전력소비량예측모델

고급 머신러닝 파이프라인

100개 건물의 시간대별 전력소비량 예측을 위한 통합적인 분석 솔루션
데이터 전처리, 피처 엔지니어링, XGBoost/LightGBM 앙상블모델링,
앙상블 기법을 활용한 머신러닝 자동화 파이프라인

프로젝트 개요

목표

100개 건물의 시간대별 전력소비량 예측 모델 개발

다양한 건물 유형과 설비에 일반화 가능한 솔루션 구축

주요 데이터

- 건물정보: 100개 건물의 유형, 면적, 태양광/ESS/PCS 설비
- 기상정보: 기온, 습도, 풍속, 강수량, 일조, 일사
- 전력소비량: 타깃 피쳐

평가지표

SMAPE

(Symmetric Mean Absolute Percentage Error)

$$SMAPE = 100 \times \frac{2}{n} \times \sum \left(\frac{|\text{예측값} - \text{실제값}|}{(|\text{예측값}| + |\text{실제 값}| + \varepsilon)} \right)$$

낮을수록 좋은 지표로,
예측값과 실제값의 절대적 차이를 백분율로 표현

접근 방법

- 고급 피처 엔지니어링 및 데이터 증강
- XGBoost/LightGBM 앙상블 모델
- Optuna 기반 자동 하이퍼파라미터 최적화
- 세그먼트/유형/건물별 앙상블 전략

전체 워크플로우

전력소비량 예측을 위한 7단계 파이프라인 구조

1
환경설정 및 시드
재현성 보장을 위한
시드 고정과 환경 준 비

2
데이터 로드 및 병합
train, test, building_info
데이터 통합

3
피처 엔지니어링
시간 기반, 태양 기하학,
건물 특성 피처 생성

4
이상치/결측치 처리
시각적 분석 기반
이상치 제거 및 결측치 보간

5
통계 및 증강 피처
계층적 백오프 기반
통계 피처와 15분 단위 데이터 증강

6
모델링 및 최적화
XGBoost/LightGBM 하이브리드 모델과
Optuna 최적화

7
앙상블 및 제출
세그먼트유형/건물별 예측의
가중 앙상블과 최종 결과 생성

데이터 구조 및 전처리

데이터셋 구성

- train.csv: 시간별 전력소비량 학습 데이터 (6-8월)
- test.csv: 예측 대상 기간 데이터 (일부 컬럼 제외)
- building_info.csv: 100개 건물 정보 (유형, 면적, 설비)
- sample_submission.csv: 제출 양식

초기 전처리 단계

- 한글 컬럼명 → 영문 컬럼명 변환
- 결측치 처리: pvc, ess, pcs 등 0으로 대체
- 이상치 처리: 풍속/습도 0값 → 선형 보간
- 데이터 타입 최적화 및 메모리 효율화
- train, test와 building_info 병합 (left join)

컬럼 매핑 및 병합

원본 컬럼 명	매핑 컬럼 명
건물번호	building_num
기온(℃)	temperature
습도(%)	humidity
전력소비량(kWh)	power_consumption
태양광용량(kW)	pvc

전처리 흐름도

피처 엔지니어링(1)
시간 기반 피처

날짜/시간 파생 변수

- 기본 추출: 일시 문자열에서 date 변환
- 시간 성분: minute, hour, day, month
- 주간 성분: dow (요일, 0-6), day_of_year (연중일)
- 휴일 처리: holidays 변수 (0: 평일, 1: 휴일, 2: 특별일)

SIN/COS 주기성 인코딩

주기적인 시간 변수를 SIN/COS 변환으로 연속적인 특성으로 인코딩

풍속/습도 0값 처리

물리적으로 풍속과 습도는 실제로 0이 될 수 없음 → 결측치로 간주하고 처리

- 0값을 np.nan으로 변환
- 건물별로 그룹핑하여 처리 (groupby('building_num'))
- 선형 보간(interpolate)으로 결측치 채우기
- 남은 결측치는 앞뒤 값으로 채우기 (ffill, bfill)

복합 시간 기반 피처

피처명	설명
SIN_Time, COS_Time	하루 전체 분 단위 주기성 (1440분)
SIN_minute, COS_minute	시간 내 분 단위 주기성 (60분)
SIN_day, COS_day	월 내 일 단위 주기성 (월별 일수 고려)
SIN_month, COS_month	연 내 월 단위 주기성 (12개월)
SIN_summer, COS_summer	여름 계절 내 주기성 (6월-9월)
SIN/COS_day_of_year	연중 일 주기성 (365일)

