GIST 창의 융합 경진대회

Github Repository: https://github.com/dalcw/GIST-Innovative-Convergence-Technology-Contest-2024-

캠퍼스 EMS 개발 콘테스트

팀명

- AICONIC

참가자 정보

- 나유경 - 전남대학교, 인공지능학부
- 문성수 - 전남대학교, 인공지능학부

제출파일 (PATH 정보)

```
C:.
├─Challenge01
│  │  2022_train.csv: 2022 학습 데이터 셋 (기상 정보 + 건물별 전력 사용량)
│  │  2023_train.csv: 2023 학습 데이터 셋 (기상 정보 + 건물별 전력 사용량)
│  │  data_preprocessing_2022.py: excel 파일을 csv로 정리하는 코드
│  │  data_preprocessing_2023.py: excel 파일을 csv로 정리하는 코드
│  │  deep_learning_modeling.py: 딥러닝 모델 (CNN-LSTM)
│  │  elec_inference_result.csv: 2023.08.31.에 대한 전력 사용량 추론 결과
│  │  elec_max.pickle: min-max 정규화를 위한 건물별 전력 사용량 최댓값
│  │  elec_min.pickle: min-max 정규화를 위한 건물별 전력 사용량 최솟값
│  │  ensemble.py: XGBoost + CNN-LSTM
│  │  inference.csv: 추론을 위한 입력 데이터
│  │  inference_dataset.py: inference.csv를 생성
│  │  machine_learning_modeling.py: 머신러닝 모델 (XGBoost)
│  │
│  └─model_result
│      ├─cnnlstm: CNN-LSTM 학습 weight 저장
│      └─machinelearning: XGBoost 모델 저장
│
├─Challenge02
│  │  2022_train.csv: 2022 학습 데이터 셋 (기상 정보 + 태양광 발전량)
│  │  2022_weather.csv: 2022 날씨 정보
│  │  2023_train.csv: 2023 학습 데이터 셋 (기상 정보 + 태양광 발전량)
│  │  2023_weather.csv: 2023 날씨 정보
│  │  data_preprocessing_2022.py: excel 파일을 csv로 정리하는 코드
│  │  data_preprocessing_2023.py: excel 파일을 csv로 정리하는 코드
│  │  deep_learning_modeling.py: 딥러닝 모델 (CNN-LSTM)
│  │  ensemble.py: XGBoost + CNN-LSTM
│  │  gen_max.pickle: min-max 정규화를 위한 발전량 최댓값
│  │  gen_min.pickle: min-max 정규화를 위한 발전량 최솟값
│  │  inference.csv: 추론을 위한 입력 데이터 셋
│  │  inference_dataset.py: inference.csv 생성
│  │  machine_learning_modeling.py: 머신러닝 모델 (XGBoost)
│  │  solargrid_inference_result.csv: 2023.08.31.에 대한 전력 생성량 추론 결과
│  │
│  └─model_result
│      ├─cnnlstm: CNN-LSTM 학습 weight 저장
│      └─machinelearning: XGBoost 모델 저장
```

```
│
└─Challenge03
        elec_inference_result.csv: 2023.08.31.에 대한 전력 사용량 추론 결과
        optimization.py: 2023.08.31. 일시의 예측 결과를 가지고 전기 요금 최적화
        solargrid_inference_result.csv: 2023.08.31.에 대한 전력 생성량 추론 결과
        extract_valid_building.py: 추론한 결과(건물별 부하량, 패널별 발전량)에서 총합을 계산
```

- **예측 결과물**
  - **건물 별 부하 데이터: Challenge03/elec_inference_result.csv**
    ⇒ 지정해주신 건물들에서의 부하의 총량은 Valid_sum_of_value Column에 위치합니다.
  - **태양광 발전량 데이터: Challenge03/solargrid_inference_result.csv**
    ⇒ 시간당 태양광 발전의 총량은 Total Column에 위치합니다.

제공된 데이터 셋에는 예측 날짜인 2023.08.31.일에 해당하는 전력 부하 및 태양광 에너지 생산량이 존재한다. 그러나 해당 정보들은 학습에 사용하면 안된다. (model cheating)

프로젝트 개요

본 프로젝트의 목적은 캠퍼스 내에서의 전기 에너지 사용량 및 태양광 패널을 통한 에너지 생산량을 예측하는 모델을 설계하고, 이를 바탕으로 전력 계통을 최적화하여 전기 요금을 최소화하는 것이다. 이를 달성하기 위해 프로젝트는 세 가지 주요 도전 과제로 구성된다.

도전 과제

[Challenge 1] 부하 예측 알고리즘 개발 및 검증

캠퍼스 내 전력 부하를 정확히 예측하기 위해, 본 프로젝트에서는 딥러닝 알고리즘인 CNN-LSTM과 머신러닝 알고리즘인 XGBoost를 앙상블하여 사용하였다. CNN-LSTM은 1D Convolution Layer와 LSTM을 결합한 구조로, 시계열 데이터를 효과적으로 처리하는 데 유리하다. 이 모델은 시계열 데이터의 패턴을 학습하여 장기적인 추세와 변동성을 예측하는 데 강점을 가진다. 반면, XGBoost는 특정 시점의 특징을 기반으로 예측을 수행하며, 다양한 특징 간의 인과 관계를 바탕으로 예측을 수행한다. 두 모델의 앙상블을 통해 시계열적 성질과 특정 시점의 특징을 모두 고려하여 보다 정밀한 예측 모델을 구축하였다.

[Challenge 2] 태양광 발전량 예측 알고리즘 개발 및 검증

태양광 발전량 예측 또한 전력 부하 예측과 유사한 접근 방식을 사용하였다. CNN-LSTM을 활용하여 시계열 데이터의 패턴을 학습함으로써 장기적인 발전량 변화를 예측하며, XGBoost를 통해 특정 시점의 특징들의 인과관계를 고려하여 예측의 정확성을 높였다. 이러한 두 모델의 결합을 통해, 보다 정밀한 태양광 발전량 예측 모델을 설계하였다.

[Challenge 3] 전기 요금 최소화 알고리즘 개발 및 검증

최적화 문제는 예측된 전력 부하 및 태양광 발전량을 활용하여 전기 요금을 최소화하는 것을 목표로 한다. 이를 위해 본 프로젝트에서는 컨벡스 최적화 알고리즘을 사용하였다. 컨벡스 최적화는 목적 함수(Objective function)과 제약조건(Constraint)으로 구성되며, 제약조건을 만족하면서 목적 함수를 최소화 하는 최적의 해를 도출하는 과정이다. 이 알고리즘은 전력 부하와 태양광 방전량을 반영하여 전기 요금을 효과적으로 줄일 수 있는 해법을 제공한다.

