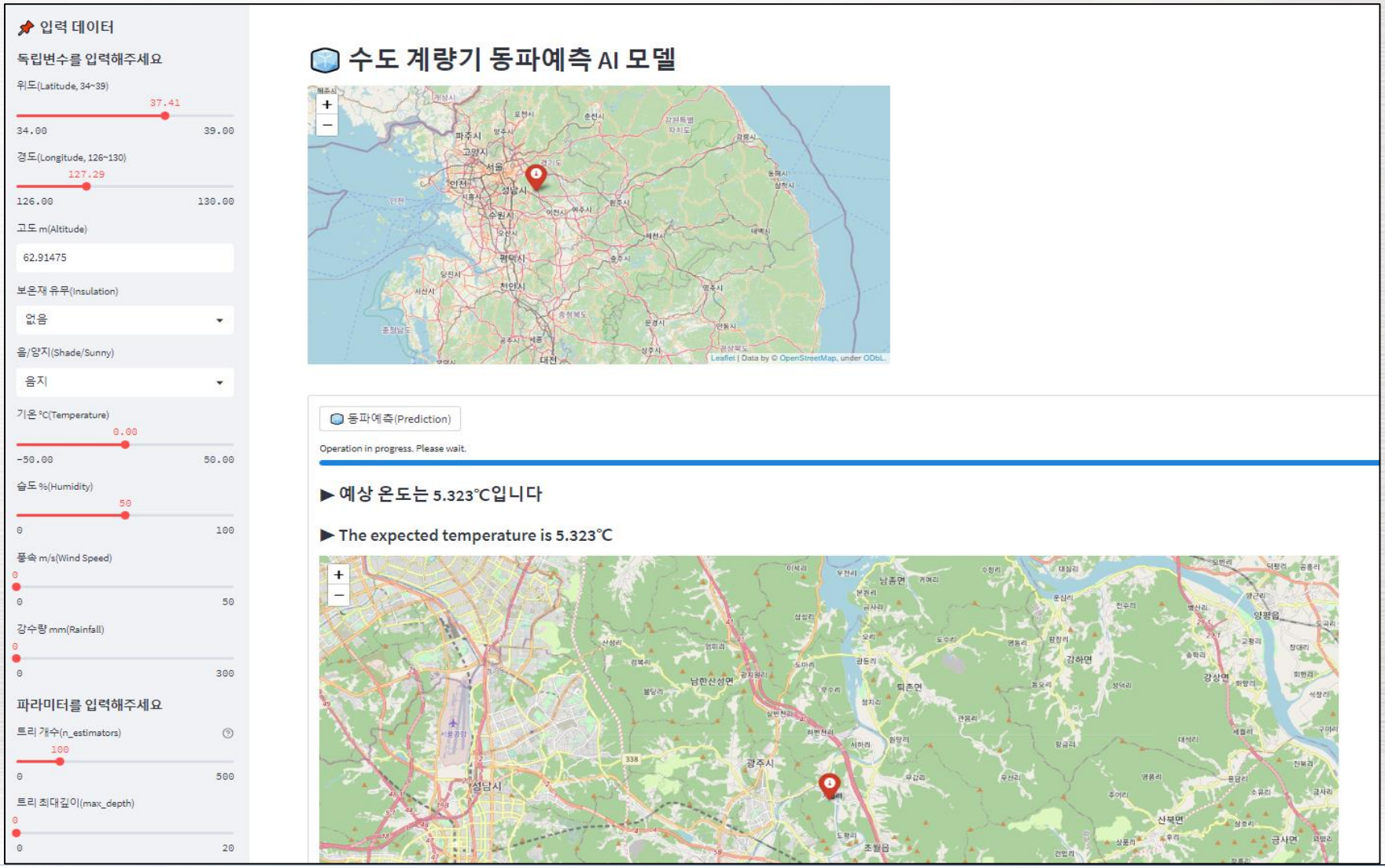
(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 국가상수도정보시스템(https://waternow.go.kr) - [동파 위험 정보] 서비스



(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 수도미터 동파예측 시각화 데모용 모듈(Streamlit) 고도화



(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 개발현황

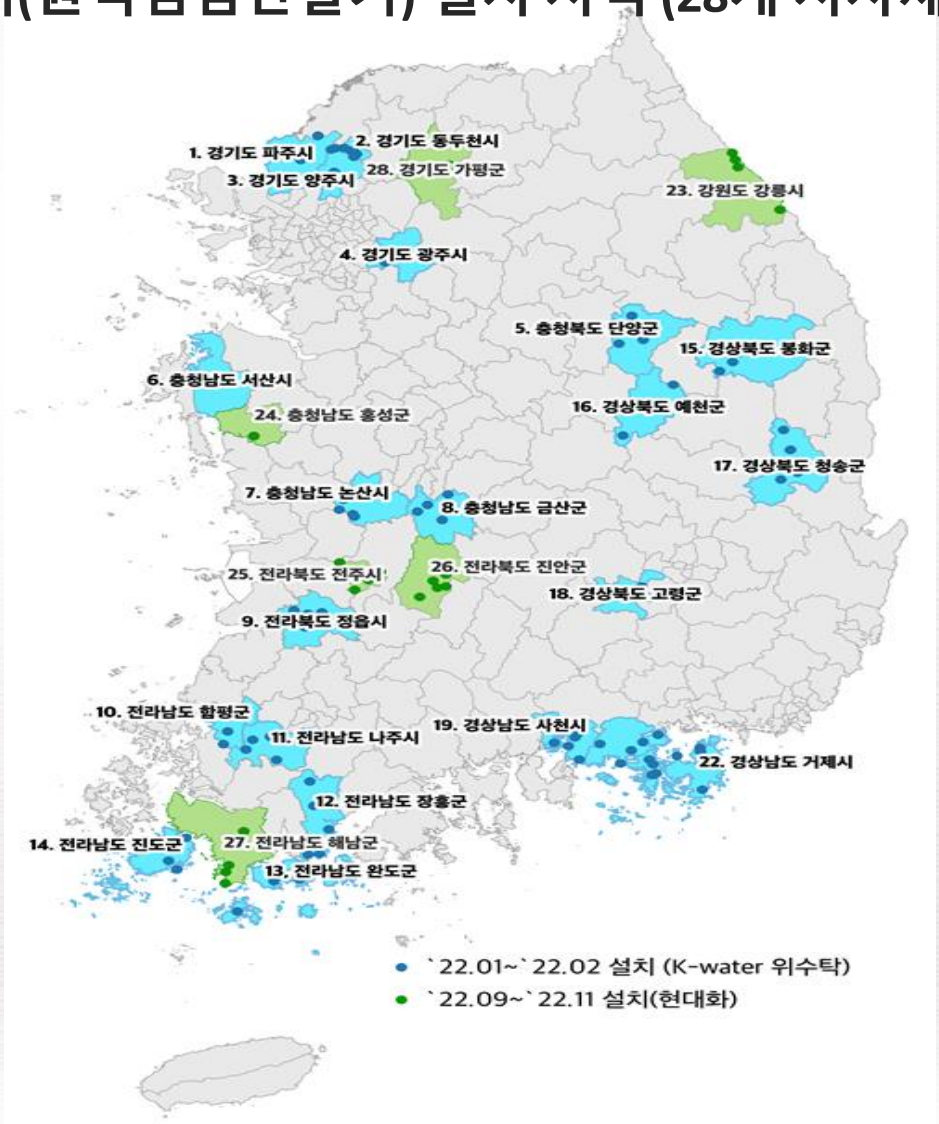
□ AI 모델 및 시각화 데모용 모듈 개발('22)

