

전력 예측 프로젝트

Team | ML 1팀

21기 김연주 21기 송상현

22기 금강산 22기 남수빈

CONTENTS

01

주제 소개

- DACON 프로젝트 소개
- 변수 설명

02

EDA

- 데이터 분포 및 상관관계 분석
- 접근 방향 정리

03

전처리

- 범주형 변수 인코딩
- 결측치 처리
- 이상치 처리
- 파생 변수 생성

04

결과

- 모델링 1
- 모델링 2





01. 주제 소개

01. 주제 소개

DACON

DATA TO VALUE

2025 전력사용량 예측 AI 경진대회

#알고리즘 #정형 #시계열 #에너지 #SMAPE

프로젝트 목표

- 건물의 **전력사용량**을 예측할 수 있는 AI 모델 개발

데이터 개요

- 10개 유형* 100개 건물의 전력소비량 (1시간 단위, 2024.06.01 ~ 08.31)
- 기상데이터 (기온, 강수량, 풍속, 습도, 일조, 일사)
- 건물 개요 정보 (연면적, 냉방 면적, 태양광 용량, ESS 용량 등)
- Train: 2024.06.01 ~ 08.24 (85일)
- Test: 2024.08.25 ~ 08.31 (7일)

주요 연구 질문

- 기상/건물 특성 중 전력 사용량에 가장 큰 영향을 주는 요인은 무엇인가?
- 건물별 특성을 반영한 맞춤형 모델과 통합형 모델 중 어떤 접근이 더 효과적인가?
- 시간별 전력 사용 패턴을 어떻게 반영해야 예측 성능을 극대화할 수 있는가?

01. 변수 설명

<식별자>

- num_date_time: 건물 번호와 시간으로 구성된 고유 ID
- building_number: 건물 번호 (1 ~ 100)

<건물 정보>

- building_type: 건물 유형 (공공, 학교, 백화점, 병원 등 10종)
- total_area: 건물 연면적 (m²)
- cooling_area: 냉방 면적 (m²)
- solar_power_capacity: 태양광 발전 설비 용량 (kW)
- ESS_capacity: 에너지 저장장치(ESS) 저장 용량 (kWh)
- PCS_capacity: 전력 변환장치(PCS) 용량 (kW)

<시계열 데이터>

- temperature(°C): 기온
- rainfall(mm): 강수량
- wind_speed(m/s): 풍속
- humidity(%): 습도
- sunshine(hr): 일조 시간
- solar_radiation(MJ/m²): 일사량

<타겟>

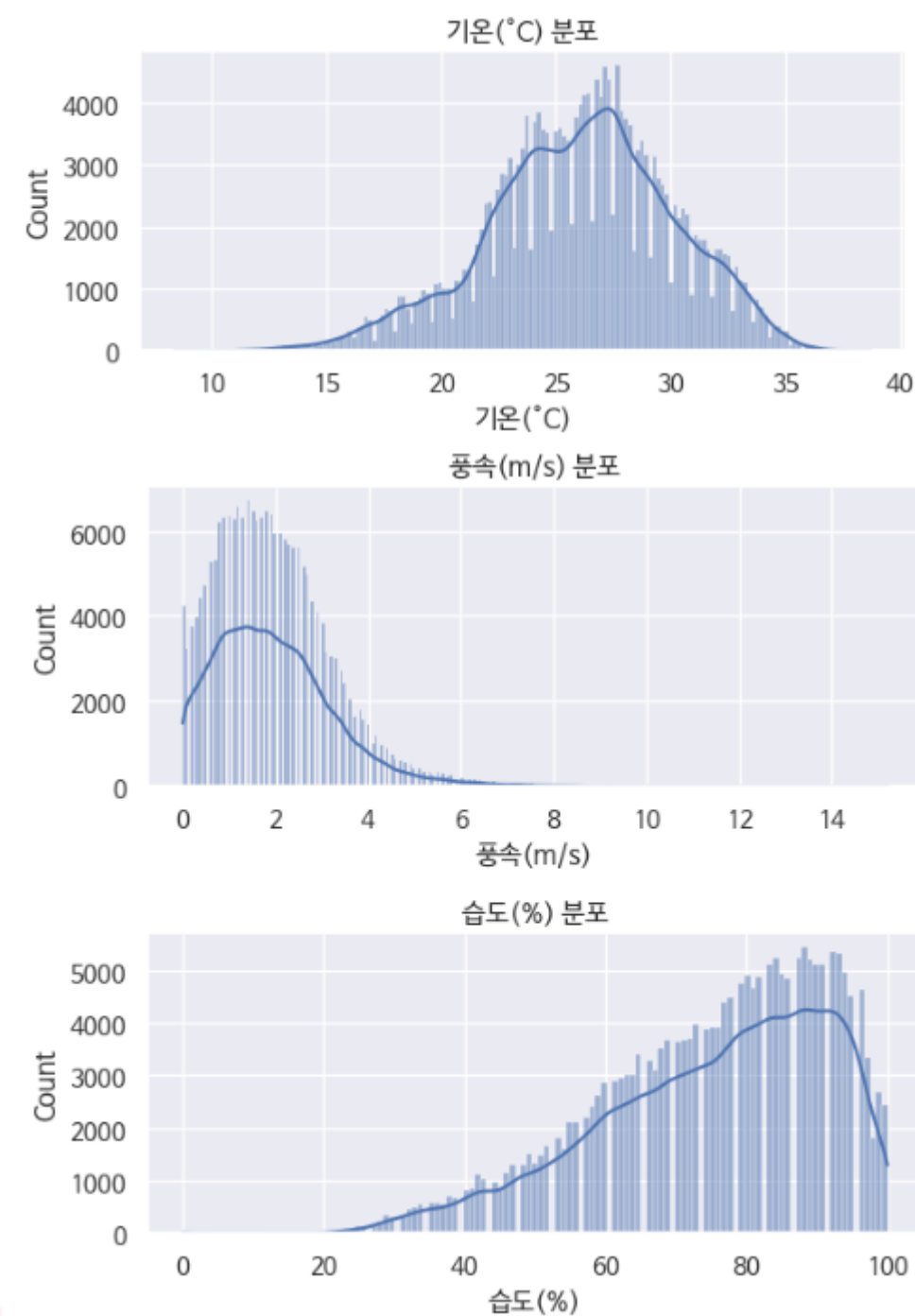
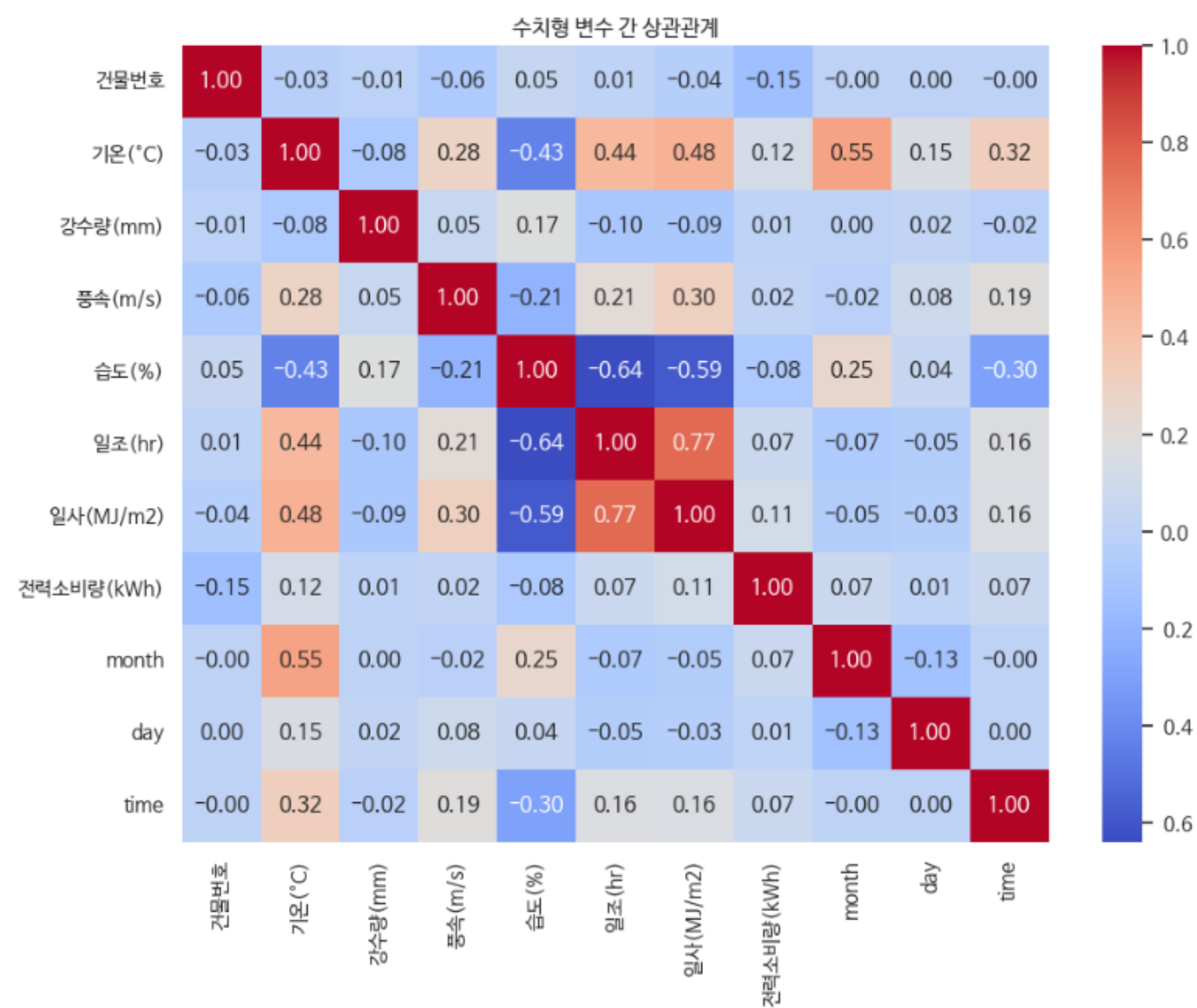
- power_consumption(kWh): 전력 사용량 (train에만 존재)