개발 환경 및 사용된 라이브러리

- 개발 환경은 Windows 11에서 수행되었다.
- 파이썬 프로그래밍 언어를 이용하여 개발되었으며, 사용된 라이브러리는 다음과 같다.

```
# requirements.txt
numpy
pandas
holidays
```

```
matplotlib
torch
tqdm
xgboost
sklearn
cvxpy
```

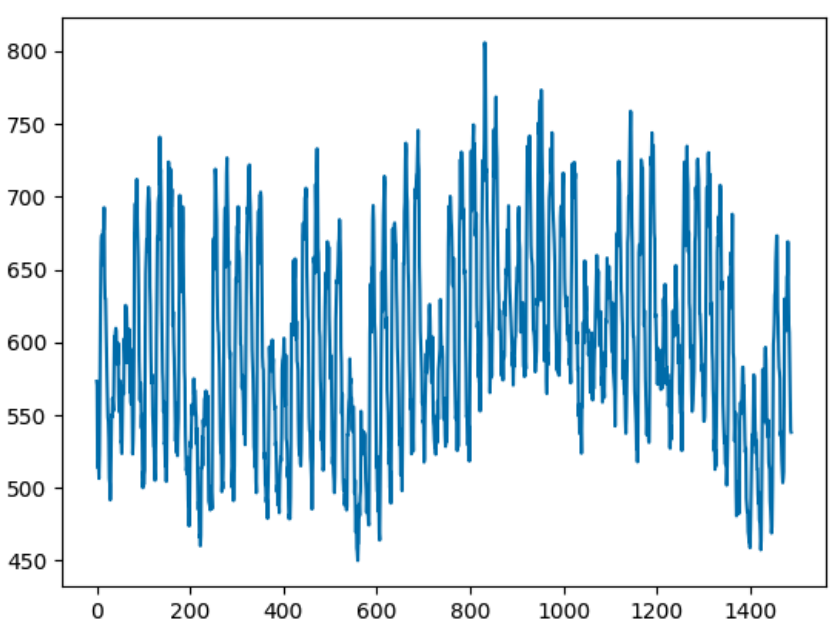
Challenge 1 - 부하 예측 알고리즘 개발 및 검증

디렉토리: Challenge01

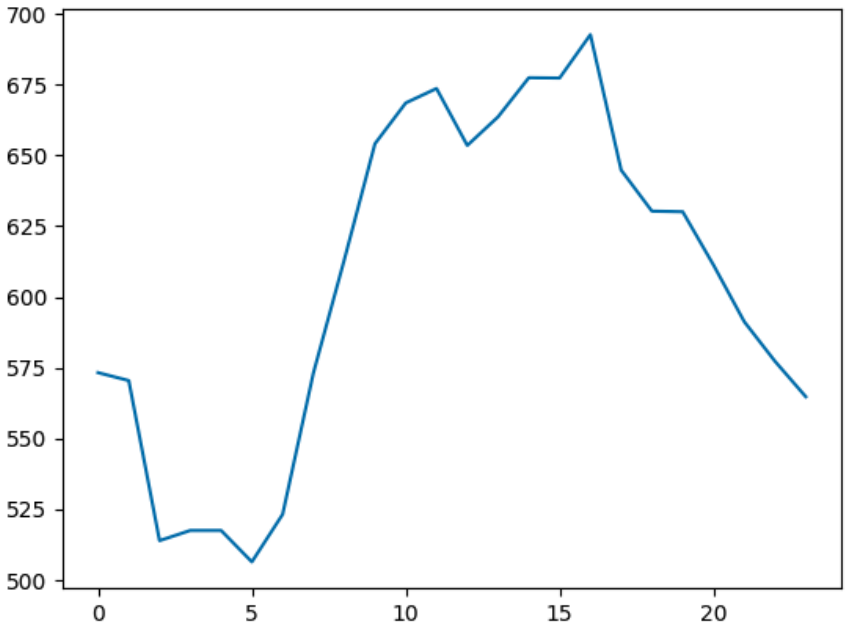
데이터 셋

- "***전력 일보***" 데이터에는 2022, 2023년도 7월 1일~8월 31일까지의 데이터가 제공된다. 해당 데이터에는 각 건물 별 **전류 R, S, T, 유효 전력, 유효 전력량**에 대한 정보가 기록되어 있다.
- 본 프로젝트에서는 **유효 전력**에 대한 데이터만 사용한다. 데이터 수집 주기가 1시간 단위이기 때문에 유효 전력의 값 자체가 전력 사용량으로 취급된다. (kw * 1h = kwh)
  (해당 데이터에 존재하는 유효 전력량은 특정 시점에서부터 누적된 전력량이므로 해당 특징을 사용하는 것은 적절하지 않다.)
- 모델이 추론(Inference)을 수행하기 위해서는 입력 값이 필요하다. 일반적으로 전력 시스템에서 부하 예측 알고리즘을 개발할 때는 주로 기상 데이터를 입력으로 사용한다. 예를 들면, CU-BEMS와 같은 데이터셋에서는 측정 시점의 기상 정보가 함께 제공된다. 따라서 본 프로젝트에서도 각 시점의 기상 정보를 메타데이터로 추가하였다. 기상 정보는 "기상청 - 기상자료개방포털"에서 다운로드할 수 있다.
- 뿐만 아니라, "휴일"이라는 특징을 정의해 줌으로써 모델이 휴일에 대한 전력 패턴을 학습할 수 있도록 하였다. (휴일인 경우 1, 휴일이 아닌 경우 0)
- 최종적으로 모델의 입력으로 "기온", "강수량", "풍속", "풍향", "습도", "현지기압", "일조", "일사", "지면온도", "휴일정보"가 사용되었으며, 모델의 출력으로는 건물에 대한 전력 사용량을 예측한다. 이때, 모델은 건물 별로 생성(학습)된다.
- 본 프로젝트에서는 22년도 관측 결과와 23년도 관측 결과가 주어졌다. 그러나, 23년도 데이터의 경우 결측치가 상당수 존재하여 학습에 사용하기에는 부적절하였다. 따라서, 학습 목적으로는 22년도 데이터만을 사용하였다.

데이터 시각화



다산 빌딩에 대한 장기 전력 사용량
(x축: 시간, y축: 전력 사용량 (kwh)



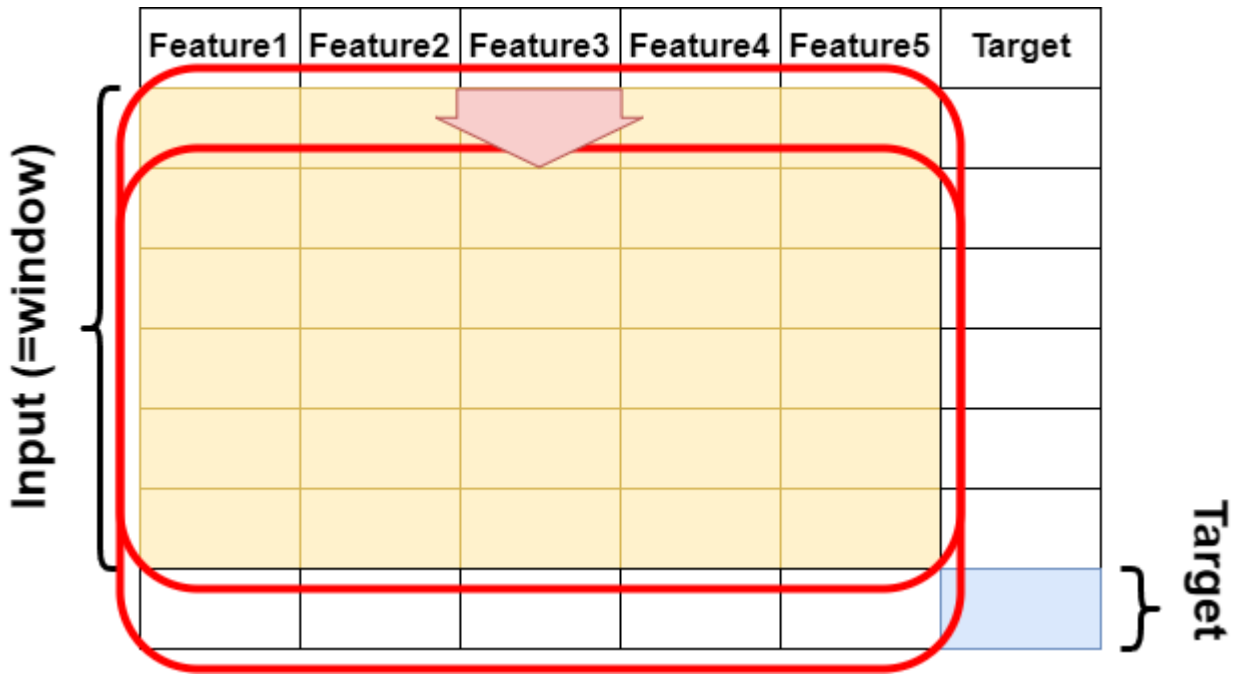
다산 빌딩에 대한 단기 전력 사용량
(x축: 시간, y축: 전력 사용량 (kwh)

- 위 그림은 **"다산 빌딩"**에 대한 장/단기 전력 사용량을 시각화한 결과이다.
- 오른쪽 그림을 보았을 때, 이른 오전에는 낮은 전력 사용량을 보이고, 주간에 높은 전력 사용량을 보이는 패턴을 확인할 수 있다. 이는 데이터가 시계열성을 띈다는 의미이며, 시계열 처리에 특화된 모델을 사용하기에 적합하다.