- IoT온도센서 **미설치지역 계량기함** 내 **온도 예측** 모델 개발
- AI 분석모형 : 랜덤 포레스트(RandForest)
- 시각화 데모용 : Streamlit-https://dongpa.waterai.world

구분	수도계량기함 정보					기상 정보(기상청 제공)				예측대상(Y)
특성	고정 자료(시간에 따른 변동 없음)					시계열 자료(시간 단위 데이터)				
레코드	위도	경도	고도	양지/음지	보온재 여부	기온	강수량	풍속	습도	계량기함 온도

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ IoT온도센서(원격검침단말기) 설치 지역 (28개 지자체, 140개)

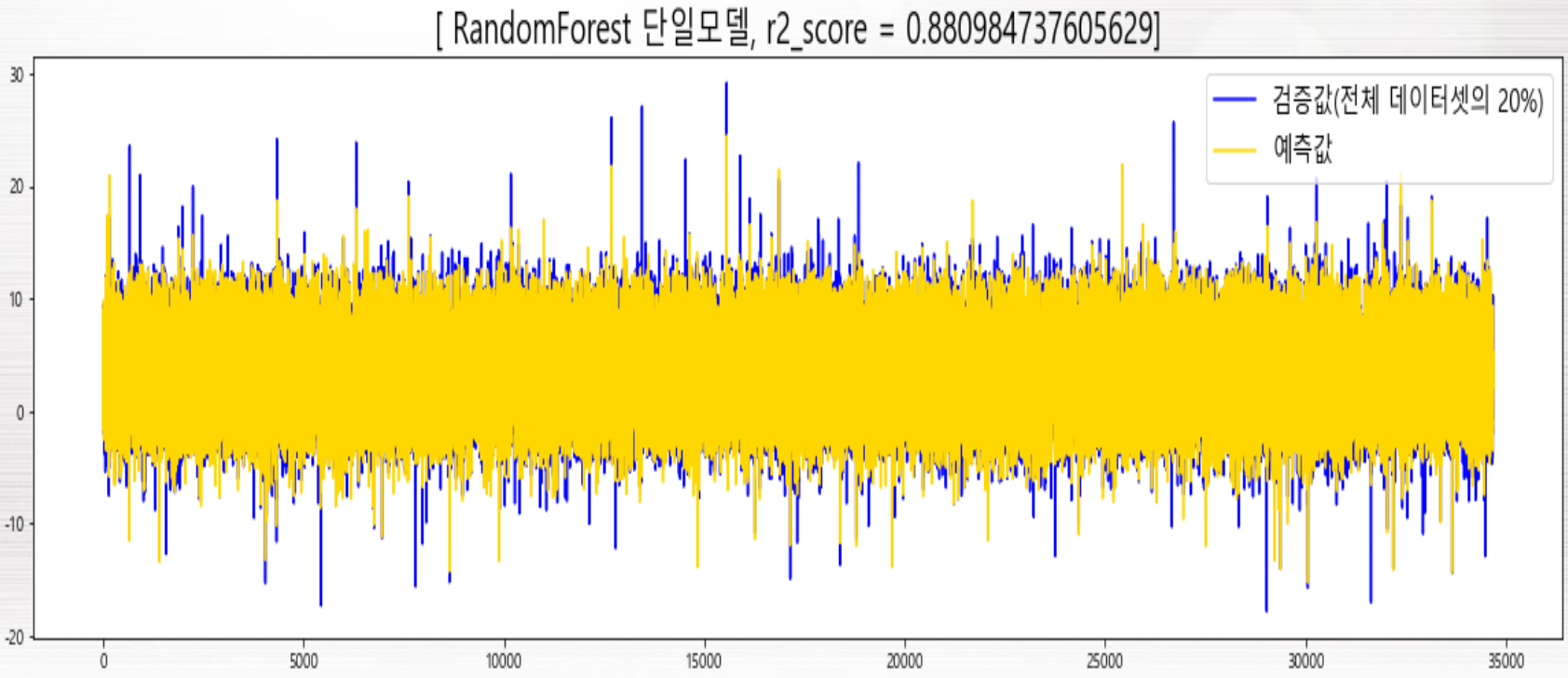


(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 모델 예측분석

□ (현황) Random Forest 기반의 AI 모델로 수도계량기함 내 온도 예측 中

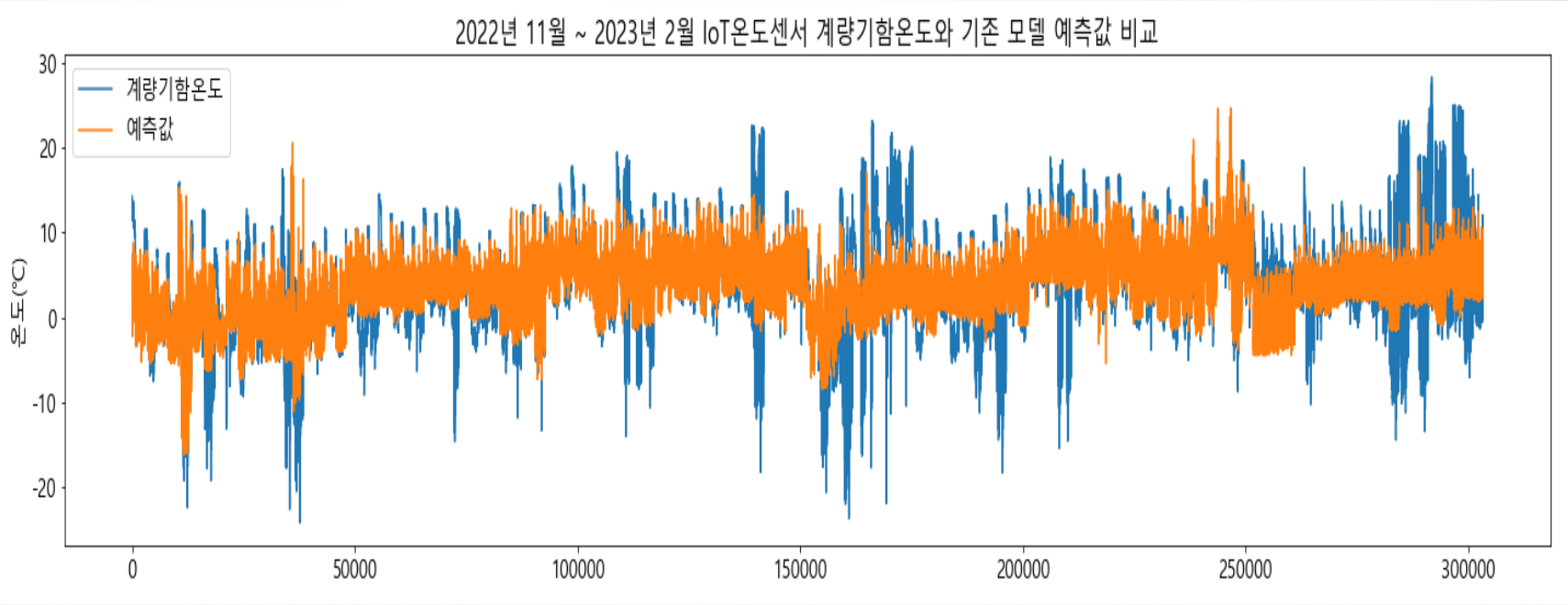
- (기존성능) 결정계수 R^2 약 0.88, 오차(MAE) ±0.9℃ 수준



(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ (실제 온도예측) 기 구축한 AI모델을 사용하여 2022년 11월 ~ 2023년 2월 계량기함 온도 예측

- (예측대상) 28개 지자체, 140개 IoT온도센서 대상 약 30만개 데이터
- (성능) 결정계수 R^2 약 0.67, 오차(MAE) ±1.8℃ 수준



[`22년 동절기 신규 온도 데이터('22.11~'23.02)와 기존 AI모델 예측값 차이]

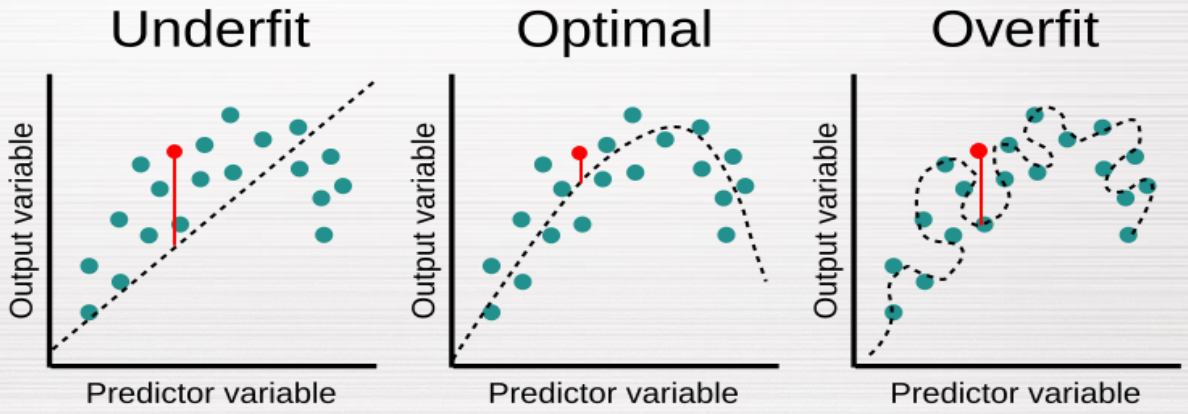
(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ (분석결과) 당초 모델 개발 시 TEST 데이터 예측 성능과 `22년 동절기 신규 취득 데이터 예측 성능에 많은 차이 발생

- 학습 데이터 부족*으로 과적합(overfitting)** 발생, 기존 모델 재학습 필요

* 학습데이터(`21.11~`22.2) : 14만개 / `22년 동절기 신규 취득 데이터(`22.11~`23.2) : 30만개