02. EDA

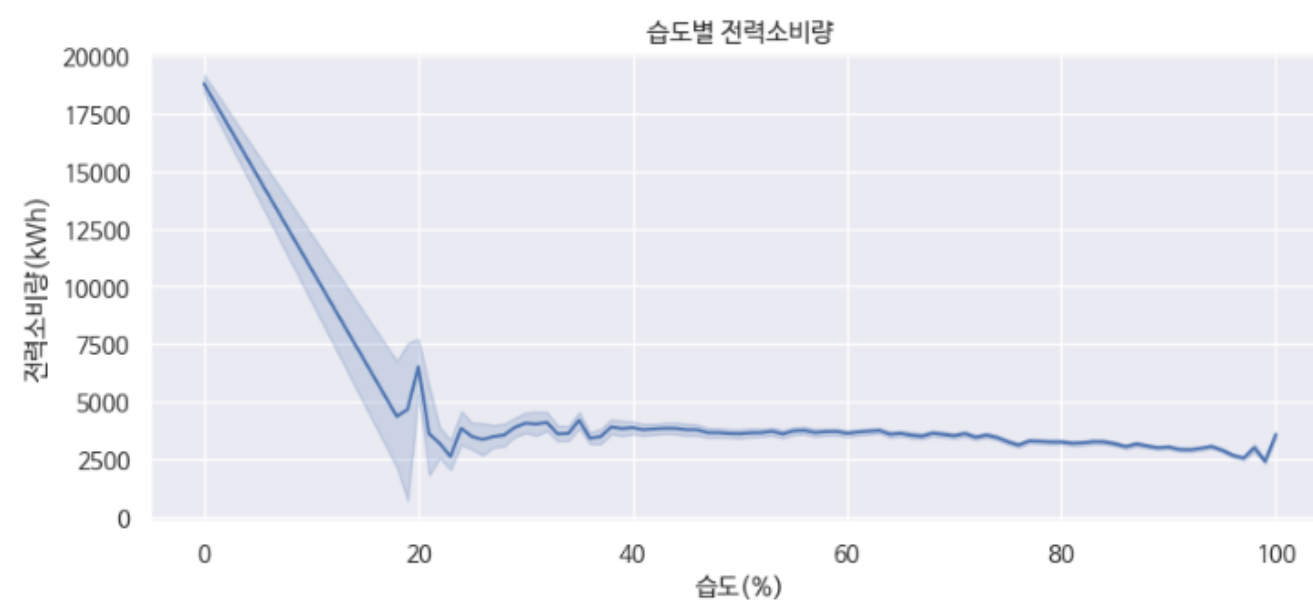
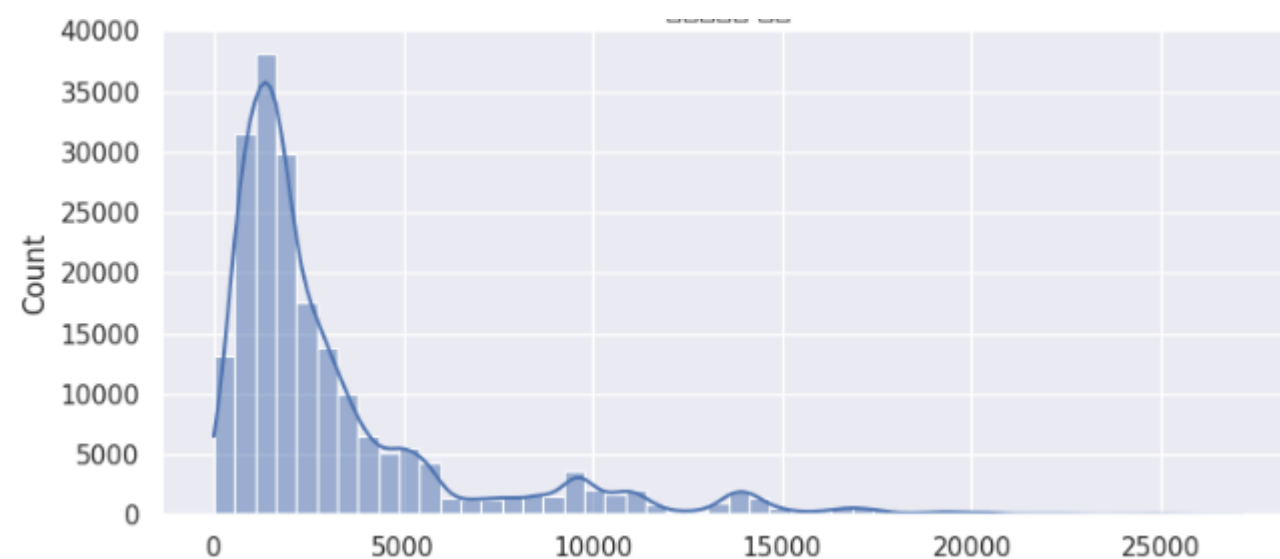
02. EDA