모델링: CNN-LSTM, XGBOOST

- 모델은 시계열 데이터 처리에 탁월한 CNN-LSTM 딥러닝 모델과 특징 정보를 기반으로 값을 추정하는 XGBOOST 머신러닝 모델을 앙상블하여 사용하였다.
- 많은 사람들이 시계열 데이터 처리에 특화된 딥러닝 모델(e.g., RNN, LSTM, GRU, Transformer 등)이 일반적으로 좋은 성능을 보일것일다고 생각하는 경향이 있다. 물론 틀린 말은 아니지만, 본인의 경험과 경진대회 우수 모델들을 리뷰하였을때 항상 그런것만은 아니다. 오히려 특징을 기반으로동작하는 머신러닝 모델이 더 나은 성능을 보이는 경우도 있다. 따라서, 본 프로젝트에서 역시 머신러닝 모델 중 좋은 성능을 보이는 XGBoost를 혼용하여 사용하였다.
- 두 모델은 $\alpha \cdot \text{CNN-LSTM} + (1-\alpha) \cdot \text{XGBoost}$의 형태로 앙상블을 하였다.
- 예측 모델은 각 건물 별로 설계된다.

CNN-LSTM



데이터 셋을 구조화 하는 방법

- 모델은 위 그림과 같이 특정 시간 범위(이하 윈도우)에 해당하는 특징들을 입력 받고, 그 다음 시점의 전력 사용량을 예측한다. 이를 위해, 윈도우의 크기를 결정해주어야하는데, 본 프로젝트에서는 이를 24로 결정하였다. 이는 24시간 주기로 반복되는 전기 사용량 패턴을 모델이 효율적으로 학습하기 위함이다.
- 모델은 윈도우 크기만큼의 기상 데이터(e.g., 기온, 강수량, 풍속, 풍향, 습도, 현지기압, 일조, 일사, 지면온도)과 휴일 정보를 입력으로 받아 전력 사용량을 예측한다.
- 입력 및 정답 데이터에 대해서는 시계열 처리 모델에서 주로 사용하는 최소-최대 정규화(Min-Max Normalization) 방법을 이용하여 전처리하였다.
- 모델은 1D Convolution NN과 LSTM을 결합한 CNN-LSTM으로 설계하였으며, 구체적인 모델은 다음과 같다.

```
class CNNLSTM(nn.Module):
    def __init__(self):
        super(CNNLSTM, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=24, out_channels=24,
                               kernel_size=3, padding=1)
        self.lstm = nn.LSTM(input_size=24, hidden_size=64,
                            num_layers=5, batch_first=True)
        self.linear1 = nn.Linear(576, 256)
        self.linear2 = nn.Linear(256, 128)
        self.linear3 = nn.Linear(128, 1)

    def forward(self, x):
        x = self.conv1(x)
        x = F.leaky_relu(x)
        x = x.permute(0, 2, 1)
        x, _ = self.lstm(x)
```

```
        x = torch.reshape(x, (x.shape[0], -1))
        x = self.linear1(x)
        x = F.leaky_relu(x)
        x = self.linear2(x)
        x = F.leaky_relu(x)
        x = self.linear3(x)
        x = torch.flatten(x)
        return x
```

XGBoost

- XGBoost는 특정 시점에서의 특징 값들을 입력으로 받고, 그에 맞는 전력 사용량을 예측하기 위한 모델이다.
- XGBoost와 같은 트리 기반 모델의 경우 정규화가 불필요하기에, 정규화 작업은 따로 수행하지 않았다.
- XGBoost의 최적 모델 파라미터를 구하기 위해서 랜덤 서치 방법을 이용하였으며, 파라미터 풀(Pool)은 다음과 같이 설정되었다.

```
param_dist = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9]
}
```

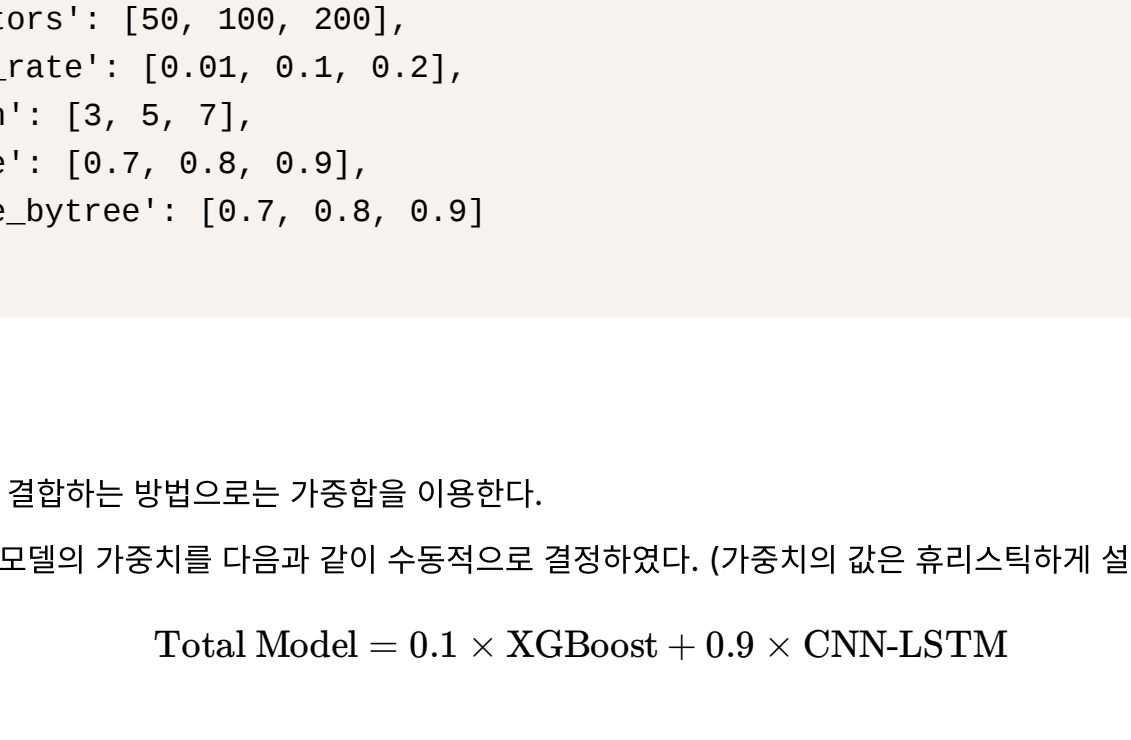
Ensemble

- 두 모델의 출력 결과를 결합하는 방법으로는 가중합을 이용한다.
- 본 프로젝트에서는 각 모델의 가중치를 다음과 같이 수동적으로 결정하였다. (가중치의 값은 휴리스틱하게 설정되었다.)

