8. 에너지 사용량예측 딥러닝모델 개발

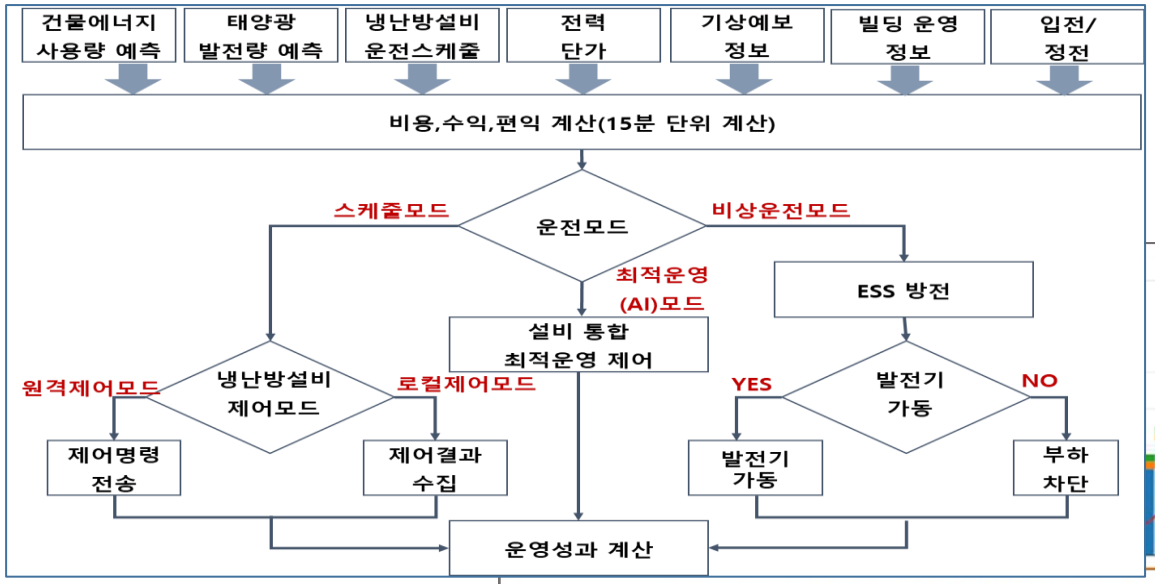
박경규

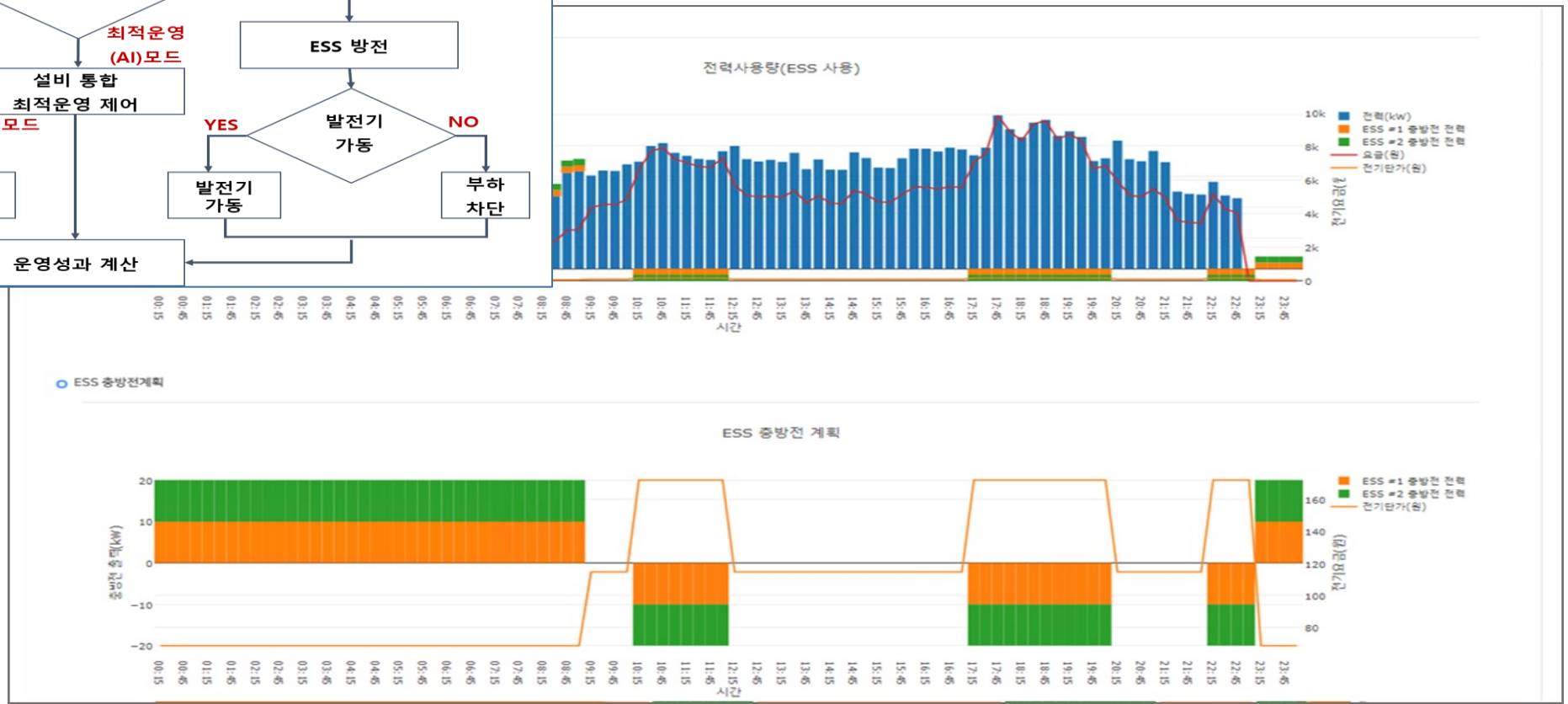
에너지 사용량 예측모델

에너지 사용량 예측은 건물 에너지 최적화에 필수적인 기본기능입니다.

에너지 사용량 예측

에너지 사용량 예측은 건물 에너지 최적화에 필수적인 기본기능입니다.
예측 모델은 대부분의 운영 최적화, 스케줄링에 필요하므로 다양한 분야에 활용이 가능합니다.