- 데이터 분포 확인



02. EDA

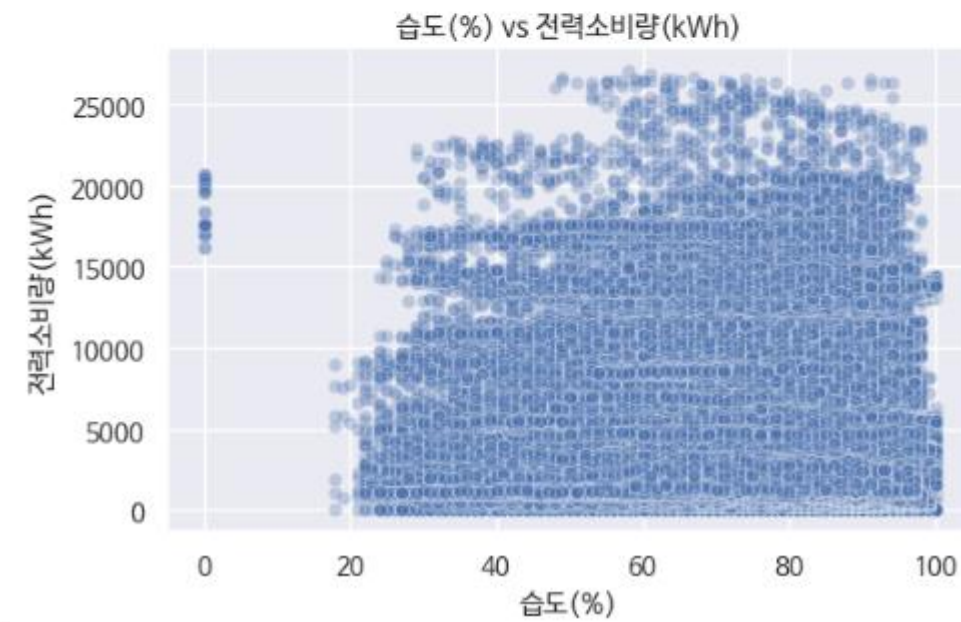
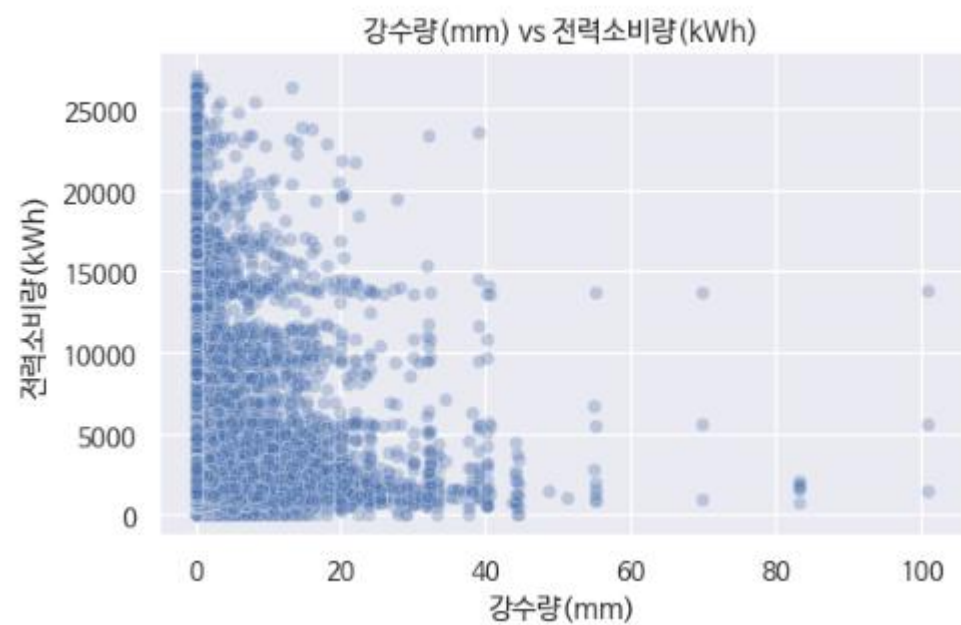
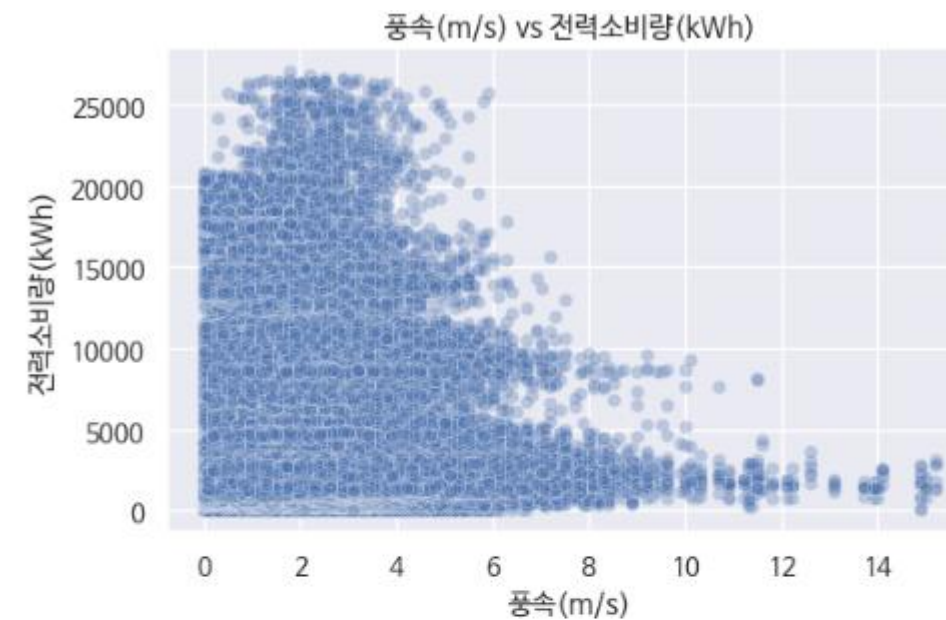
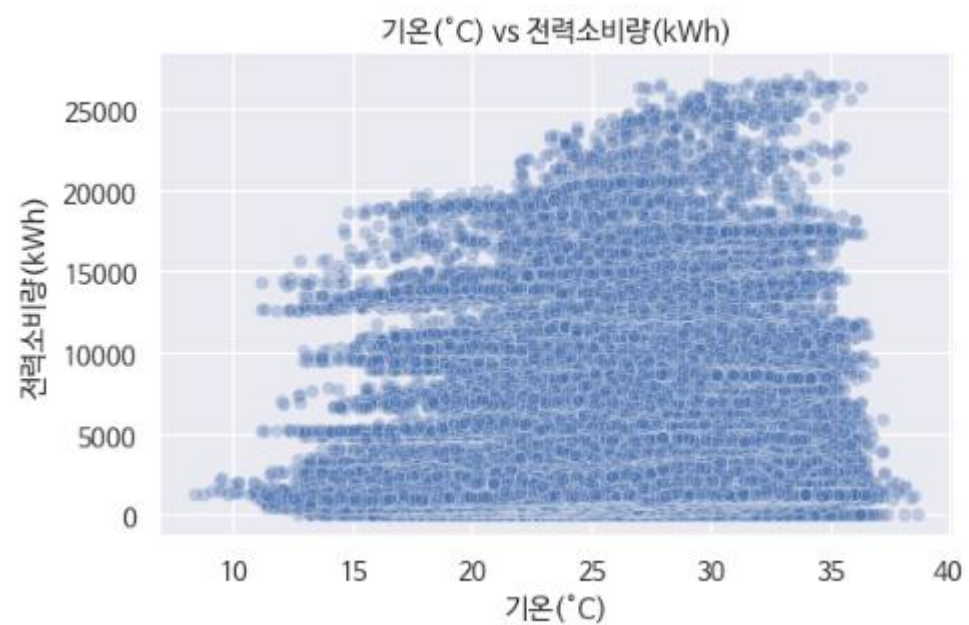
Target : 전력 소비량 분포



	building_type	consumption
0	IDC(전화국)	10316.94
1	병원	4454.06
2	학교	3462.68
3	호텔	3175.02
4	백화점	2729.74
5	상용	2513.70
6	건물기타	2285.96
7	연구소	2111.67
8	공공	1625.91
9	아파트	1106.31