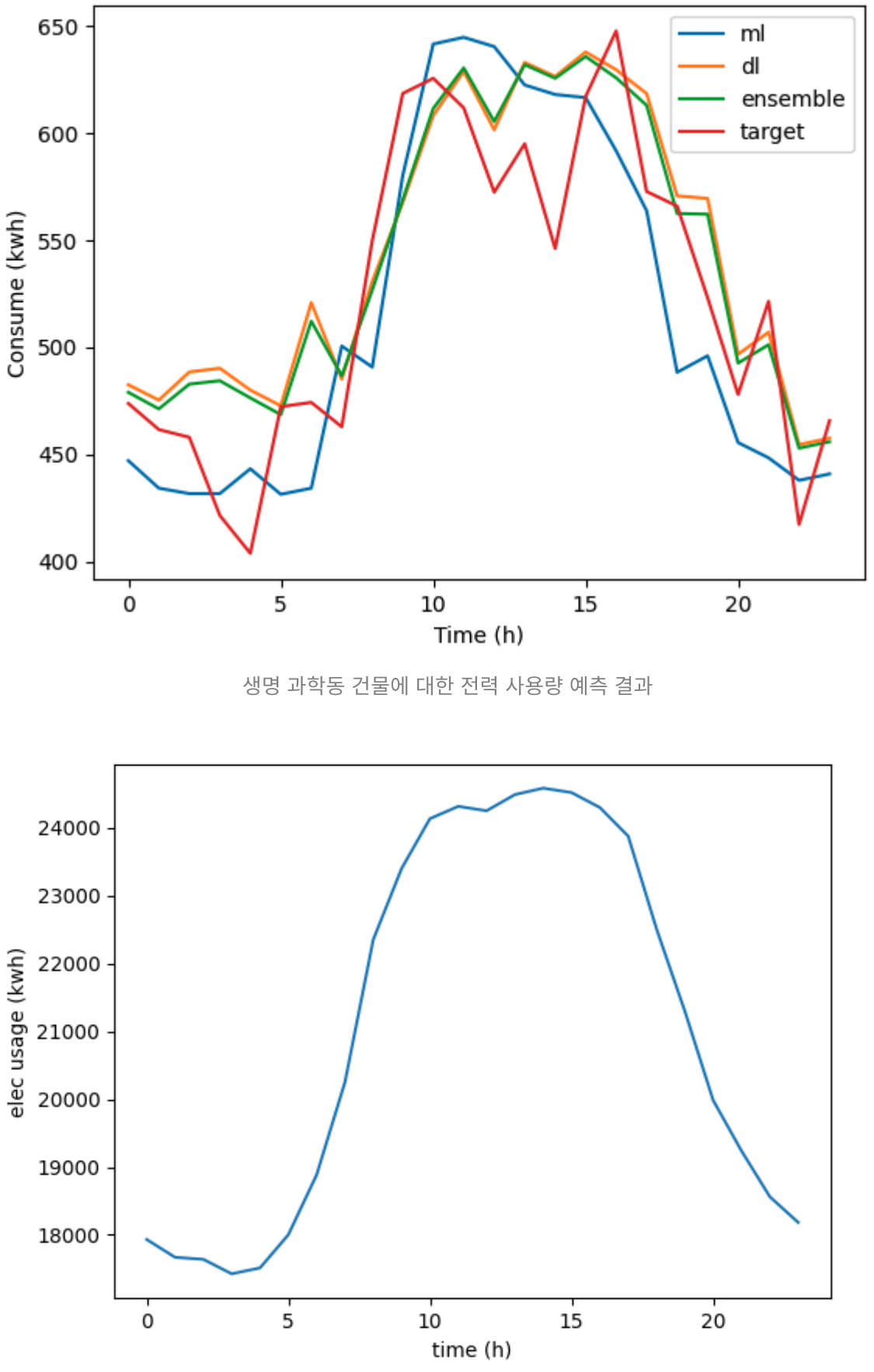
$$\text{Total Model} = 0.1 \times \text{XGBoost} + 0.9 \times \text{CNN-LSTM}$$

결과

[2023.08.31.]에 대한 예측 결과



신재생 공학동 건물에 대한 전력 사용량 예측 결과



생명 과학동 건물에 대한 전력 사용량 예측 결과

캠퍼스 내 전체 전력 사용량

- 모델의 예측 결과와 정답 데이터를 비교하였을때 나름 유사한 경향성을 보임을 확인할 수 있다.
- 전체 건물에 대한 예측 결과는 CSV로 생성한다. (**Challenge03/**elec_inference_result.csv)
- 모델링 후 확인한 결과, HV-NM1 건물에 대해서는 값이 0으로 측정되어 예측을 수행하는 것이 무의미하다고 판단되며, 따라서, 캠퍼스 전체의 전력 사용량을 산출할때는 해당 건물에 대한 결과는 제외하였다. (부하 예측 과정은 우선 모든 칼럼에 대해서 수행하였다.)

2024.08.06. 공지에 의하면, 추가 조건으로 인해 예측 대상의 건물이 지정되었지만, 본 과정에서는 일단 모든 column에 대한 예측을 수행한다. 해당 공지의 내용 반영은 “Challenge 3 - 전기 요금 최소화 알고리즘 및 검증” 단계에서 반영된다.

Challenge 2 - 태양광 발전량 예측 알고리즘 개발 및 검증

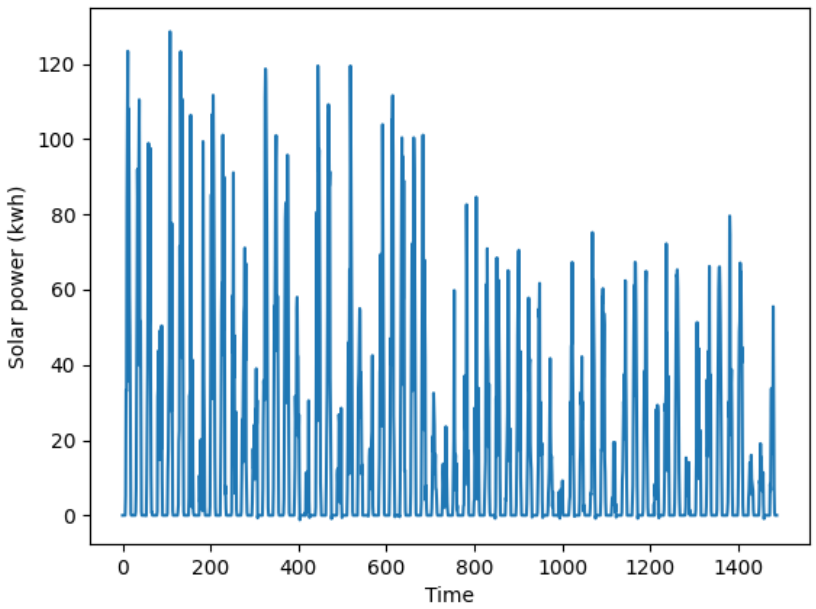
디렉토리: Challenge02

데이터셋

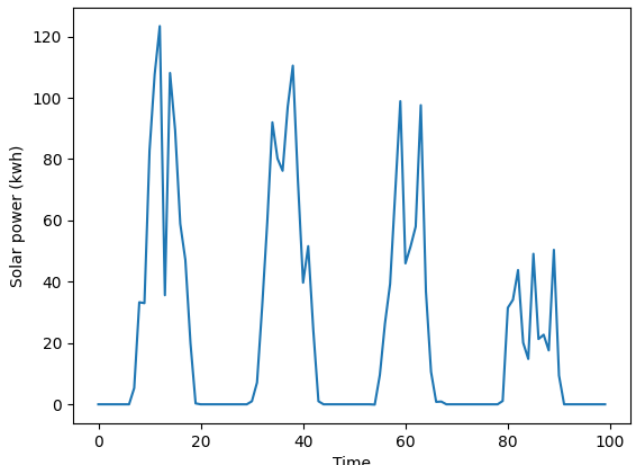
- “태양광발전 일보” 데이터셋에는 환경 감시 정보와 패널별 발전량이 기록되어 있다. 본 프로젝트에서는 머신러닝 및 딥러닝 모델을 활용하여 이 문제를 해결하고자 하였으며, 환경 감시 정보와 기상 정보를 결합하여 입력으로 제공하였다. 모델은 이러한 입력을 바탕으로 특정 상황에서의 태양광 발전량을 예측하도록 설계되었다.