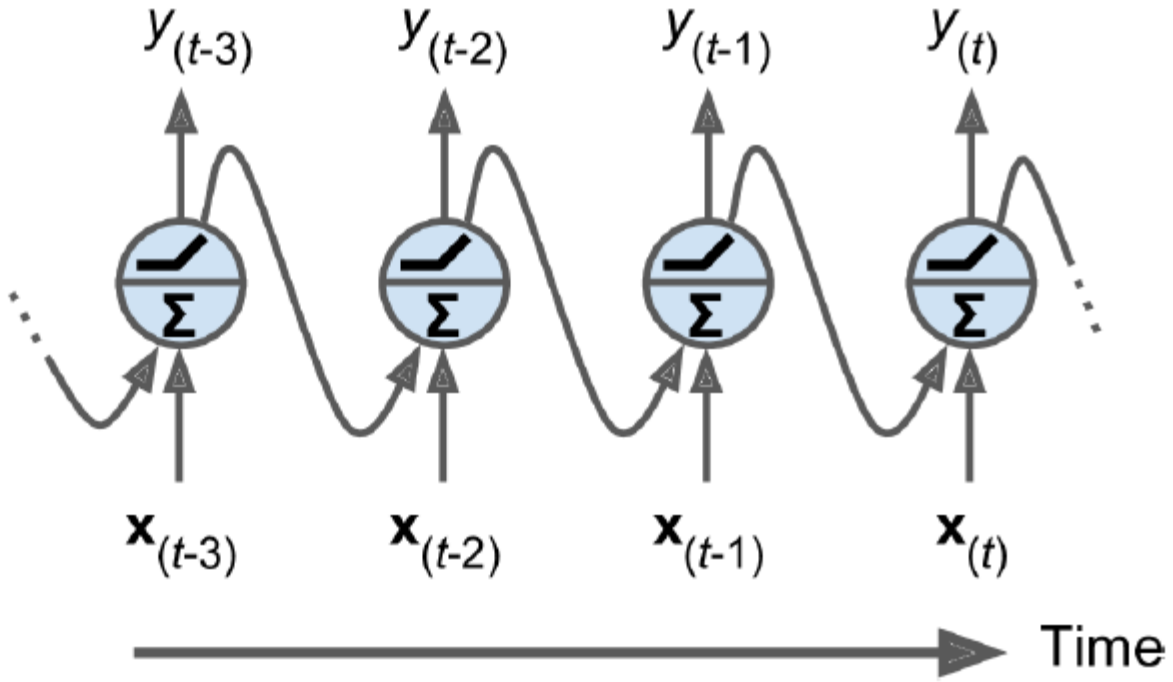


예측 모델 - RNN(Recurrent Neural Network)

순환신경망은 고정 길이 입력이 아닌 임의 길이를 가진 시퀀스를 다룰 수 있습니다. 순환신경망은 시계열 데이터를 분석해서 미래값을 예측하고 문장, 오디오를 입력으로 받아 자동번역, 자연어처리에 유용합니다.