피처 엔지니어링(2)
태양관련피처

태양 관련 피처

- 일출/일몰 시각 계산: 위도, 경도, 날짜 기반 정밀 계산
- 태양 고도각: 주간/야간 구분 및 태양광 발전 효율 반영
- 일조/일사량 연계: PV 발전량과 전력소비 패턴 상관관계 분석

물리 기반 파생 피처

- 체감온도(PT): 온도, 습도, 풍속 조합 계산 냉방도일
- 냉방도일(CDH): max(temperature - 26, 0)
- 불쾌지수(DI): 0.81T + 0.01H(0.99T-14.3) + 46.3
- 설비 밀도: (ess + pcs) / (all_area + ε)

건물 그룹핑 전략

동일 지역/유사 운영 특성을 갖는 건물들을 30개 그룹으로 군집화하여 기상조건 공유 및 운영패턴 유사성 활용

건물 그룹 예시:

Group 1: [28] Group 2: [72] Group 3: [19,58,75,91]

Group 4: [77] Group 5: [24] Group 6: [61,74,81]

Group 7: [32,42,65,79,99]

같은 일시에 동일한 날씨를 갖는 건물들을 그룹화 (기상조건 패턴 기반)

주간/야간 구분 및 계절성

- 주간/야간 플래그: 시간 경계 전후 기준 3개 지표
  - daylight_prev: [t-1, t) 구간이 주간과 겹치면 1
  - daylight_instant: 시각 t가 주간이면 1
  - daylight_next: [t, t+1) 구간이 주간과 겹치면 1
- SIN/COS 변환: 여름 피크 및 계절 주기성 반영

```
season_start = pd.to_datetime(yr + "-06-01")
season_end = pd.to_datetime(yr + "-09-01")
pos = (df['date'] - season_start).dt.total_seconds() / period_sec
phi = 2 * np.pi * pos
df['SIN_summer'] = np.where(in_season, np.sin(phi), np.nan)
```

통계 기반 피처 생성

계층적 백오프 전략 개요

1. **가장 세분화된 통계**
   (건물번호, 시간, 요일) 기준 평균/표준편차
2. **휴일 기반 통계**
   (건물번호, 시간, 휴일여부) 기준 평균/표준편차
3. **시간별 통계**
   (건물번호, 시간) 기준 평균/표준편차
4. **건물 전체 통계**
   (건물번호) 기준 평균/표준편차
5. **글로벌 통계 (최후 백오프)**
   전체 데이터셋 평균/표준편차

주요 그룹 통계 피처

- dow_hour_mean/std: 건물-시간-요일별 통계
- holiday_mean/std: 건물-시간-휴일별 통계
- hour_mean/std: 건물-시간별 통계
- building_mean/std: 건물별 전체 통계
- global_mean/std: 전체 데이터 통계

* 모든 통계는 train 데이터에서만 계산하여 test에 적용

계층적 백오프 구현 코드

```
df['dow_hour_mean'] =
        (df['dow_hour_mean
        '])
            .fillna(df['holiday_mean'])
            .fillna(df['hour_mean'])
            .fillna(df['building_mean'])
)           .fillna(global_mean)
```

백오프 전략의 장점:

- 결측치와 드문 패턴에 대한 강건한 추정
- 도메인 지식을 반영한 체계적 대체 방법
- 세부 그룹이 없을 때도 의미있는 값 유지
- SMAPE 오차 최소화에 효과적

데이터 증강 및 이상치 처리

시계열 데이터 증강

- 목적: 시간 해상도 향상, 패턴 감지 향상, 더 세밀한 예측
- 방법: 15분 단위 선형 보간(Linear Interpolation)
- 기능: 1시간 간격 데이터를 15분 단위로 확장(4배 증강)
- 구현: upsample_20min_linear() 함수 사용

이상치 탐지 및 처리

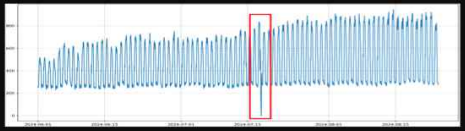
- 이상치 유형: 구간 이상치(intervals), 단일 시점 이상치(singles)
- 탐지 방법: 시각적 분석 + 통계 기반 이상 감지
- 처리 방법: 제거 후 통계/예측 기반 대체

```
intervals = { 5: [(2024080407, 2024080408)],
              28: [(2024071714, 2024071715), (2024060906, 2024060920)],
              # 특정 건물별 이상치 구간 직접 지정 }

singles = { 3: [2024071714],
            # 단일 시점 이상치 # 건물별 특정 시점 이상치 목록 }
```