학습데이터가 경험해보지 못한 다양한 조건에서의 계량기함 온도를 모델에 반영하지 못함

* * 과적합: 모델이 학습 데이터에 지나치게 최적화되어, 새로운 데이터에 대한 예측 성능 저하



- 신규 특성인자 발굴 및 Random Forest 외 타 알고리즘 적용 검토 필요

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ Dataset (유역수도지원처 제공)

1. IoT온도센서 Dataset (수온제외 140개 센서)

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '계량기함온도']
- 515,184 rows × 14 columns

2. IoT온도센서 Dataset (수온포함 109개 센서)

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '수온', '계량기함온도']
- 351,576 rows × 15 columns

3. 경기도 광주시, 충청남도 논산시, 충청남도 서산시 Dataset(5.75GB)

- ['일시', '기온', '강수량', '습도', '풍속', 'ACCOUNT_NO', 'T_MAC', 'METER_DTM', 'TEMP', '지자체']
- 62,742,055 rows × 10 columns

* ACCOUNT_NO : 수용가번호, T_MAC : 원격검침단말기 일련번호, METER_DTM : 원격검침일시, TEMP : 계량기함온도

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ Data 전처리

1. IoT온도센서 Dataset (수온제외 140개 센서)
2. IoT온도센서 Dataset (수온포함 109개 센서)

- 지자체, 일련번호, 일시 기준으로 sorting
- 전체 Dataset row에 대하여 1일~7일전 기온을 추가 칼럼으로 insert

3. 경기도 광주시, 충청남도 논산시, 충청남도 서산시 Dataset

- 지자체 칼럼 추가, 3개 dataset union
- 'ACCOUNT_NO', 'T_MAC', 'METER_DTM' 기준으로 sorting

광주시 28,416,769
논산시 15,111,268
서산시 19,214,018
(62,742,055, 10)