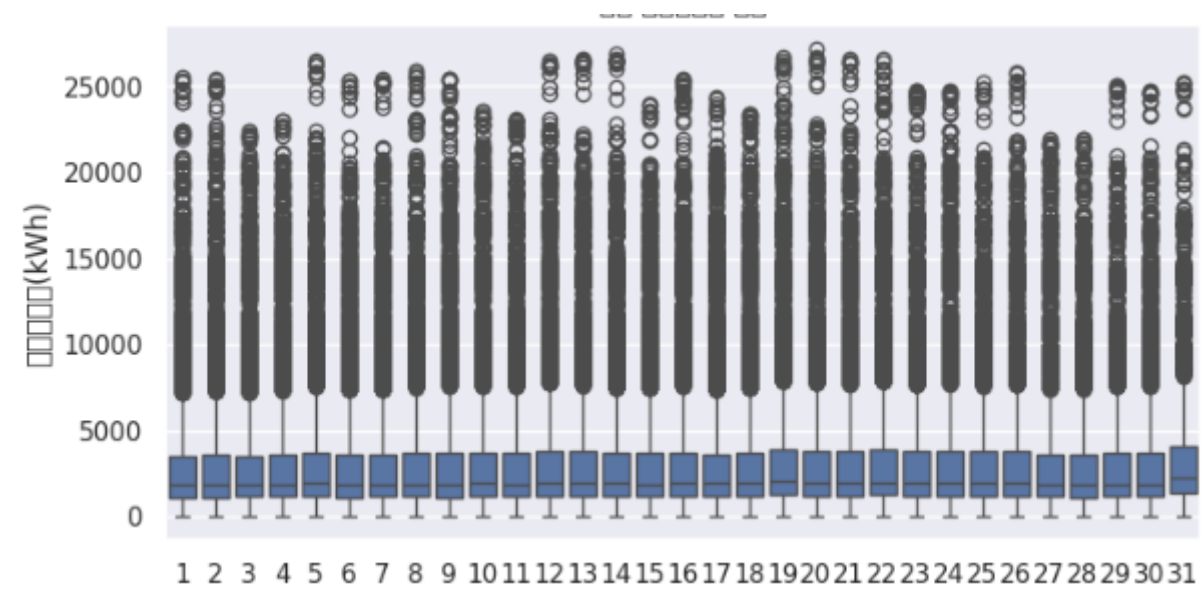
02. EDA

기온, 풍속, 강수량, 습도 - 전력소비량 분포

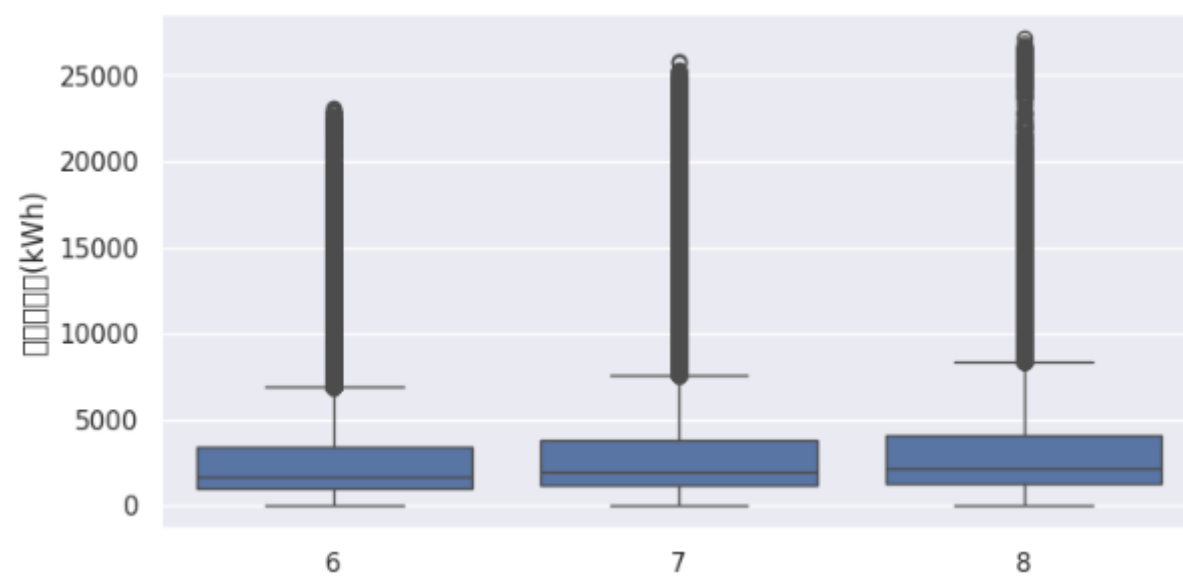


02. EDA

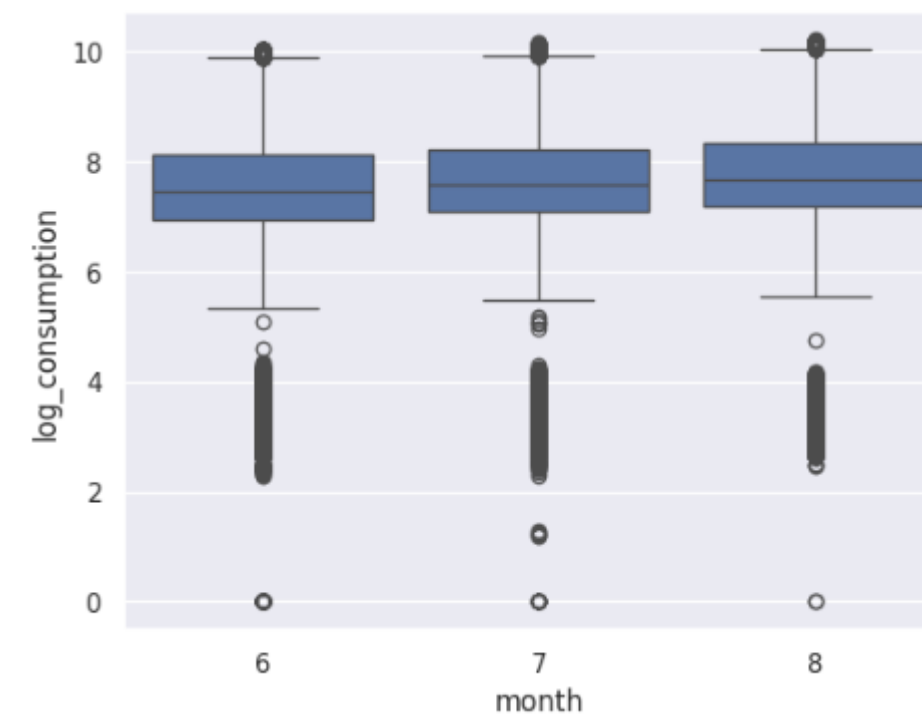
일별 전력 소비량 분포



월별 전력 소비량 분포

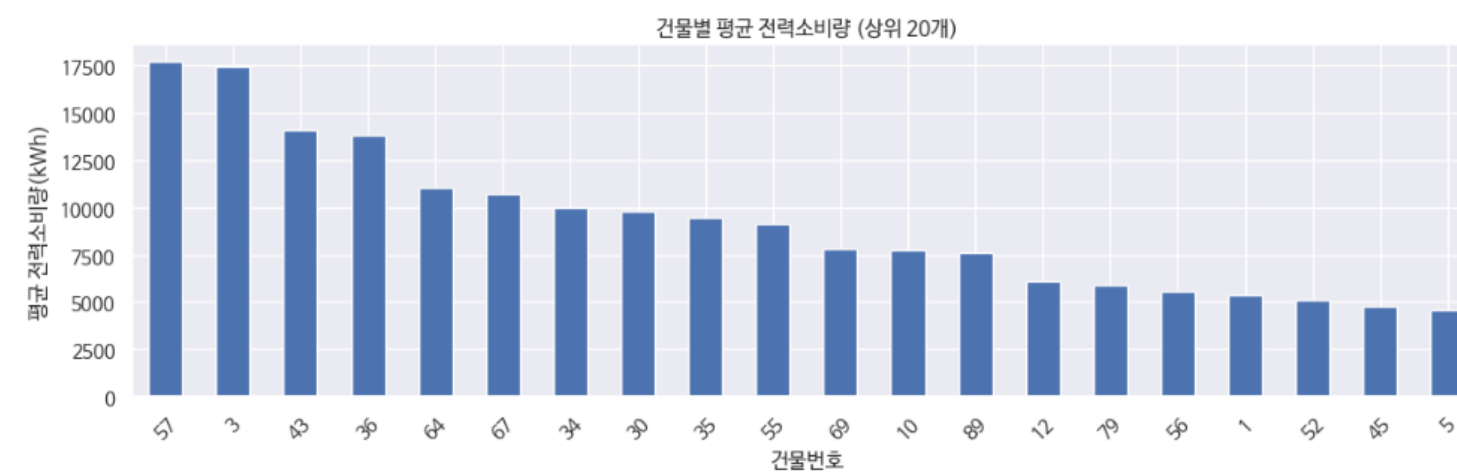
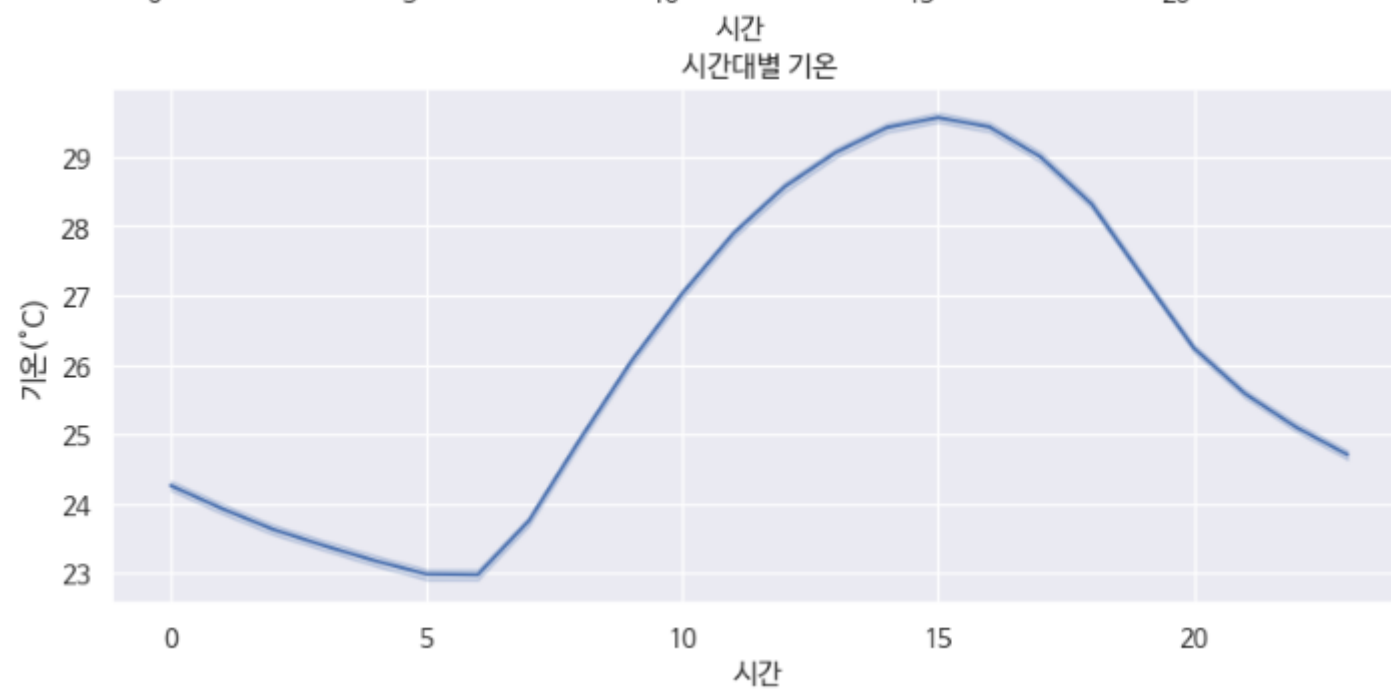
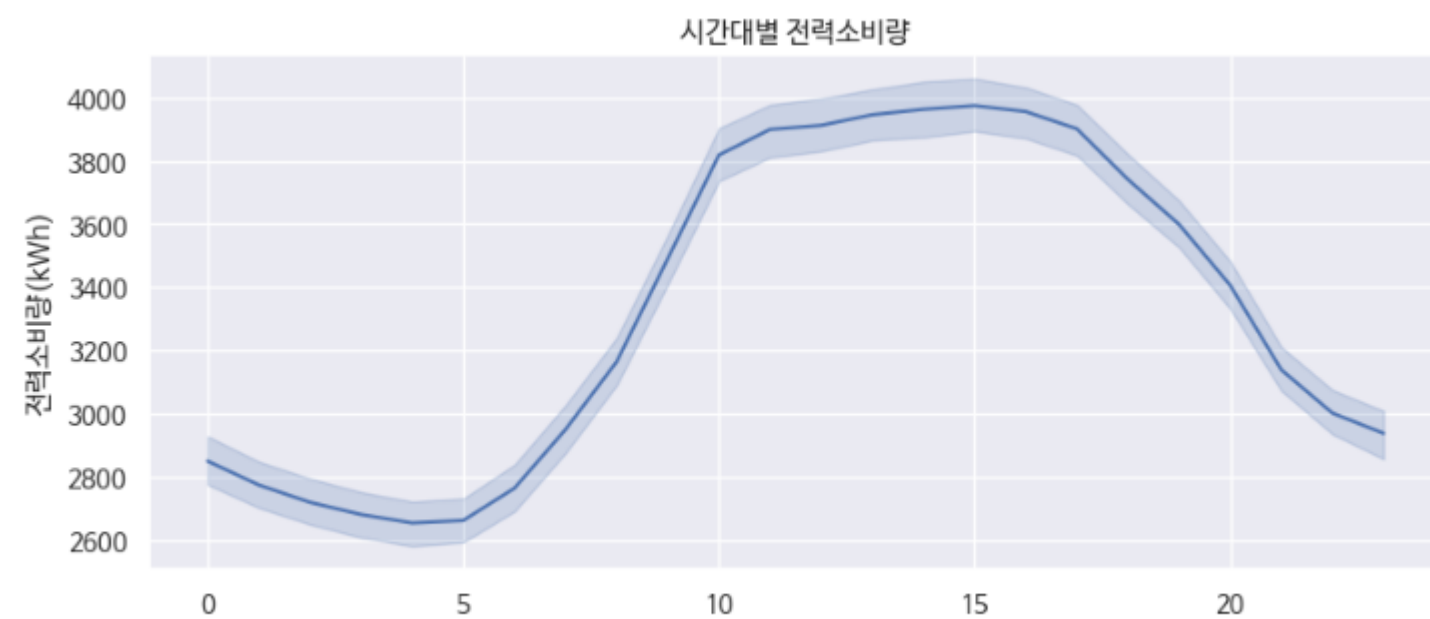


월별 로그 전력 소비량 분포



02. EDA

Target : 전력 소비량 분포 (시간대별 / 건물별)