■ 순환뉴런

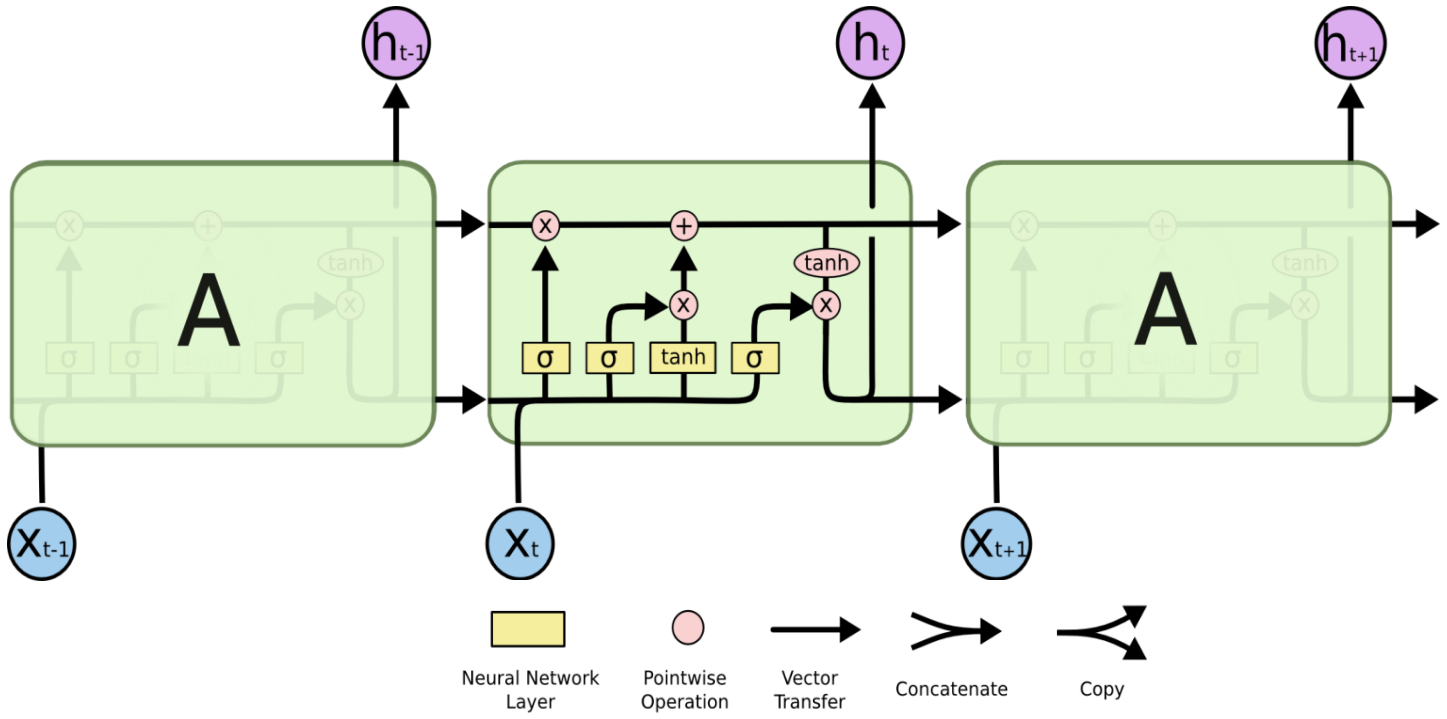
■ 순환뉴런을 타임 스텝으로 펼친 모습



출처 : 도서, 핸즈온 머신러닝 2판

예측 모델 - LSTM(Long Short-Term Memory)

LSTM 네트워크는 장기적인 종속성을 학습할 수 있는 특수한 종류의 RNN입니다.
LSTM은 RNN과 동일하게 입력과 출력사이 신경망이 재귀하는 구조를 갖고 있습니다.
그러나 RNN은 재귀를 통한 정보전이 및 전파가 하나의 레이어로 제어되는 반면
LSTM은 Forget gate, Input gate, Output gate를 통한 정보전이 및 전파를 제어합니다.

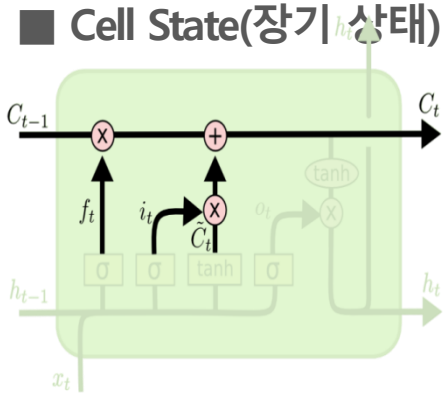


출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