- 따라서, 모델에 입력으로 제공되는 정보는 "수평면", "외기온도", "경사면", "모듈온도", "기온", "강수량", "풍속", "풍향", "습도", "현지기압", "일조", "일사", "지면온도"이며, 모델은 이를 바탕으로 당시의 "시간당 발전량"을 예측한다.
- 데이터에는 발전량이 음수값으로 측정된 문제가 존재한다. 따라서, 음수 발전값인 경우 모두 0으로 대체하였다.
- 태양광 발전량 예측은 앞서 수행한 부하 예측에 비해 기상 정보(e.g., 강수, 일조, 일사 등)에 더 민감하게 반응할 것으로 사료되며, 관련 특징들을 모델들에게 잘 주어지는 것이 중요할 것으로 보인다.
- 모델 학습 과정에서는 22년도 데이터 셋과 결측치가 존재하지 않는 23년도 데이터 셋의 일부를 혼합하여 사용하였다.

데이터 시각화



축구장에서의 태양광 발전량 (장기)



축구장에서의 태양광 발전량 (단기)

- 해가 떠 있는 주간에만 전력이 생성되기 때문에, 명확한 주기성을 지닌다.
- 태양광 발전은 오직 환경적 요인에만 영향을 받기 때문에, 주말 여부와 같은 외적 요인을 고려할 필요는 없다.

모델링: CNN-LSTM, XGBOOST

- 태양광 발전량 예측 모델에서도 부하 예측 알고리즘과 마찬가지로 CNN-LSTM과 XGBoost를 앙상블한 모델을 사용하였다. 따라서, 모델의 구조와 관련된 설명은 간략히 하겠다.

CNN-LSTM

- CNN-LSTM은 Convolution Neural Network와 LSTM을 결합한 모델로 시계열 예측에 특화된 모델로 알려져있다.
- 윈도우 크기는 24로 설정하였다.
- 최소 최대 정규화(Min-max Normalization)를 진행하였다.
- 구체적인 모델 아키텍쳐는 다음과 같다.

```
class CNNLSTM(nn.Module):
    def __init__(self):
        super(CNNLSTM, self).__init__()
        self.conv1 = nn.Conv1d(in_channels=24, out_channels=24,
                               kernel_size=3, padding=1)
        self.lstm = nn.LSTM(input_size=24, hidden_size=64,
                            num_layers=5, batch_first=True)
        self.linear1 = nn.Linear(832, 256)
        self.linear2 = nn.Linear(256, 128)
        self.linear3 = nn.Linear(128, 1)

    def forward(self, x):
        x = self.conv1(x)
```

```
        x = F.leaky_relu(x)
        x = x.permute(0, 2, 1)
        x, _ = self.lstm(x)
        x = torch.reshape(x, (x.shape[0], -1))
        x = self.linear1(x)
        x = F.leaky_relu(x)
        x = self.linear2(x)
        x = F.leaky_relu(x)
        x = self.linear3(x)
        x = torch.flatten(x)
        return x
```

XGBoost

- XGBoost는 특정 시점에서의 특징 값들을 입력으로 받고, 그에 맞는 전력 사용량을 예측하기 위한 모델이다.
- XGBoost와 같은 트리 기반 모델의 경우 정규화가 불필요하기에, 정규화 작업은 따로 수행하지 않았다.
- XGBoost의 최적 모델 파라미터를 구하기 위해서 랜덤 서치 방법을 이용하였으며, 파라미터 풀(Pool)은 다음과 같이 설정되었다.

```
param_dist = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9]
}
```

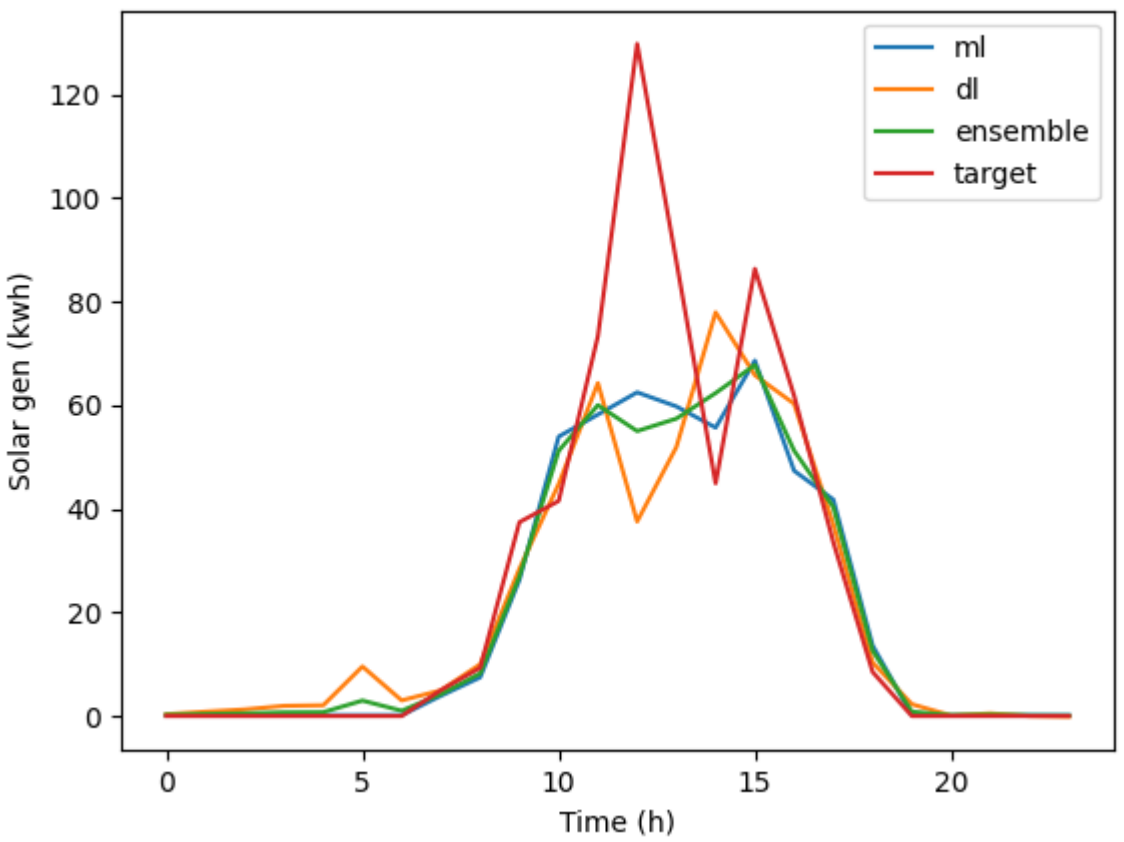
Ensemble

- 두 모델의 출력 결과를 결합하는 방법으로는 가중합을 이용한다.
- 본 프로젝트에서는 각 모델의 가중치를 다음과 같이 결정하였다.(가중치의 값은 휴리스틱하게 설정되었다.)