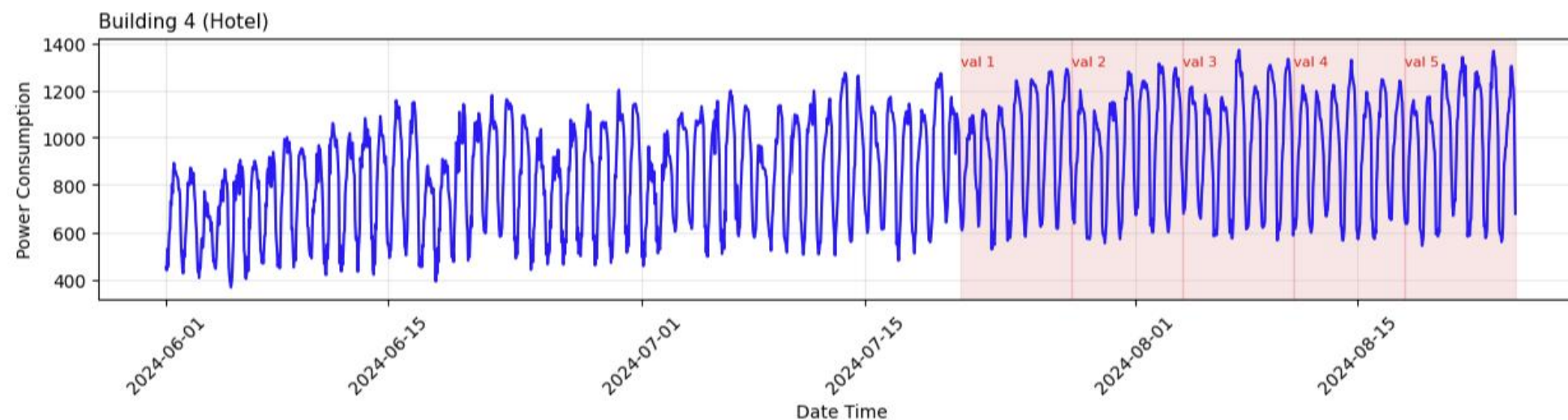




03. 전처리 - 모델 1

03. 전처리 - 모델 1

- 로그 변환
- 전력사용량의 분포가 긴 꼬리를 가지고 분산이 크므로, 로그 변환으로 분포 안정화
- 주기성 인코딩(sine / cos 변환)
- 단순 정수 인덱스 → 순환적 성격 반영 X, 사인/코사인 변환으로 연속적으로 주기적인 관계 학습 가능하게
 - ex. 23시와 00시가 가까운 시간대라는 주기 학습
- 여름 기간 진행률 사이클릭 특성 : **day_of_year**



- 전력 사용량의 특성 : 6월 ~ 8월까지 전력사용량 우상향 확인 + test 기간이 8월 말의 경우 사용량이 꺾이는 드리프트 가능성 존재 → 자연스러운 여름주기성을 학습시키기 위함