시각적 이상치 분석

전통적 이상치 탐지(IQR, Z-Score)로 감지되지 않는 패턴 이상을 시각적 분석으로 식별



결측치 예측 보간

원본 결측을 머신러닝으로 보간

XGBoost + LightGBM 앙상블
TimeSeriesSplit 기반 교차검증으로 과적합 방지

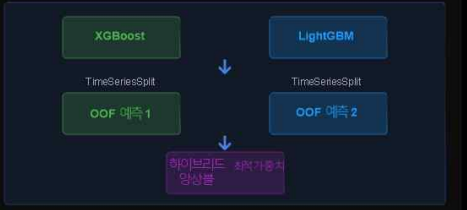
일사량(solar) 결측 보간
일사량이 0으로 잘못 기록된 건물의 주간 구간 예측
impute_solar_for_zero_bnos_cv_optuna()

태스트셋 일사량/일조시간 예측
일조/일사 둘 태스트 구간 예측으로 특성 보강
train_predict_test_target_cv_optuna()

이상치 처리와 결측 보간이 SMAPE 성능 약 **12%** 향상

모델 앙상블 전략

앙상블 모델 구조



핵심 모델링 기법

- 시계열 분할 검증: TimeSeriesSplit으로 미래 데이터 누수 방지
- 로그 변환: log1p() 변환으로 타깃 분포 개선 (back: exp1m)
- 최적 파라미터: Optuna로 모델별 최적 파라미터 자동 탐색
- 최적 반복: 모델별 best_iteration 활용 (과적합 방지)
- OOF 검증: 전체 파이프라인의 성능 검증 활용

3가지 모델링 전략

세그먼트별 모델

건물을 설비 기준으로 세그먼트화:

- ESS 보유 건물
- PVC 보유 건물
- 일반 건물

각 세그먼트에 특화된 피처셋 하이퍼파라미터 적용

유형별 모델

건물 유형별 분류:

- 공공
- 학교
- 데이터센터
- 아파트
- etc

유형별 소비 패턴 특성을 반영한 모델 학습

건물별 모델

개별 건물 특화:

- 100개 건물별 특화
- 건물별 휴일 패턴
- 건물별 이상치 처리

특이 패턴 건물에 효과적인 개별화 접근

모델 코드 구현

```
def _train_one_model_cv_optuna(X, y, X_test, bnum, params=None):
        #TimeSeriesSplit으로 시계열 특성 보존
        tscv = TimeSeriesSplit(n_splits=N_SPLITS_MODEL)

        #XGBoost & LightGBM 하이브리드 모델 학습
        xgb_oof = np.zeros(len(X))
        lgb_oof = np.zeros(len(X))

        #최적 반복 횟수 추적
        xgb_iterations = []
        lgb_iterations = []

        #각 Fold별 학습 및 예측
        for i, (train_idx, val_idx) in enumerate(tscv.split(X)):
                # XGBoost 학습 ...
                # LightGBM 학습 ...
                # OOF 예측 저장 ...
```

하이퍼파라미터 튜닝(최적화) - optuna

Optuna 베이지안 최적화

1. 목적함수 정의: 최소화할 SMAPE 평가지표
2. 탐색 공간 설정: XGBoost/LightGBM 파라미터 동시 최적화
3. TPE(Tree-structured Parzen Estimator) 샘플러 적용
4. MedianPruner로 유망하지 않은 트라이얼 조기 종료
5. N_TRIALS_MODEL(25회) 반복 실행 후 최적 파라미터 선택 및 저장

Optuna 구현 예시

```
# 목적함수 정의
def objective(trial):
    params_xgb = {
        'learning_rate': trial.suggest_float('xgb_lr', 0.01, 0.
        'max_depth': trial.suggest_int('xgb_depth', 3, 10),
        'subsample': trial.suggest_float('xgb_ss', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('xgb_cs', 0.6,
        'min_child_weight': trial.suggest_int('xgb_mcw', 1, 10)
```

MD5 해시 기반 캐시 시스템

- 피처셋 해상: 피처 구성이 바뀔 때만 새로운 최적화 실행
- 결과 캐싱: JSON으로 최적 파라미터 저장 및 재사용
- 재현성 보장: 모든 단계에서 동일한 시드 유지
- 실험 추적: 피처/파라미터 구성별 성능 비교 가능

```
def get_features_hash(features):
    features_str = ','.join(sorted(features))
    return hashlib.md5(features_str.encode()).hexdigest()[:10]

cache_key = f"{bnum}_{get_features_hash(features)}_{SEED}"
```

성능 평가 및 결과

OOF 검증 결과 (SMAPE)