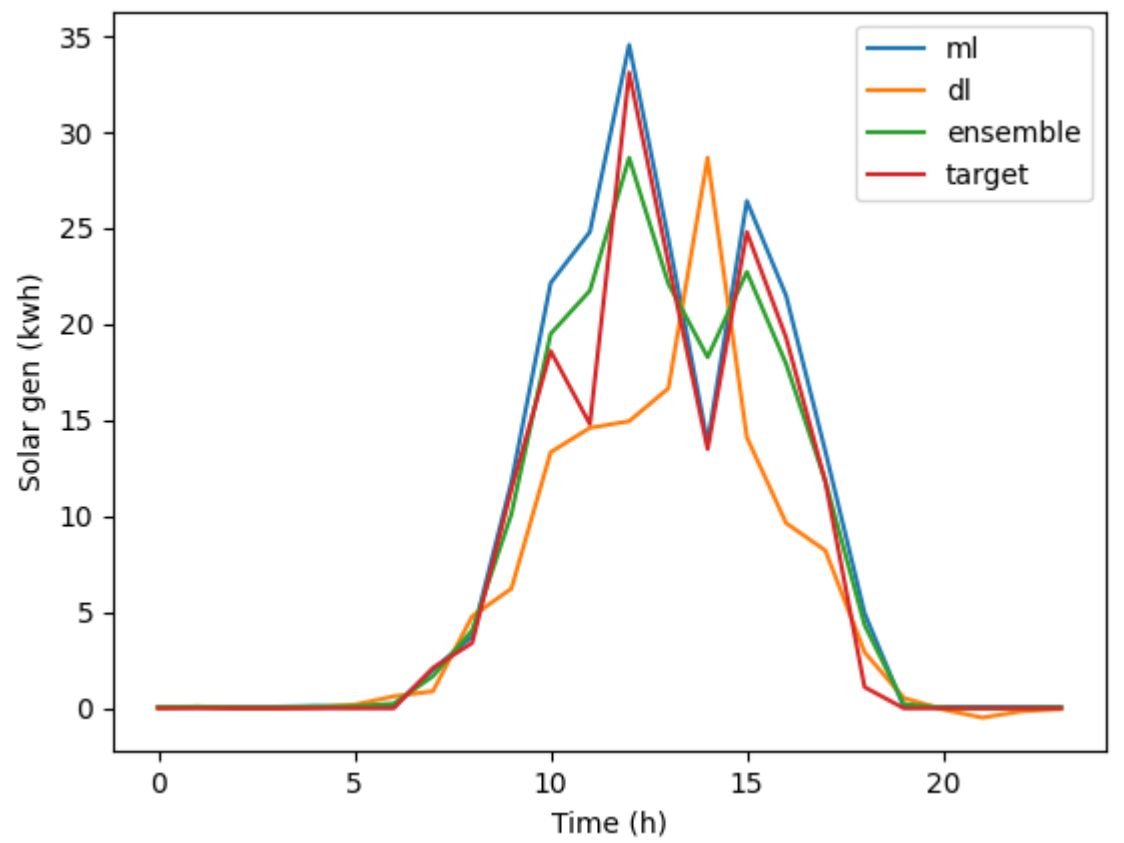
$$\text{Total Model} = 0.7 \times \text{XGBoost} + 0.3 \times \text{CNN-LSTM}$$

결과

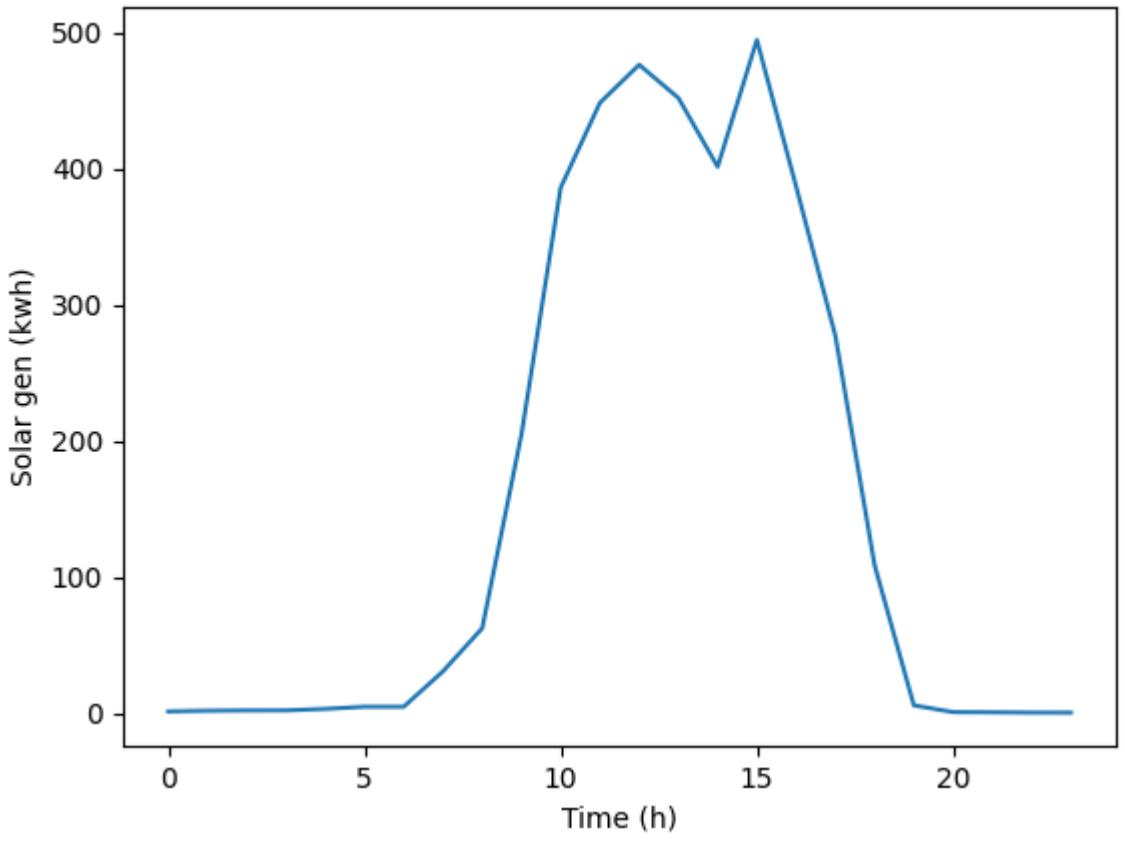
[2023.08.31.]에 대한 예측 결과



축구장에서 생산된 태양광 발전량



학생회관에서 생산된 태양광 발전량



캠퍼스 내 전체 태양광 발전량

- 모델의 출력 값이 정답 값과 유사한 것을 확인할 수 있다. 이는 모델이 적절한 추론을 하고 있다는 의미이다.
- 전체 발전량과 관련해서는 추가 데이터로 제출한다. (Challenge3/solargrid_inference_result.csv)
- 모델이 예측한 값이 음수인 경우 0으로 클리핑(clipping)을 진행하였다.