03. 전처리 - 모델 1

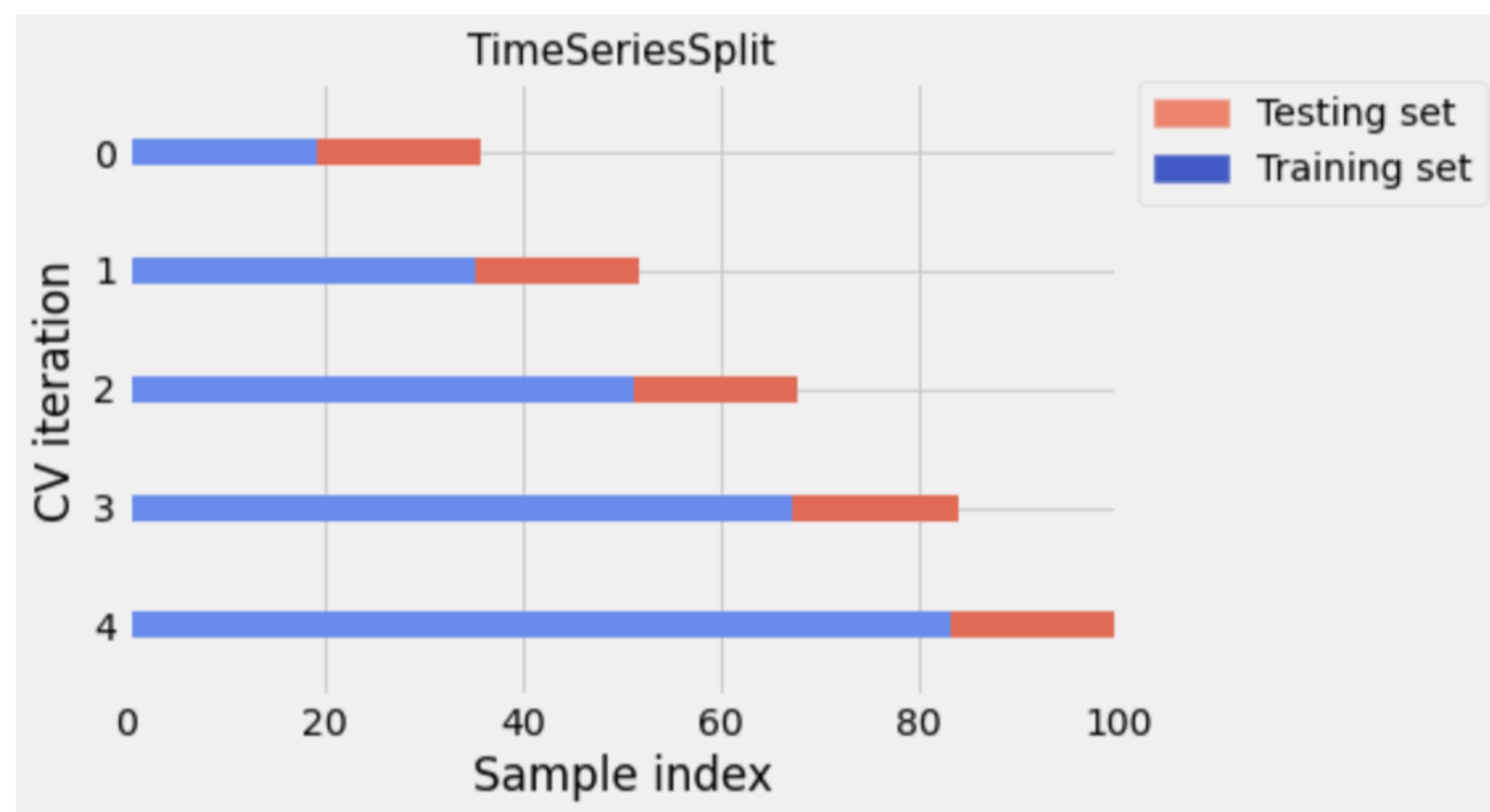
- 전일 기온 기반 파생 피쳐
- 건물별 일별 기온 요약치(max_temp, mean_temp, min_temp) 구해 전일 값으로 시프트
 - 왜 당일의 기온 요약치가 아닌 전일을 사용?
 - 당일의 요약치를 사용하는 경우, 만약 val set의 정보까지 포함하여 train set을 학습할 수 있음 → leakage 방지를 위함
- 전력사용과 관련된 파생변수 및 기상 지표
- cooling_area / total_area → cooling_ratio : 건물별 냉방 수요 민감도
- THI(Temperature-Humidity Index, 온습도 지수)
- CDH (Discomfort Index, 불쾌지수)
- HI (Heat Index, 체감온도)

03. 전처리 - 모델 1

- 휴일 처리
 - `holiday = 1` if 주말(`dow >= 5`) 또는 지정 공휴일(2024-06-06, 2024-08-15)
 - baseline1의 경우, 전력 사용량이 백분위 하위 99%의 경우 의도적으로 휴일로 설정하였었지만, 이 경우 정보 손실 ↑ → 명확한 주말 및 공휴일만 휴일로 사용
- 폴드별 집단 통계 피처
 - `_build_time_stats(df_sub)`: 해당 폴드의 train 구간만으로 통계를 만듦
 - 건물별 시간대-요일에 대한 평균 값, 표준편차를 학습시켜 모델이 건물을 학습할 때 그 건물의 대표적인 전력 사용량을 알려주는 지표로 활용

03. 전처리 - 모델 1

- 건물별 전력사용량 시각화



- 시계열 데이터이므로 단순한 K-FOLD 대신 `TimeSeriesSplit` 사용, 검증구간과 훈련구간이 어떻게 구성되는지 확인하는 작업
- 일반적인 빨간구간이 각 fold의 val 구간으로, fold가 커질수록 학습데이터 양이 많아지고, val 구간이 예측해야할 test 구간과 인접해진다는 특징이 존재

03. 전처리 - 모델 1

- **test_size** 설정
- test_size를 기본값으로 설정시 검증길이가 510시간으로 고정되기 때문에 학습 구간이 너무 짧아짐

분할	학습 구간(포함)	검증 구간(포함)	학습 길이	검증 길이
Split 1	6/1 00:00 ~ 6/22 05:00	6/22 06:00 ~ 7/13 11:00	510h	510h
Split 2	6/1 00:00 ~ 7/13 11:00	7/13 12:00 ~ 8/03 17:00	1,020h	510h
Split 3	6/1 00:00 ~ 8/03 17:00	8/03 18:00 ~ 8/24 23:00	1,530h	510h

- test_size 의도적 설정(실제 예측 길이 : 2024.08.25 00:00 ~2024.08.31 23:00로 24*7와 동일하게 설정)
→ 학습 구간 월등히 증가

예시(`test_size=168, gap=24, n_splits=3`)의 시간 경계(정확):

분할	학습 구간(포함)	(gap)	검증 구간(포함)
Split 1	6/1 00:00 ~ 7/31 23:00	8/1 00:00 ~ 8/1 23:00 제외	8/2 00:00 ~ 8/8 23:00
Split 2	6/1 00:00 ~ 8/8 23:00	8/9 00:00 ~ 8/9 23:00 제외	8/10 00:00 ~ 8/16 23:00
Split 3	6/1 00:00 ~ 8/16 23:00	8/17 00:00 ~ 8/17 23:00 제외	8/18 00:00 ~ 8/24 23:00

03. 전처리 - 모델 1


- **n_splits**
- n_splits = 5로 진행, 보다 모델의 일반화된 성능을 확인해보고자 설정 → 다만, 3개의 split으로 진행할때 각 모델이 더욱 긴 훈련 데이터를 학습하므로 학습 안정성을 위해선 split3이 더 좋을 수도 있어 보임.



n_splits = 5



n_splits = 3



04. 결과 - 모델 1

04. 학습 1 - 건물개별 학습 XGB

- 커스텀 함수 정의
- `smape(y_true, y_pred)` : 각 OOF/CV 결과 집계 시 smape 지표 사용, 모델 예측값과 실제값의 smape 계산
- `smape_eval(y_pred, y_true)` : 훈련 중 운영 지표 모니터링
- `weighted_mse(alpha=3.0)` : 목적함수로 weighted mse 활용. smape를 목적함수로 놓는 경우 학습과정에서 계산이 어려울 수 있으므로, 대신 과소예측에 더 큰 비용을 주도록 mse 커스텀

```
#####  
# 최근성 가중 smape 함수 그대로 유지  
def recent_weighted_score(fold_smape_list, decay=0.5):  
    K = len(fold_smape_list)  
    weights = [decay ** (K - 1 - i) for i in range(K)]  
    weights = np.array(weights) / sum(weights)  
    weighted_mean = np.sum(weights * np.array(fold_smape_list))  
    weighted_std = np.sqrt(np.sum(weights * (np.array(fold_smape_list) - weighted_mean) ** 2))  
    lambda_penalty = 1.0  
    score = weighted_mean + lambda_penalty * weighted_std  
    return score, weighted_mean, weighted_std, weights
```