예측 모델 - LSTM(Long Short-Term Memory)

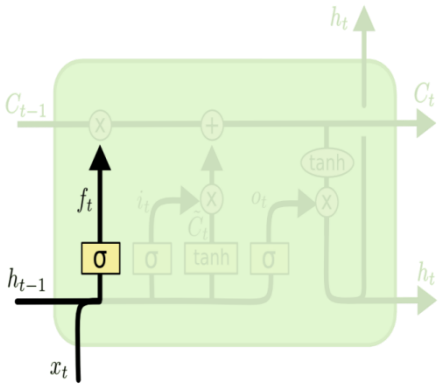
■ LSTM 구조

■ Cell State(장기 상태)



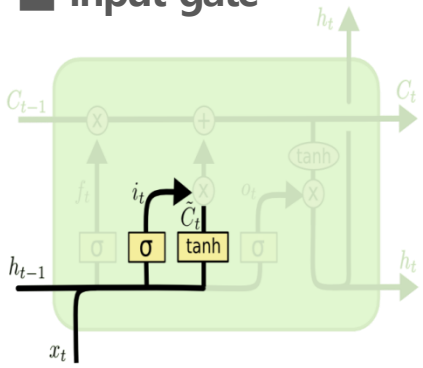
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

■ forget gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

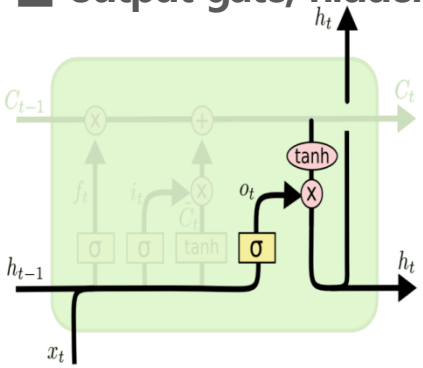
■ input gate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

■ output gate, hidden state(단기 상태)