Challenge 3 - 전기 요금 최소화 알고리즘 개발 및 검증

디렉토리: Challenge03

- 전기 요금 최적화를 위해서 “**컨벡스 최적화 (Convex optimization)**” 방법을 이용하였다. 컨벡스 최적화는 조건식을 만족시키면서, 목적 함수을 최적화 하기 위한 방법론이다.
- 본 문제에서는 “**전력 사용량 예측 결과”와 “태양광 발전 생산량 예측 결과**”를 기반으로 전기요금을 최적화 하기 위한 솔루션을 찾는다.
- 전력 사용량에 대해서는 모든 칼럼에 대하여 예측을 진행 하였으나, 공지에 따르면 건물과 관련된 칼럼만을 사용하라고하였기에 이를 반영한다. 따라서, 전기전자컴퓨터공학동, 신소재공학동, 생명과학동, 기계공학동, LG도서관, 창업B동, 금호관, 기혼자아파트E동, 기숙사9동, 고등광/극초단, 신재생에너지동, 대학B동, 대학기숙사A동, 제2학생회관, 교원아파트, 대학C동, 중앙연구기기센터, 대학A동, 다산빌딩, 산학협력연구동과 관련된 예측 결과만을 사용한다.
  (
  elec_inference_result.csv 에서 Valid_sum_of_value column이 해당 건물들에서의 총합 전력 사용량을 의미한다.)
- 해당 문제의 가정에는 BESS가 존재하기 때문에, 전기 요금이 저렴한 시점에서 배터리를 충전할 수 있다는 사실을 고려하여야한다. (사실 컨벡스 최적화 과정에서 최적화 대상 변수로 설정해둔다면 최적화가 자동으로 진행된다.)

조건식

- 문제를 해결하기 위해 다음과 같은 수식을 작성하였다. 각 수식에 대한 근거는 OT 자료에 해당하는 제약 조건으로부터 기인하였다.

Electricity Cost Optimization Problem

$$\text{Minimize} \quad \sum_{t=1}^{T} c_t \cdot P_{\text{grid},t} \tag{1}$$

$$\text{subject to} \quad \text{SOC}_{t+1} = \text{SOC}_t + (\eta_{\text{charge}} \cdot P_{\text{charge},t} - \frac{P_{\text{discharge},t}}{\eta_{\text{discharge}}}), \quad \forall t, \tag{2}$$

$$P_{\text{load},t} = P_{\text{pv},t} + P_{\text{discharge},t} - P_{\text{charge},t} + P_{\text{grid},t}, \quad \forall t, \tag{3}$$

$$0 \leq P_{\text{charge},t} \leq P_{\text{bess,max}}, \quad \forall t, \tag{4}$$

$$0 \leq P_{\text{discharge},t} \leq P_{\text{bess,max}}, \quad \forall t, \tag{5}$$

$$\text{SOC}_{\text{min}} \leq \text{SOC}_t \leq \text{SOC}_{\text{max}}, \quad \forall t, \tag{6}$$

$$\text{SOC}_0 = \text{SOC}_T = \text{SOC}_{\text{initial}}, \tag{7}$$

$$U_{C,t} + U_{D,t} \leq 1, \quad \forall t, \tag{8}$$

$$|P_t - P_{t-1}| \leq \text{ramp\_rate}, \quad \forall t > 0. \tag{9}$$

전기 요금 최적화를 위한 목적식과 그에 따른 조건

(1) 최적화하기 위한 목적식이다. 전체 시간 동안 외부 전력망으로부터 구매한 전력량에 대한 총 비용을 의미한다. (c_t: 시간 t에서의 전기 요금, $P_{grid,t}$: 시간 t에서 외부 전력망에서 구매하는 전력량)

(2) BESS의 충전 및 방전 효율을 고려한 SOC의 업데이트 식이다. (SOC_t: 배터리의 상태, η: 배터리의 충/방전 효율, P: 충/방전되는 전력량)

(3) 전력 부하 균형식이다. 전체 전력 사용량은 태양광 발전, 방전 , 충전, 전력망에서의 구매량의 합으로써 정의된다.

(4)~(5) 충전과 방전 속도는 설정된 속도를 초과할 수 없음을 의미하는 제한식이다. 본 프로젝트에서는 250으로 설정되었다.

(6) 배터리 충전 상태와 관련된 제약식이다. 충전 상태는 설정된 최소, 최대의 경계를 벗어나서는 안된다. 본 프로젝트에서 설정된 SOC의 범위는 20~80% 이다.

(7) 하루의 시작과 끝의 배터리 충전량은 동일해야한다는 제약식이다. 본 프로젝트에서는 50%로 설정되었다.

(8) 배터리의 상호 배타성을 표현하기 위한 제약식이다. 배터리는 충전과 방전을 동시에 수행할 수 없다. ($U_{C,t}$: t 시간에 배터리가 충전 상태인지 여부, $U_{D,t}$: t시간에 배터리가 방전 상태인지 여부)

⇒ 8번 조건식 같은 경우에는 교수님 면담 후 추가한 제약식이다.
(참고 논문: Abunima, Hamza, et al. "Two-stage stochastic optimization for operating a renewable-based microgrid." *Applied Energy* 325 (2022): 119848.)

(9) 램프 레이트라는 제약조건이 존재한다. 이는 충전 또는 방전 전력이 단위 시간당 변할 수 있는 최대 속도를 의미한다. 일반적으로 배터리의 출력량의 10~20%를 변수로 결정한다. 본 프로젝트에서는 20kw로 결정한다.

⇒ 9번 조건식 같은 경우에는 교수님 면담 후 추가한 제약식이다.

(참고 논문: Badiei, Masoud, Na Li, and Adam Wierman. "Online convex optimization with ramp constraints." *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015.)

전기 요금

교육용전력(을)

* 계약전력 1,000kW이상

구분		기본요금 (원/kW)	전력량요금 (원/kWh)			
			시간대	여름철 (6~8월)	봄·가을철 (3~5, 9~10월)	겨울철 (11~2월)
고압A	선택Ⅰ	6,090	경부하	44.8	44.8	48.8
			중간부하	89.5	59.2	88.0
			최대부하	155.4	79.7	126.7
	선택Ⅱ	6,980	경부하	40.3	40.3	44.3
			중간부하	85.0	54.7	83.5
			최대부하	150.9	75.2	122.2
고압B	선택Ⅰ	6,090	경부하	43.3	43.3	47.1
			중간부하	86.8	57.5	85.1
			최대부하	149.7	77.3	122.4
	선택Ⅱ	6,980	경부하	38.8	38.8	42.6
			중간부하	82.3	53.0	80.6
			최대부하	145.2	72.8	117.9

시간대 \ 계절	여름철 (6~8월)	봄·가을철 (3~5, 9~10월)	겨울철 (11~2월)
경부하	23:00~09:00	23:00~09:00	23:00~09:00
중간부하	09:00~10:00 12:00~13:00 17:00~23:00	09:00~10:00 12:00~13:00 17:00~23:00	09:00~10:00 12:00~17:00 20:00~22:00
최대부하	10:00~12:00 13:00~17:00	10:00~12:00 13:00~17:00	10:00~12:00 17:00~20:00 22:00~23:00

전압구분

구분	적용범위
저압	표준전압 220V, 380V 고객
고압A	표준전압 3,300V 이상 66,000V 이하 고객
고압B	표준전압 154,000V 고객
고압C	표준전압 345,000V 이상 고객