- 전체 Dataset row 에 대하여 1일~7일전 기온을 추가 칼럼으로 insert

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 입출력 Parquet(파케이 - 빅데이터를 위한 파일 형식) 적용

- Apache Parquet은 효율적인 데이터 저장 및 검색을 위해 설계된 열 중심의 오픈소스 데이터 파일 형식
- 복잡한 데이터를 대량으로 처리할 수 있도록 향상된 성능과 함께 효율적인 데이터 압축 및 인코딩 체계를 제공

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ Parquet(파케이 - 빅데이터를 위한 파일 형식) 적용

- Parquet는 열 기반 데이터 처리로 csv 파일형식과 같은 전통적 방식의 행 기반 처리 기법과 비교 시 데이트의 압축률이 높고

- 필요한 열의 데이터만 읽어서 처리하기 때문에 불필요한 I/O감소로 데이터 처리 성능을 극대화

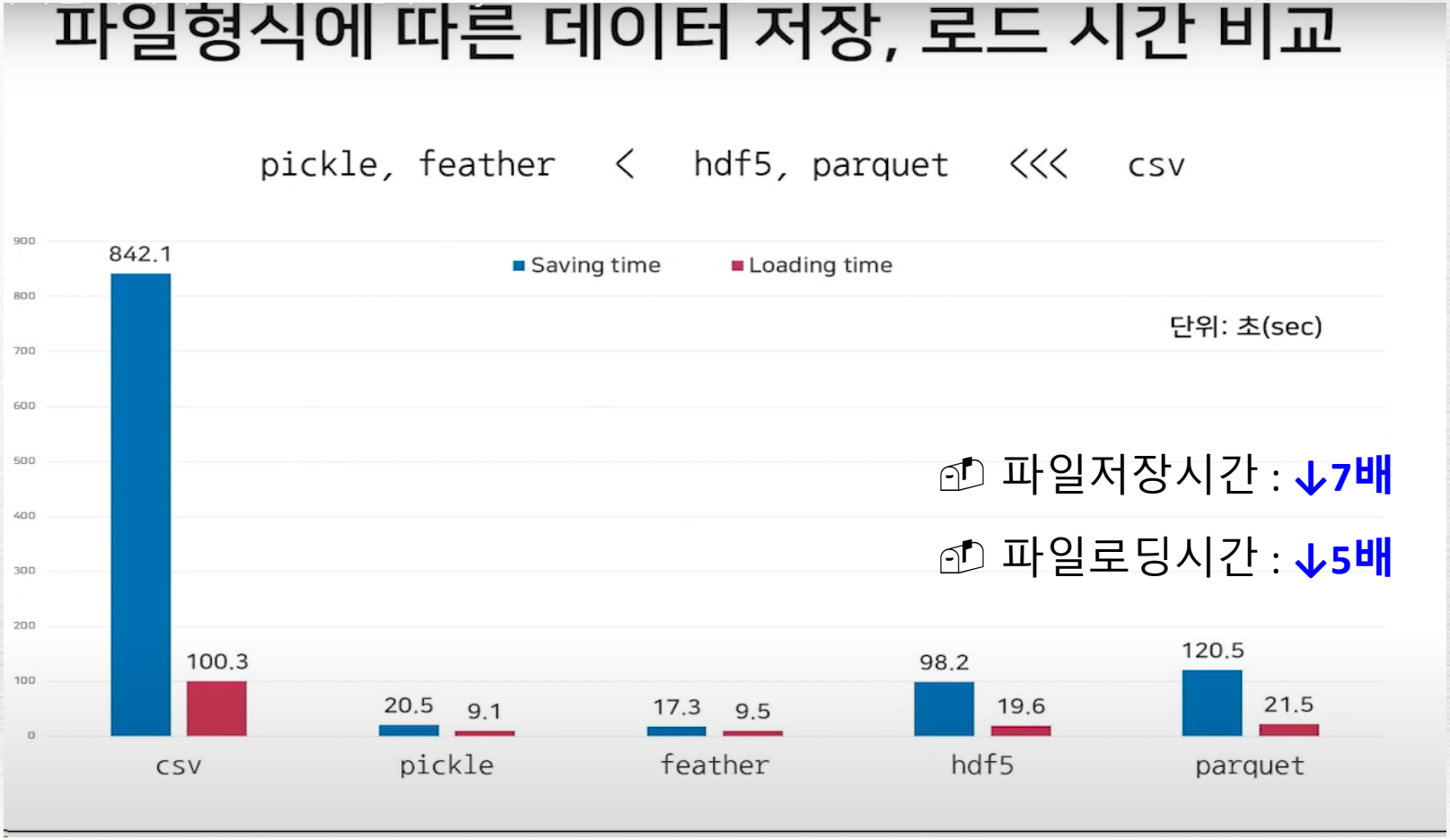
- 6천 2백만건(5.75GB) 지방상수도 수용가 정보 데이터 처리 성능비교

구분	Parquet	CSV	실행 구문
read time	19s	2min 22s	df_csv = pd.read_csv('./total.csv') df_pq = pd.read_parquet('./total.parquet', engine='pyarrow')
write time	1m 1s	9min 46s	df_csv.to_csv('./total2.csv')
file size	0.5GB	5.75GB	df_pq.to_parquet('./total.parquet',engine='pyarrow')