$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

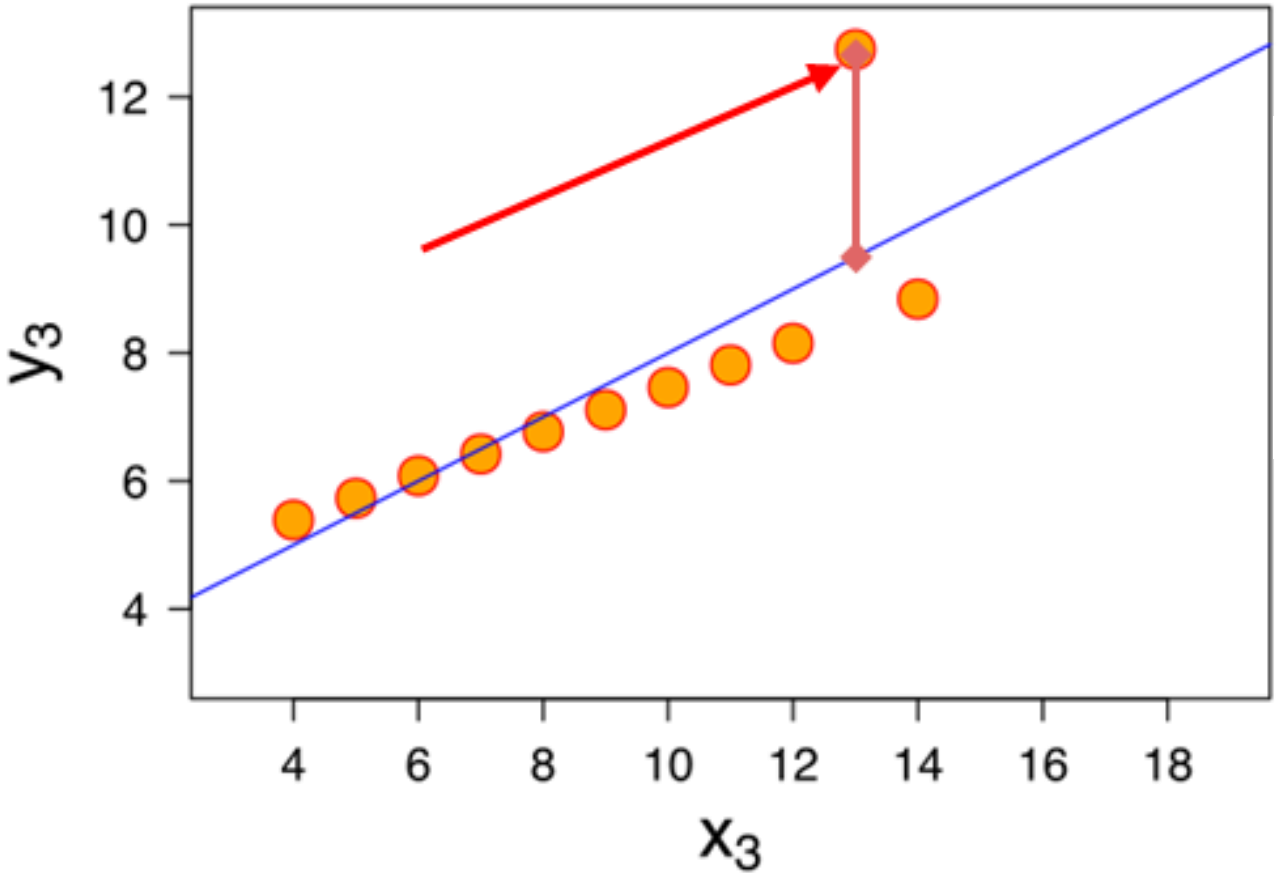
$$h_t = o_t * \tanh\left(C_t\right)$$

출처 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

예측 모델 - 성능측정

예측 모델의 성능 측정은 MSE를 사용하며, 함수로 학습을 진행하면서 지속적으로 측정을 합니다.