교육용 전기 요금

- 시간에 따라 전력량 요금이 변동하기 때문에, 최적화 과정에서 시간별 요금을 고려하는 것은 매우 중요하다.
- 공지에 따르면 전기 요금은 “고압A, 선택 2”로 제한하라고 하였기 때문에 이를 반영한다. 이때 기본 요금에 대해서는 “공급 용량에 따른 계약”에 따라 달라지기에 사실상 최적화가 불가능하다. 따라서 전기 요금 산출 단계에서 기본요금은 반영하지 않았다.
- 예측 대상 일시(2023.08.31.)는 8월 중이기에 전력량 요금 계산은 여름철 시간대를 적용한다 .

```
# 전기 요금 데이터
cost_per_kWh = np.array([
    40.3,  # 00:00
    40.3,  # 01:00
    40.3,  # 02:00
    40.3,  # 03:00
    40.3,  # 04:00
    40.3,  # 05:00
    40.3,  # 06:00
    40.3,  # 07:00
    40.3,  # 08:00
    85.0,  # 09:00
    150.9, # 10:00
    150.9, # 11:00
    85.0,  # 12:00
    150.9, # 13:00
    150.9, # 14:00
    150.9, # 15:00
    150.9, # 16:00
    85.0,  # 17:00
    85.0,  # 18:00
    85.0,  # 19:00
    85.0,  # 20:00
    85.0,  # 21:00
    85.0,  # 22:00
    40.3   # 23:00
])
```

모델링

- 최적화 과정에서 불변하는 값들은 상수로 설정한다.
  ⇒ 설정된 상수는 OT 자료로부터 근거한다.

```
P_bess_max = 250  # speed of bess
C_bess = 750  # cap of bess
rate_charge = 0.93
rate_discharge = 0.93
SOC_min = 0.2 * C_bess
SOC_max = 0.8 * C_bess
SOC_init = 0.5 * C_bess
```

- 충전 전력, 방전 전력, 배터리의 상태, 외부 전력 구매량, 단위시간(1시간) 동안의 충전 상태 비율, 단위시간(1시간) 동안의 방전 상태 비율을 최적화변수로 선언한다. 최적화 변수로 선언된다면, 변수의 값이 가변적으로 변하면서 목적식을 만족하는 최적의 값을 찾는다.

```
# optimization variable
P_charge = cp.Variable(T, nonneg=True)  # charge
P_discharge = cp.Variable(T, nonneg=True)  # discharge
SOC = cp.Variable(T+1, nonneg=True)  # SOC
P_grid = cp.Variable(T, nonneg=True)  # grid
charge_rate = cp.Variable(T, nonneg=True) # exclusivity charge
discharge_rate = cp.Variable(T, nonneg=True) # exclusivity discharge
```

- 전기 요금을 최적화 하기 위한 목적 함수과 제약식을 설정한다. 제약식은 시간(T)이 증가함에 따라 추가되도록 설정하였다.
  - 위에서 언급한 수식에 맞추어 최적화 코드를 작성함
  - 이때, **충전 및 방전의 상호 배타 조건** 같은 경우에는 시간 단위로 충전 방전을 이산적으로 적용하기 어렵기 때문에, 단위 시간(1 시간)에서의 충전 비율과 방전 비율로 최적화 하였다. 따라서, 해당 조건을 만족시키기 위해서는 “충전 비율 + 방전 비율이 1 이하”가 되는 조건으로 제한식을 정의하였다. 또한, 1시간당 최대 250kwh 만큼의 충전 또는 방전을 할 수 있지만, 이를 충전 및 방전 비율로써 제한하였다.

```
# objective function
objective = cp.Minimize(cp.sum(cp.multiply(cost_per_kWh, P_grid)))

# constraint
constraints = [
    SOC[0] == SOC_init,
    SOC[T] == SOC_init
]

for t in range(T):
    # update
    constraints  += [
        # SOC update
        SOC[t+1] == SOC[t] + (P_charge[t] * rate_charge - P_discharge[t] / rate_discharg
e),

        # balance
        sum_of_usage[t] == sum_of_gen[t] + P_discharge[t] - P_charge[t] + P_grid[t],

        # charge and discharge constraint (e.g., SOC cap-, speed)
        P_charge[t] <= P_bess_max,
        P_discharge[t] <= P_bess_max,
```

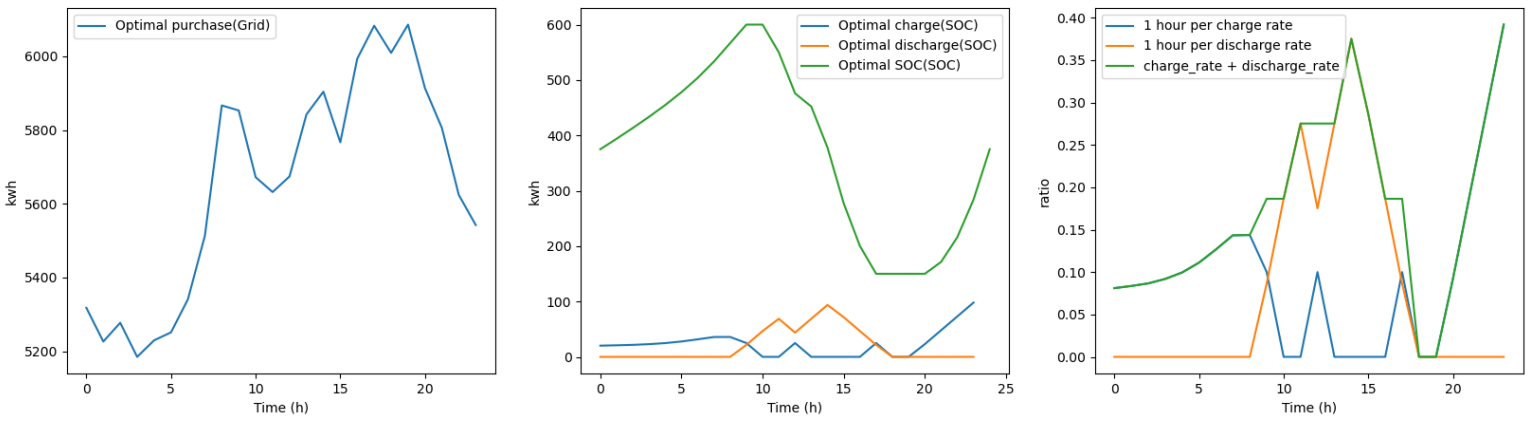
```
        SOC[t+1] >= SOC_min,
        SOC[t+1] <= SOC_max,

        # exclusivity - charge or discharge
        charge_rate[t] + discharge_rate[t] <= 1,
        P_charge[t] == P_bess_max * charge_rate[t],
        P_discharge[t] == P_bess_max * discharge_rate[t]
    ]

    # 램프 레이트 제약 조건 수정

    if t > 0:
        constraints += [  # 충/방전량이 올라갈 때, 낮아질 때 고려
            P_charge[t] - P_charge[t-1] <= ramp_rate,
            P_charge[t-1] - P_charge[t] <= ramp_rate,
            P_discharge[t] - P_discharge[t-1] <= ramp_rate,
            P_discharge[t-1] - P_discharge[t] <= ramp_rate
        ]
```

결과




```
최소화된 총 전기 요금 (원): 11418483.19

최적화 하지 않았을 때의 전기 요금 (원): 12421228.71
```

- 해당 그래프는 왼쪽부터, 외부(한국 전력 공사)에서 구입한 전력량, SOC의 충전, 방전 및 상태, 1시간 단위의 충전과 방전 비율을 의미한다
  .
- 전기 요금을 최적화하기 위해 배터리는 충전과 방전 과정을 반복한다.
- 한 가지 특징점으로는 전기 요금이 저렴한 시간대에 배터리를 충전해두고, 비싼 시간에 배터리를 방전하는 경향을 보인다.
- 해당 알고리즘으로 인해 대략 100만원 가량의 전기 요금을 절약할 수 있게 되었다.