*** read time : 7.4배, write time : 9.6배, file size : 11배 차이**

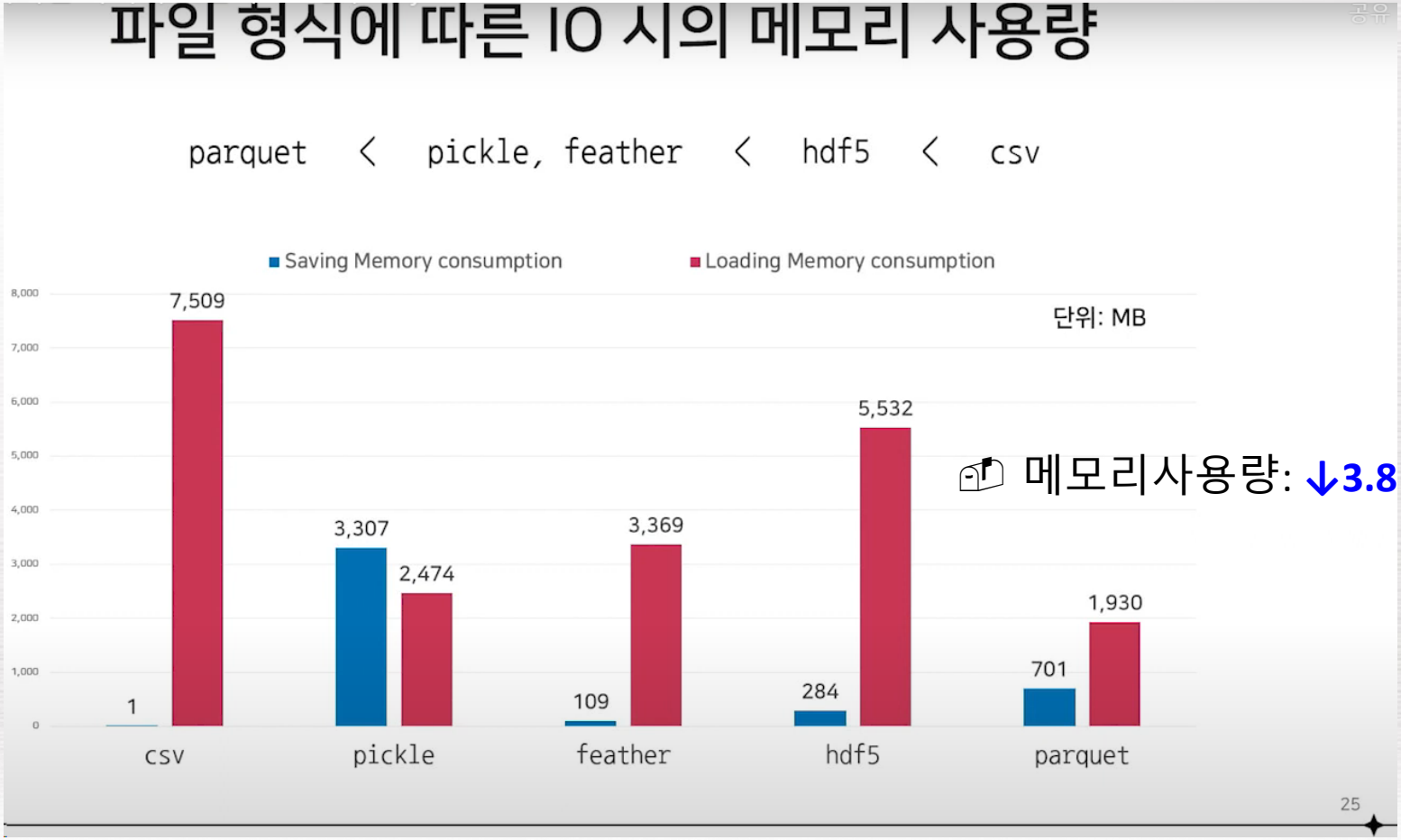
(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ CSV VS Parquet 성능비교(출처 : 파이콘)



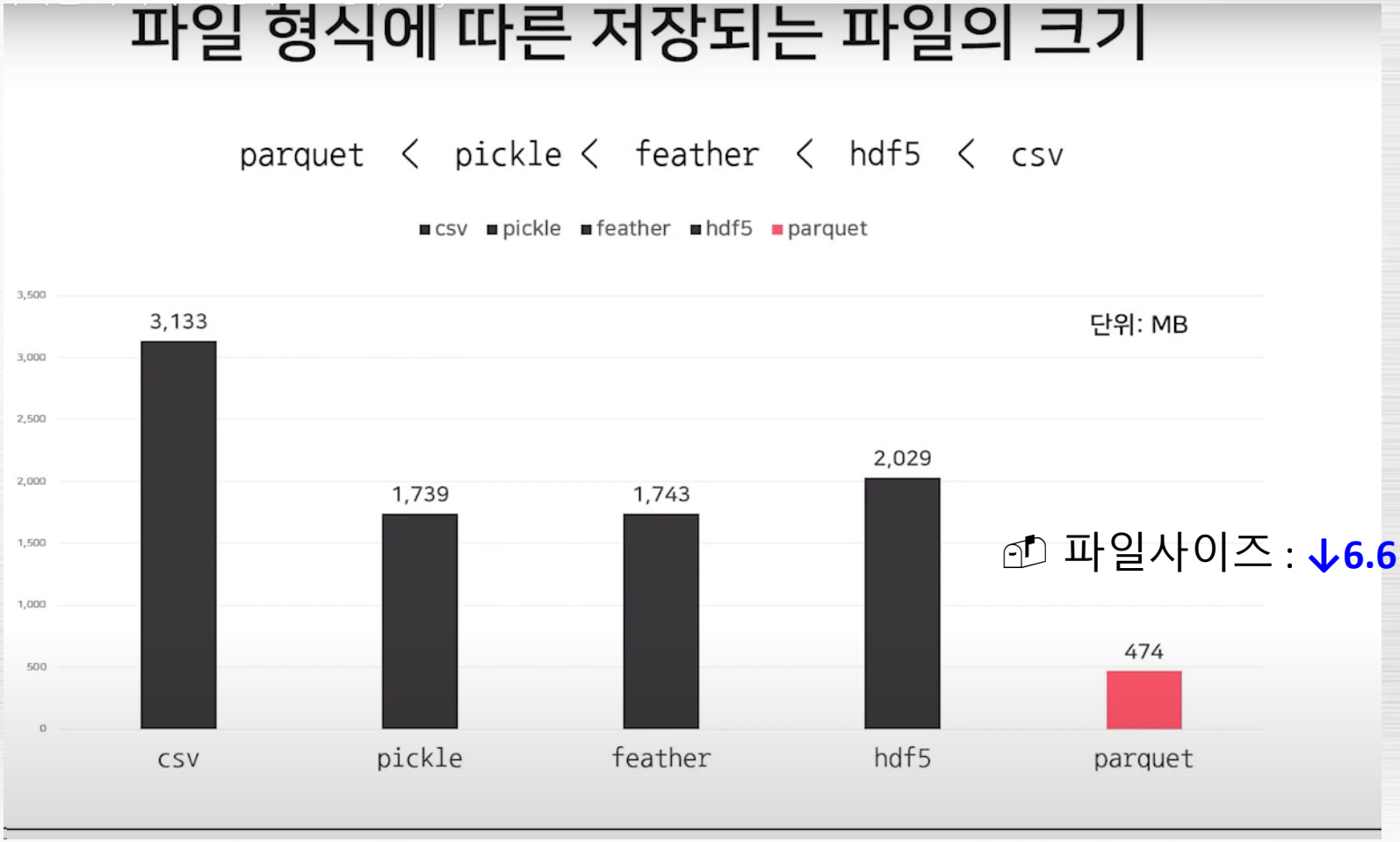
(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ CSV VS Parquet 성능비교(출처 : 파이콘)