- 하이퍼 파라미터 평가 위한 성능 지표. 최근 fold의 예측값일수록 가중치를 더욱 높게 주어 smape의 가중합을 구하고, 또한 폴드별 OOF 값이 전부 학습에 들어가기 때문에 안정성을 위하여 폴드별 smape 값의 차이가 너무 큰 경우 패널티를 주기 위함

04. 학습 1 - 건물개별 학습 XGB

- 가중치

- $\tilde{\omega}_i = decay^{K-i}, \omega_i = \frac{\tilde{\omega}_i}{\sum_{j=1}^K \tilde{\omega}_j}, i = 1, \dots, K$
- 코드와 동일하게 $i = K$ 가 가장 최근 폴드, ω_K 가 가장 큼, $decay \in (0,1]$

- 최근가중 평균 sMAPE

- $\mu_{\omega} = \sum_{i=1}^K \omega_i s_i$

- 최종 선택 점수 (위험회피형)

- $J(\theta) = \mu_{\omega}(\theta) + \lambda \sigma_{\omega}(\theta), \theta = \text{하이퍼 파라미터 벡터}, \lambda(> 0) = \text{변동성(불안정성)에 대한 페널티 강도}$

- 목표

- $\theta^* = \operatorname{argmin}_{\theta} J(\theta)$

- Ex) decay=0.5인 경우 fold5의 최근성 가중치는 약 0.52, fold1의 가중치는 0.03.

→ 가중치로 활용하여 가중 평균 smape + 폴드 간 변동성이 작을수록 성능이 높은 하이퍼 파라미터로 선택

- 최근가중 표준편차 (모수형)

- $\sigma_{\omega} = \sqrt{\sum_{i=1}^K \omega_i (s_i - \mu_{\omega})^2}$

04. 학습 2 - 건물개별 학습 XGB

- 앙상블 - 소프트보팅
- 모델훈련으로 생긴 총 10개의 모델(건물개별 학습 5개, 건물유형별 학습 5개)에 test 데이터 입력, pred 값 생성, 위의 방식과 동일하게 최근성 반영 smape으로 가중치 부여

가중치 계산

```
fold_smape = cv_results[fold]['smape']  
smape_weight = 1.0 / (fold_smape + 1e-8)  
temporal_weight = (0.7) ** (4 - fold)  
final_weight = smape_weight * temporal_weight
```

- Ex) 개별 모델 5개 예측: [100, 110, 120, 105, 115], 유형 모델 5개 예측: [108, 112, 118, 107, 111]
- 가중치 계산 :
 - 개별 fold 0 → CV SMAPE 10.0, → 가중치 0.10
 - 개별 fold 4 → CV SMAPE 6.0, 최근 폴드라 가중치 0.25
 - 유형 fold 0 → CV SMAPE 9.0, 가중치 0.12
 - 유형 fold 4 → CV SMAPE 7.0, 최근 폴드라 가중치 0.20
 - 최종 예측 = $\sum (\text{가중치} \times \text{예측값}) = 0.10 \times 100 + 0.25 \times 115 + 0.12 \times 108 + 0.20 \times 111 + \dots \approx 111.3$ 예측값 계산

04. 실험 결과 및 결론

실험	결과
timeseriessplit 기본값	public 20.17
timeseriessplit test_size 수정	public 11.58
test_size 수정 + 앙상블	public 10.56
test_size 수정 + 하이퍼 파라미터 튜닝 후 앙상블	public 7.6

- 기존의 가장 큰 문제였던 LB상의 점수와 CV 상의 점수가 차이점에 대해 기존 K-FOLD 방식의 학습 및 검증의 경우 미래의 전력사용량 값을 보기 때문에 데이터 누수가 발생 가능함을 발견
- 시계열 데이터 처리 위한 학습 검증 방식을 통해 누수를 막을 수 있었지만, 학습 데이터가 적어진다는 문제로 인해 제대로 학습이 이루어지지 않아 성능이 매우 낮아졌음
- 주어진 데이터의 길이와 특성을 파악하여 test_size를 조절하고, 최근의 fold일수록 가중치를 더욱 부여하여 학습 및 예측을 진행할 때 성능이 크게 향상됨을 확인

05. 한계점 및 제언

- 스택킹과 같은 앙상블 학습과정에서 샘플 가중치 등을 고려하여 학습을 하려했으나, 이러한 점이 **weight 학습에 실제로 큰 영향을 끼치지 못하여 스택킹을 포기**했다는 점이 아쉬움
- PatchTsT 등의 deep 기반 시계열 모델을 시도해보았으나 성능이 크게 높지 않아 아쉬운 점이 있었고, 딥러닝 및 트랜스포머 기반 모델을 적용하기에는 훈련데이터의 기간이 너무 짧지 않았을까 추론해봄
- 데이터 전처리 과정에서 $n_splits = 3$ 인 상황도 진행해보고자 했으나 GPU 문제로 시도해보지 못함



03. 전처리 - 모델2

03. Baseline 접근 - XGBoost 단일 파이프라인

전처리

→ 제출 결과 11.7674256821

1. 데이터 정리

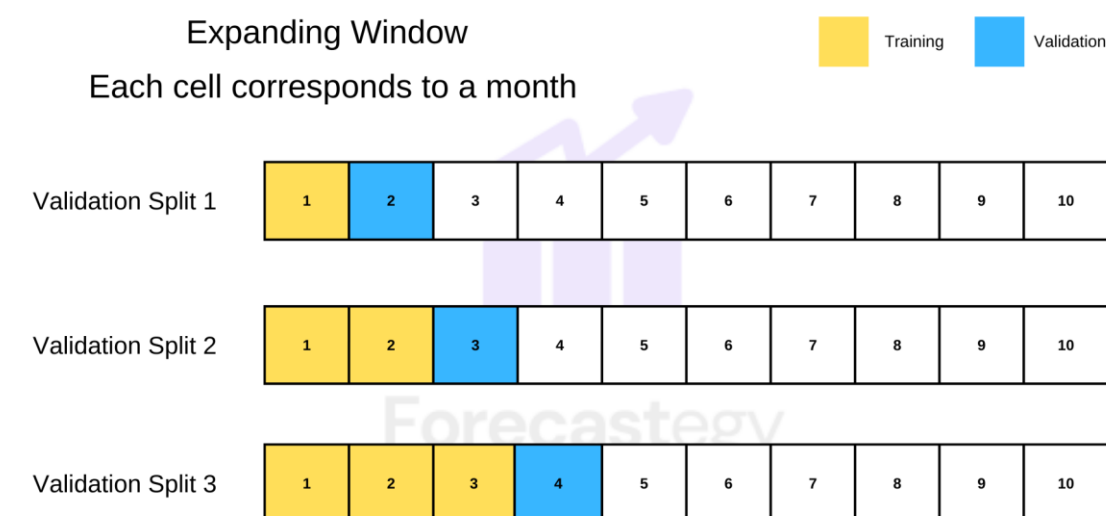
- "-" → NaN 치환, test에 없는 변수 제거
- datetime 변환 & 건물 특성 조인

2. 시간/주기성 변수

- hour, dow, is_weekend
- 여름 전용 위상(summer_cos) : $\text{summer_cos} = \cos((\text{month}-6)*\pi/3)$
→ 6-8월만 있어 연중 주기 대신 여름 위상을 쓰는 설계
- sin/cos 주기 인코딩

모델링 접근

- Target: log1p 변환 → expm1 복원
- CV: Expanding CV (시간 누출 최소화)



03. 문제점 (Baseline 한계)

1. 예측 안정성 부족
 - 폴드·빌딩 간 SMAPE 분산 큼
2. 피크 수요 과소예측
 - 냉방부하 급등 시간대(폭염, 주간 피크)에서 예측이 눌림
3. 저수요 구간 SMAPE 폭증
 - SMAPE는 분모($|y|+|\hat{y}|$)가 작을 때 민감.
 - 새벽·비영업 시 분모 작아 민감
4. OOF ↔ Test 괴리
 - 시점 경계 구간(말일→말일+1주 등)에서 성능 급락