13.8%	14.9%	7.7%
세그먼트 모델 OOF SMAPE	유형별 모델 OOF SMAPE	건물별 모델 OOF SMAPE

* OOF: Out-of-Fold 검증 - TimeSeriesSplit을 통한 시계열 검증

최종 결과

10.97%	5.15%	5.52%
최종 앙상블 LOCAL SMAPE	8th public score	6th private score

- 세 가지 모델(세그먼트/유형/건물별) 동등 가중치 앙상블
- 특정 건물 유형별 성능 차이 발견 (상업시설 > 교육시설)
- 휴일/영업일 구분 모델링으로 패턴 예측 향상

요약

재현성 보장

- 다중 시드 고정: PYTHONHASHSEED, random, numpy 시드 일관성
- 버전 태깅: 각 실험에 고유 버전 코드 부여 (`version = "LETS_GO"`)
- 알고리즘: 모든 랜덤 과정에 seed 파라미터 적용
- 분리된 경로: 실험별 독립 저장 경로로 덮어쓰기 방지

```
SEED = 775
os.environ["PYTHONHASHSEED"] = str(SEED)
random.seed(SEED)
np.random.seed(SEED)
save_path = f"./Energy/03/{SEED}_submission_{version}/"
```

캐시 및 결과 관리

- MD5 해시 기반 캐시: 피처셋 + 파라미터 조합의 해시값으로 캐시 키 생성
- 구간별 중간 저장: 전처리/모델링 각 단계 결과 저장
- 실험 로깅: 주요 결정 사항과 성능 지표 자동 기록
- 다양한 평가 지표: SMAPE, OOF, 예측 분포 검증

캐시 시스템 성능

재탐색 없이 즉시 로드 ~80% 이상 속도 향상

확장성 및 향후 개선 방향

- 모델 확장성: 세그먼트/유형/건물별 특화 모델 추가 용이
- 피처 플러그인: 새로운 피처 생성 함수 즉시 통합 가능
- 멀티프로세싱: 병렬화 위한 프레임워크 준비됨
- 추가 개선 가능 영역:
  1) 메타 피처 자동 생성 시스템
  2) 클라우드 기반 분산 학습 지원
  3) 실시간 예측 API 구현

실제 현장 적용 가능성

건물 에너지 모니터링 시스템

☆ ☆ ☆ ☆ ☆

재현성 + 확장성 + 적응성

후기및 소감

모델 구성 전략 및 효과

1. "보조 타깃 생성 → 메인 타깃 예측 → 앙상블/보정"의 3단계로 고정
2. 세분화 + 시계열 초점형 피처 + 가벼운 앙상블이 가장 일관되게 점수를 끌어올림
3. optuna sampling으로 모델 실행 시간을 획기적으로 줄였음

모델 고도화의 한계점

무차별 잔차 보정 : 코드의 복잡도 ↑검증 누수는 성능보다 변동성을 키웠음

코드의 복잡도를 개선할 필요가 있고, 자원의 한계로 인해 더 많은 시도를 해보지 못한 점은 아쉬움으로 다가옴

소감

1. 이번 대회의 교훈은 피처 엔지니어링의 중요성을 단적으로 확인했다는 점에 있습니다. 일조시간과 일조량을 예측하는 모델을 시간의 한계로 더욱 정교화 및 고도화하지 못했다는 아쉬움이 깊이 남았습니다.
2. 모델을 1) 세그먼트별, 2)건물유형별, 3)건물번호별로 나눠서 학습 및 예측 앙상블 하였음에도 완벽한 모델을 만들지 못했다는 점은 저 자신의 공부량 부족이 원인임을 느꼈습니다.
3. 이 대회 기간 내내 나는 너무나도 즐거웠습니다. 수상을 하지 못하더라도 매일매일 고도화 전략을 고민하는 시간을 보냈고, 다른 사람들과도 이와 관련하여 이야기 하면서 또 다른 통찰을 얻을 수 있었습니다. 특히 optuna 샘플링 기법은 이와 관련하여 AI를 공부하지 않은 사람에게서 얻게 된 통찰이었습니다. 항상 옵튜나는 오랜시간이 걸린다는 문제점이 있었는데, 샘플링 데이터가 전체 데이터에 대한 대표성을 보일 수 있다면, 일부 샘플링만으로도 충분한 학습효과를 보이며 시간을 아낄 수 있다는 점을 확인하였습니다.

子曰 : 三人行, 必有我師焉. 擇其善者而從之, 其不善者而改之.
세 사람이 함께 길을 가면 거기에는 반드시 나의 스승이 있다.

감사합니다.

코드와 관련된 질문은 댓글로 써주시면 최대한 답변드리겠습니다.

[AI를 공부한지 얼마되지 않아 올바른 답변을 드리지 못하더라도 너그러이 양해바랍니다.]

새로운 질문과 통찰, 그리고 토론은 언제나 환영합니다.