(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ CSV VS Parquet 성능비교(출처 : 파이콘)



(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ Vectorization 적용

- 1일 ~ 7일 전 기온데이터를 전체 dataset row에서 가져와 칼럼 7개 추가 할당 시 과다 시간 소요문제 발생

1. IoT온도센서 Dataset (수온제외 140개 센서)

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '계량기함온도']
- 515,184 rows × 14 columns

2. IoT온도센서 Dataset (수온포함 109개 센서)

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '수온', '계량기함온도']
- 351,576 rows × 15 columns

3. 경기도 광주시, 충청남도 논산시, 충청남도 서산시 Dataset

- ['일시', '기온', '강수량', '습도', '풍속', 'ACCOUNT_NO', 'T_MAC', 'METER_DTM', 'TEMP', '지자체']
- 62,742,055 rows × 10 columns

* ACCOUNT_NO : 수용가번호, T_MAC : 원격검침단말기 일련번호, METER_DTM : 원격검침일시, TEMP : 계량기함온도

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ 행 또는 열 또는 전체의 셀(=원소)에 원하는 연산을 하는 일반적인 방법 : 3가지

1. Pandas iterrows() For-loop

```
distances = []

for index, row in df.iterrows():
    lev = compare_strings(row[0], row[1], email=1)
    distances.append(lev)
```

2. Pandas apply()

```
distances = df.apply(lambda x: compare_strings(x['email'], x['full_name'], email=1) ,axis=1)
```

3. Numpy vectorize()

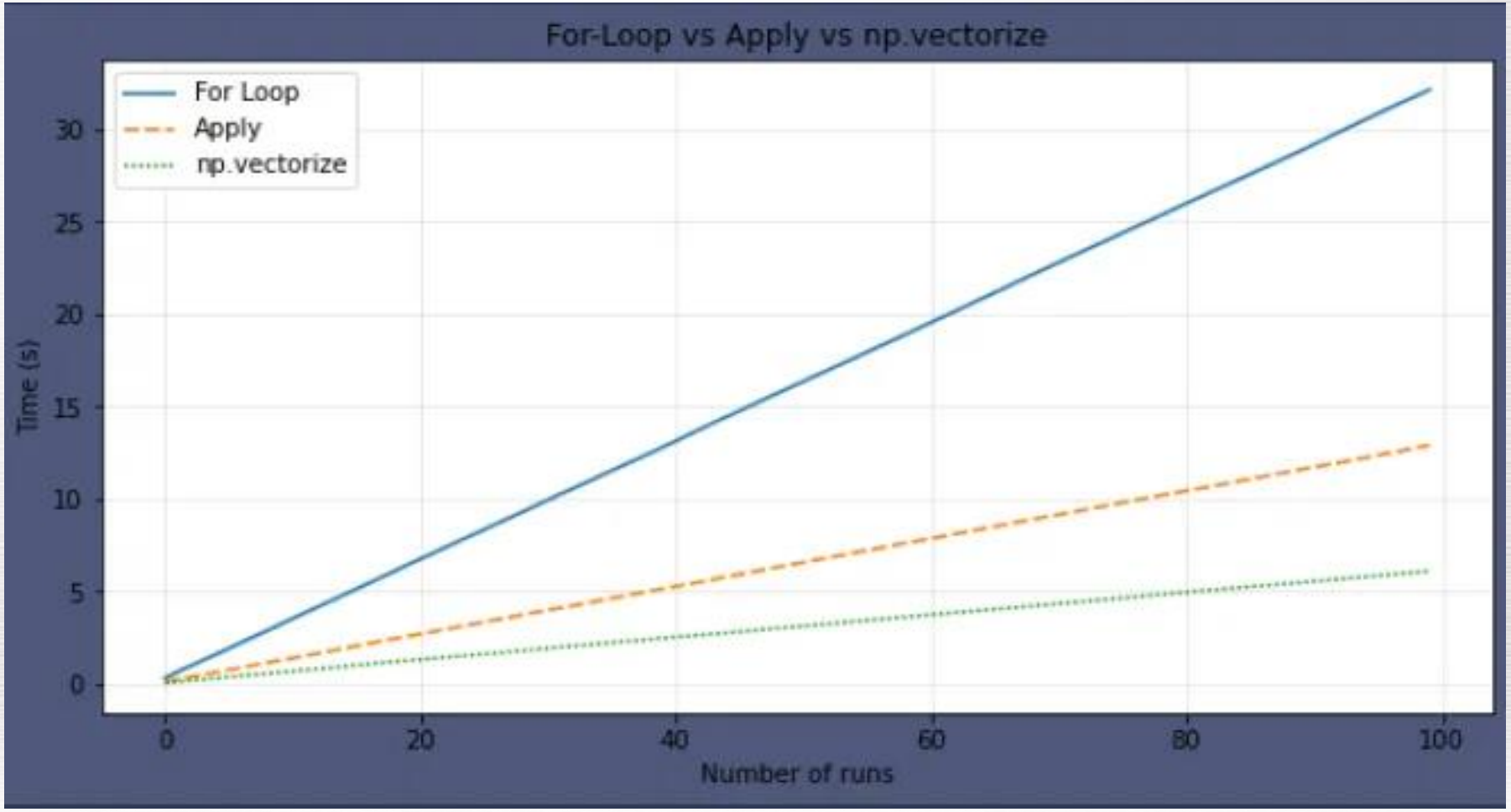
```
# first we vectorise the function compare_strings
compare_strings_vectorized = np.vectorize(compare_strings)

# the arguments we would pass to compare_strings are now passed to the vectorized function
distances = compare_strings_vectorized(df['email'], df['full_name'], email=1)
```

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ 각각 100회 실행 for 100 million rows