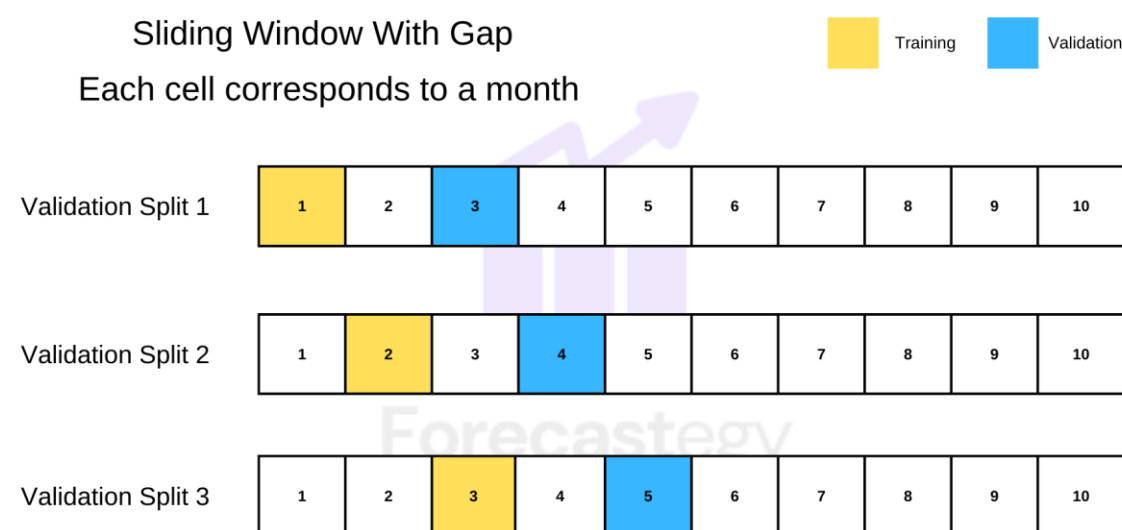
03. 문제점 (Baseline 한계)

원인분석

1. 로그 변환 복원 시 로그-정규 분포 편향
 - 평균 레벨이 낮게 복원 → 피크 억제
2. 기본 CV 설계의 낙관편향
 - 학습→검증 사이에 gap(완충 구간)이 없으면 직전 학습 구간의 잔존 자기상관 때문에 검증이 낙관적
 - 초기 폴드에서 학습 데이터가 짧아 분산 확대, 폴드 간 성능 편차 확대
3. 손실/지표 특성과 데이터 분포 미스매치
 - 학습은 MSE 계열로 최적화 (평균 레벨에 집중, 피크/꼬리 희생)
 - 반면 SMAPE는 저수요구간의 작은 오차도 크게 반영 → 지표와 학습 목표 간 불일치

03. 개선 방향

1. CV 안정화: Gap(embargo) 추가
2. 피쳐 강화: Lag(24h/168h), Rolling, 시간대별 교정
3. 예측 보정: OOF 기반 스케일 캘리브레이션(ex. 시간대별 예측/실측 비율로 보정)
4. SMAPE 민감도 대응: 저부하 구간 가중, 사후 리샘플링
5. 앙상블/보조모델: Seasonal-naive, ARIMA, 중앙값 앙상블
6. 빌딩별 맞춤 학습: 이질성 반영
7. 날씨 전처리의 피크 보존(보간 최소화/변화율 피쳐 병행)



03. 전처리 - 피처 엔지니어링 확장

1. 시간, 달력, 주기

- 공휴일/일요일 플래그(is_holiday)
- 시간대 평균/표준편차(by hour, by dow) 활용 (예측 후 보정에 사용)

2. 날씨/체감 지수


- 불쾌지수(THI), 체감온도, 온습도지수 등 합성 피처
- temp_diff, hum_diff 등 차분/변화량
- 결측 → 선형보간(interpolate) / test에 없는 변수(sunshine_hour, solar_radiation)는 제외

3. 시간 의존성(누출 방지)

- 모든 시계열 특징에 shift(1) 적용 : 하루 전 값
- Lag : (lag_1, lag_24, lag_168) / 롤링(shifted) mean, std 활용
- 168시간(1주) 주기성을 고려 → seasonal-naive와도 연결

4. 정적(건물) 특성

- 건물 유형 One-Hot, 면적 로그 변환
- 건물별 개별 모델링 전략 - 특수 건물(병원 등) 분기 실험 후 결정



04. 결과 - 모델 2

04. 모델 실험

1) TabPFN 시도

- 소표본 일반화 기대했으나 → 오류/시간 비용/일관성 문제.
- GPU·RAM 부담 크고 성능도 XGB 대비 미흡 → 최종 탈락.

→ 제출 결과 12.8547602702

2) 시계열 전용 모형 (Prophet/ARIMA)

- Prophet: 성능 부진(SMAPE 20 수준) → 제외.
- ARIMA/Seasonal naive: 보조 예측 생성해 XGB/TabPFN과 앙상블 or 스택킹 입력에 사용.
- 급변/계절성 보완은 조금 효과 있었으나 큰 개선 없음.

→ 제출 결과 20.06070902

3) 앙상블·스태킹·보정

- Seed × Estimator 중앙값 앙상블.
- OOF 기반 스케일 보정: 예측/실측 스케일 차이를 추정해 보정 계수 적용.
- 시간대 편향 보정: 시간(hour)별 평균 오차로 잔차 보정

-> XGBoost 중심 체제로 정착.

→ 제출 결과(기본 XGB) 8.17180269

04. 모델링 전략

1. 교차검증 / 데이터 분할 진화

- TimeSeriesSplit → Expanding CV → Gap Split → StartFrac 조정 → 빌딩 단위 분리
→ 초기구간 OOF 마스킹
- 단계별 발전 : 정보 누출 최소화 + 일관된 SMAPE 검증 체계 구축함

2. 예측 파이프라인 최적화

- 건물단위 캐시저장, 피처 선택, 앙상블 규모 축소, 롤링 윈도우 제한으로 성능/시간 최적화
- 예측 후 처리 : 클리핑 (음수 방지 + 빌딩별 상한 적용), 시즌널 naïve 안전망, median 앙상블

04. 하이퍼파라미터 - XGBoost

- `n_estimators` 600-1600 (조기종료 대신 폴드 고정)
- `learning_rate` 0.03-0.10
- `max_depth` 4-8
- `min_child_weight` 1-6
- `subsample` 0.6-0.9
- `colsample_bytree` 0.6-0.9
- `reg_alpha` 0-1e-2, `reg_lambda` 1-10
- `max_bin` 256(대체로 기본), `tree_method` 는 GPU 환경에 맞춰 설정
- 시간순 CV에서 SMAPE로 비교, 빌딩별로 소폭 조정만 허용(일관성 유지가 먼저).
- 버전 의존 인자 제거(`eval_metric`, `early_stopping_rounds`, `callbacks`) → 모든 제어는 외부 루프에서.
- XGBoost 튜닝: `learning_rate=0.06`, `n_estimators=1800`, `max_depth=7`, `min_child_weight=6`, `subsample/colsample=0.85`, `reg_alpha=2`, `reg_lambda=8`

04. 최종 파이프라인

1. 데이터 & 피처

- 파생: thi, apparent_temp, temp_diff, hum_diff
- 주기/자기상관: sin/cos_hour, sin/cos_dow
- 락·롤링 적용(lag {1,24,168}, rolling(24,168)) → 누출 방지

2. 모델링

- XGBoost (주력, log1p → expm1 + 소프트 클리핑)
- 보조: ARIMA, Seasonal-naïve
- 빌딩별 학습, Seed Ensemble 3개
- CV: Expanding window + Gap(24h)

3. 앙상블 & 후처리

- 중앙값 합성(XGB + ARIMA/naive)
- OOF 스케일 캘리브레이션 + 시간대 잔차 보정
- Seasonal-naive 블렌드 10%
- 음수 제거 + 소프트 클리핑

→ 최종 결과 : SMAPE 6.9

04. 결론 및 제언

주요 성능 개선 요소(체감순)

1. shift(1) 강제 + Gap CV로 누출 제거
2. test에 없는 변수 완전 제거(일조/일사) → 스키마 안정화
3. 차분·체감지수 추가(THI, temp/wind/hum/precip_diff)
4. 168h 주기 대비 랙/롤링(24/168)
5. OOF 스케일/시간대 보정
6. seasonal naive 백업 + 중앙값 앙상블
7. 건물 단위 캐시로 반복 실험 가속

개선 아이디어

- 이상치 탐지 + 보정 (비정상 피크 처리)
- 딥러닝 시도 (LSTM, Temporal Fusion Transformer)
- 날씨 예측 오차 보정 (외부 기상 데이터 활용)



Thank You