(출처 : https://michael-taverner.medium.com/stop-using-apply-and-start-using-numpys-vectorize-d589d15bc77b)



(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ **Pandas apply()**

* 50 만건 dataset 적용시(100m), 6천만건 dataset 적용시 처리 시간 측정불가

```
def get_previous_temperatures(row):
    previous_temperatures = []
    for i in range(1, 8):
        date = row['일시'] - pd.DateOffset(days=i)
        sens_id = row['일련번호']
        previous_temp = df.loc[(df['일시'] == date) & (df['일련번호'] == sens_id), '계량기함온도'].values

        if len(previous_temp) > 0:
            previous_temperatures.append(previous_temp[0])
        else:
            previous_temperatures.append(None)

    return pd.Series(previous_temperatures, index=[f'day{i}_back' for i in range(1, 8)])

new_columns = [f'mday{i}_back' for i in range(1, 8)]
df[new_columns] = df.apply(get_previous_temperatures, axis=1)
```

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ **Numpy vectorize() 으로 Vectorization 적용**

- Pandas Apply function → Numpy vectorize function 적용 : 월등한 속도 향상

  * 50 만건 dataset 적용시(100m → 4m), 6천만건 dataset 적용시 8시간 소요(측정불가 →8 hour)

```
df['일시'] = pd.to_datetime(df['일시'], format='%Y-%m-%d %H:%M:%S')

prev_temps_dict = {}

vectorized_func = np.vectorize(lambda date, sens_id, temp: prev_temps_dict.update({(date,
                        sens_id): temp}))

vectorized_func(df['일시'], df['일련번호'], df['기온'])

new_columns = [f'temp{i}_back' for i in range(1, 8)]

for i, col in enumerate(new_columns):
    hours_offset = (i + 1) * 24
    df[col] = np.vectorize(lambda date, sens_id: prev_temps_dict.get((date –
                pd.DateOffset(hours=hours_offset), sens_id)))(df['일시'], df['일련번호'])

display(df)
```

(컨설팅) 수도미터 동파경보 AI모델 고도화

K water

◈ **Numpy Vectorization 계산**

- Pandas Apply function → Numpy vectorize function 적용 : 월등한 속도 향상

* 50 만건 dataset 적용시(4m → 2m), 6천만건 dataset 적용시 8시간 소요(8 hour → 5 hour 30m)

** 다음 코드는 shift function 처리형태로 정확한 이전 기온을 가져오기 위하여 recursive numpy array 조회가 가능토록 수정이 필요함

```
df_np = df.to_numpy()

# Create the new columns for previous temperatures
new_columns = ['temp{}_back'.format(i) for i in range(1, 8)]

for i, col in enumerate(new_columns):
  hours_offset = (i + 1) * 24
  prev_temps = np.where(df_np[:, 1> (df_np[:, 1]– pd.DateOffset(hours=hours_offset)), df_np[:, 9], np.nan)
  df[col] = prev_temps[:, 0]

# Display the updated DataFrame
display(df)
```

(컨설팅) 수도미터 동파경보 AI모델 고도화

j:\python\동파\동파수온적용B.ipynb > df (328412, 21)

	index	일련번호	위도	경도	고도	양지	음지	보온재...	기온	강수량	습도	풍속	수온	계량기...	temp1_back	temp2_back	temp3_back	temp4_back	temp5_back	temp6_back	temp7_back
0	49968	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.52456...	-0.04772...	-0.57272...	-0.76853...	-1.41626...	-4.4	-0.8798596305	-0.9151440803	-0.4795491067	-0.2182739894	-1.0387806671	-1.1082815229	-0.8123797182
1	49969	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.62918...	-0.04772...	-0.32333...	-0.96734...	-1.28559...	-4.4	-0.8415038982	-0.897707602	-0.6364606532	-0.078752096	-1.1259567947	-1.1954201256	-0.882088211
2	49970	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.47225...	-0.04772...	-0.61507...	-0.63599...	-1.54694...	-4.6	-0.8031481658	-0.9151440803	-0.7759375834	-0.131072806	-1.0387806671	-1.2477032871	-1.1434950589
3	49971	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.50713...	-0.04772...	-0.50213...	-0.83480...	-1.54694...	-4.7	-0.7647924335	-1.0895088635	-0.7933721997	-0.183393516	-1.3177442756	-1.177992405	-1.0912136893
4	49972	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.69893...	-0.04772...	-0.27627...	-0.76853...	-1.54694...	-4.8	-0.7264367011	-1.1592547767	-1.0200222113	-0.3229154094	-1.4746613054	-1.4568359334	-1.2654849213
5	49973	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.66405...	-0.04772...	-0.58213...	-0.43718...	-1.54694...	-5	-0.6880809688	-1.22900069	-1.124629909	-0.4973177762	-1.6141431097	-1.5265468155	-1.4397561532
6	49974	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.73380...	-0.04772...	-0.34215...	-0.90107...	-1.54694...	-5.1	-0.6497252365	-1.2813101249	-1.2118029904	-0.4973177762	-1.6490135608	-1.6136854181	-1.4746103996
7	49975	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.64662...	-0.04772...	-0.73741...	-0.50345...	-1.54694...	-5.3	-0.6113695041	-1.4033654732	-1.0548914439	-0.4275568295	-1.7187544629	-1.8925289465	-1.5791731388
8	49976	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.75123...	-0.04772...	-0.67624...	-0.70226...	-1.28559...	-5.5	-0.5730137718	-1.5428572997	-1.1420645253	-0.6019591962	-1.6664487863	-1.7356794618	-1.5966002619
9	49977	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.29789...	-0.04772...	-1.19855...	-0.23837...	-1.54694...	-5.5	-0.5346580394	-1.1069453418	-0.8631106648	-0.3577958828	-1.3351795012	-1.3871250513	-1.1609221821
10	49978	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.01891...	-0.04772...	-1.39147...	0.026702...	-1.28559...	-5.5	-0.4963023071	-0.6535969055	-0.3226375602	-0.2531544627	-0.6726409309	-0.7597271125	-0.7252441022
11	49979	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.84455...	-0.04772...	-1.42912...	0.556860...	-1.54694...	-5.6	-0.4579465748	-0.2002484693	-0.1657260136	-0.2008337527	-0.0798432627	-0.3240340994	-0.2372846528
12	49980	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.61788...	-0.04772...	-1.52793...	1.418366...	-1.28559...	-5.6	-0.4195908424	-0.0084472078	-0.1134221648	0.1130905075	0.3211669245	-0.1671846147	-0.16757616
13	49981	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.39121...	-0.04772...	-1.60322...	0.888209...	-1.54694...	-5.4	-0.3812351101	-0.0084472078	0.0260547654	-0.0264313859	0.5826953075	0.0942311932	0.0589764415
14	49982	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.23428...	-0.04772...	-1.61263...	1.087018...	-1.54694...	-5.4	-0.3428793778	0.0264257489	-0.0088144671	-0.0264313859	0.7570475629	0.1813697958	0.1112578111
15	49983	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.32146...	-0.04772...	-1.52793...	1.683445...	-1.54694...	-5.2	-0.3045236454	0.1310446188	0.0260547654	0.0782100341	0.826788465	0.3382192805	0.0764035647
16	49984	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.40864...	-0.04772...	-1.33030...	0.821939...	-1.28559...	-5.1	-0.3568269168	0.0438622272	-0.0436836997	0.0782100341	0.6350009841	0.32079156	0.1286849343
17	49986	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-0.86199...	-0.04772...	-1.22678...	0.026702...	-1.54694...	-4.9	-0.5398883666	-0.461795644	-0.5318529555	-0.2705946994	0.1119442181	-0.1846123352	-0.3418473919
18	49987	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.07122...	-0.04772...	-1.13267...	-0.30464...	-1.28559...	-4.8	-0.7229498164	-0.514105079	-0.5667221881	-0.4624373028	-0.0624080372	-0.2717509378	-0.4986915007
19	49988	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.17584...	-0.04772...	-1.02915...	-0.37091...	-1.54694...	-4.8	-0.9060112662	-0.7233428188	-0.810806816	-0.4275568295	-0.3239364202	-0.4286004225	-0.620681363
20	49989	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.28046...	-0.04772...	-0.75153...	-0.63599...	-1.54694...	-4.8	-1.089072716	-0.8279616887	-0.9502837462	-0.3229154094	-0.3239364202	-0.5157390251	-0.8472339646
21	49990	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.38507...	-0.04772...	-0.44096...	-0.96734...	-1.54694...	-4.9	-1.3505890728	-0.9500170369	-1.002587595	-0.3403556461	-0.1147137138	-0.7422993919	-0.8995153342
22	49991	062199R...	1.261397...	-0.29369...	-0.26910...	-1.23303...	1.233032...	0	-1.38507...	-0.04772...	-0.29509...	-0.76853...	-1.54694...	-4.9	-1.4377611918	-0.8802711236	-0.8805452811	-0.3403556461	-0.2367602925	-0.7597271125	-0.9169424574

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ feature 추가 및 신규 data 변경에 따른 모델 재학습

1. IoT온도센서 Dataset (수온제외 140개 센서)

****
Average Performance
****

MAE : Random Forest	0.97	XGBoost	1.44	AdaBoost	3.36	Gradient Boosting	2.14	LightGBM	1.68	CatBoost	1.61
MSE : Random Forest	2.52	XGBoost	4.50	AdaBoost	17.16	Gradient Boosting	8.99	LightGBM	5.82	CatBoost	5.40
RMSE : Random Forest	1.56	XGBoost	2.09	AdaBoost	4.10	Gradient Boosting	2.96	LightGBM	2.38	CatBoost	2.29
R2 : Random Forest	0.91	XGBoost	0.83	AdaBoost	0.35	Gradient Boosting	0.66	LightGBM	0.78	CatBoost	0.80

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '계량기함온도']
  - 515,184 rows × 14 columns

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ feature 추가 및 신규 data 변경에 따른 모델 재학습

2. IoT온도센서 Dataset (수온포함 109개 센서)

****
Average Performance
****

MAE : Random Forest	0.58	XGBoost	0.90	AdaBoost	2.70	Gradient Boosting	1.46	LightGBM	1.05	CatBoost	1.02
MSE : Random Forest	1.23	XGBoost	2.02	AdaBoost	11.89	Gradient Boosting	4.82	LightGBM	2.58	CatBoost	2.48
RMSE : Random Forest	1.08	XGBoost	1.39	AdaBoost	3.38	Gradient Boosting	2.15	LightGBM	1.57	CatBoost	1.54
R2 : Random Forest	0.92	XGBoost	0.87	AdaBoost	0.22	Gradient Boosting	0.68	LightGBM	0.83	CatBoost	0.84

- ['지자체', '일시', '일련번호', '위도', '경도', '고도', '양지', '음지', '보온재유무', '기온', '강수량', '습도', '풍속', '수온', '계량기함온도']
- 351,576 rows × 15 columns

(컨설팅) 수도미터 동파경보 AI모델 고도화

◈ feature 추가 및 신규 data 변경에 따른 모델 재학습(Random Forest)

3. 경기도 광주시, 충청남도 논산시, 충청남도 서산시 Dataset 학습

```
********************************
        Average Performance
********************************
 MAE:    4.239729690529015
--------------------------------------
 MSE: 136.71894962956364
--------------------------------------
 RMSE: 11.692269308888656
--------------------------------------
 R2    :  0.234996311815770 63
--------------------------------------
```

- ['일시', '기온', '강수량', '습도', '풍속', 'ACCOUNT_NO', 'T_MAC', 'METER_DTM', 'TEMP', '지자체']
  - 62,742,055 rows × 10 columns

* ACCOUNT_NO : 수용가번호, T_MAC : 원격검침단말기 일련번호, METER_DTM : 원격검침일시, TEMP : 계량기함온도

***타 dataset에는 있는 ' 위도 ' , ' 경도 ' , ' 고도 ' , ' 양지 ' , ' 음지 ' , ' 보온재유무' 부재로 인한 예측값 저하**

(컨설팅) 수도미터 동파경보 기능 고도화

◈ 향후계획

□ (대용량 처리) 수용가 dataset에 대한 DASK, CUDF, 병렬처리 등 적용 검토
* 유역수도지원처에서 추가 보완 dataset (위,경도 등) 제공 예정

□ (모델 선정/튜닝) 하이퍼파라미터 수정을 통한 성능 개선 실시

□ (시각화 데모용 모듈) AI 분석 모델 변경에 따른 시각화 데모용 모듈 고도화

Machines take me by surprise with great frequency.

Alan Mathison Turing

Q & A

감사합니다