■ MSE(Mean Squared Error)

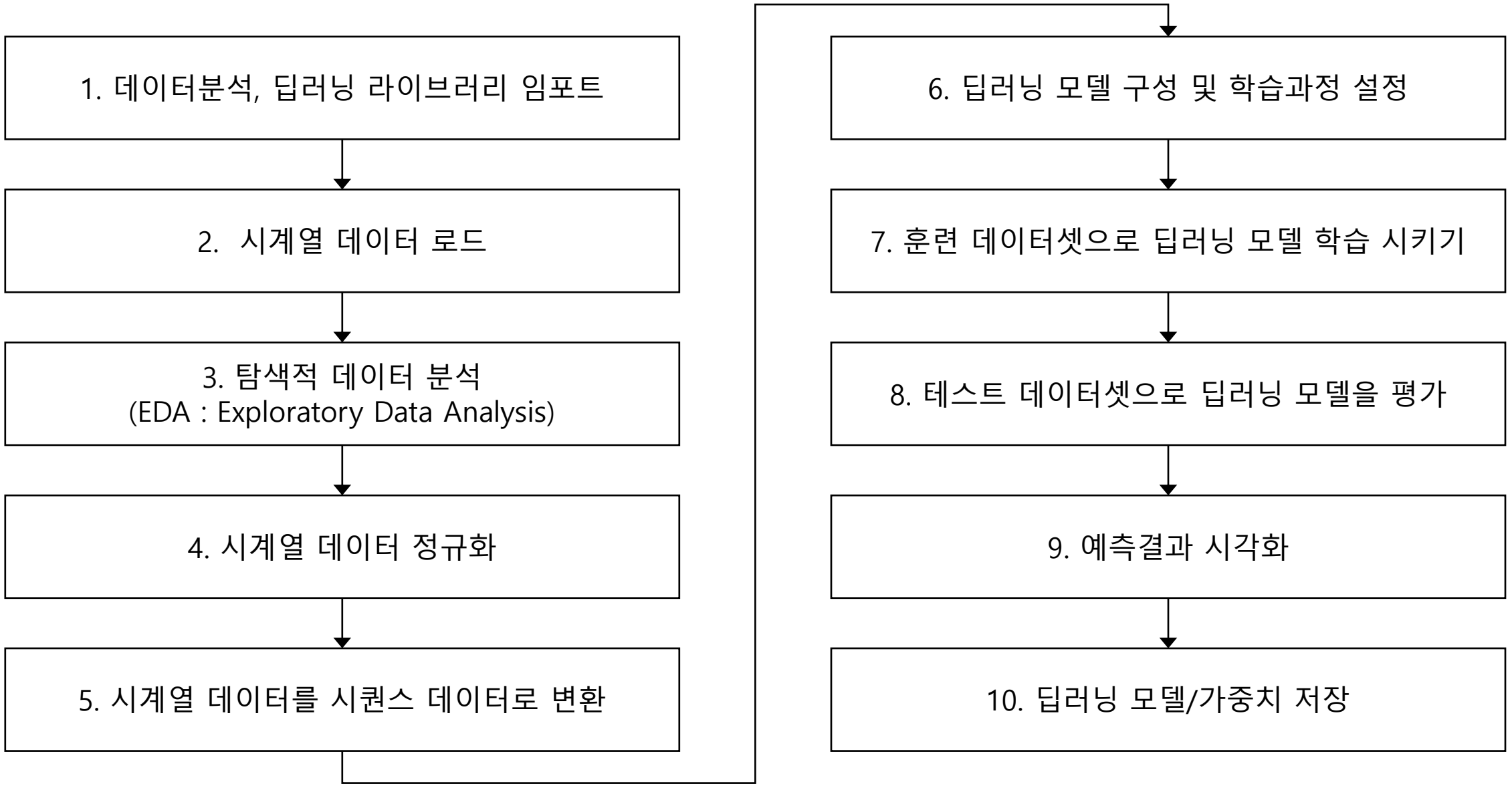


$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

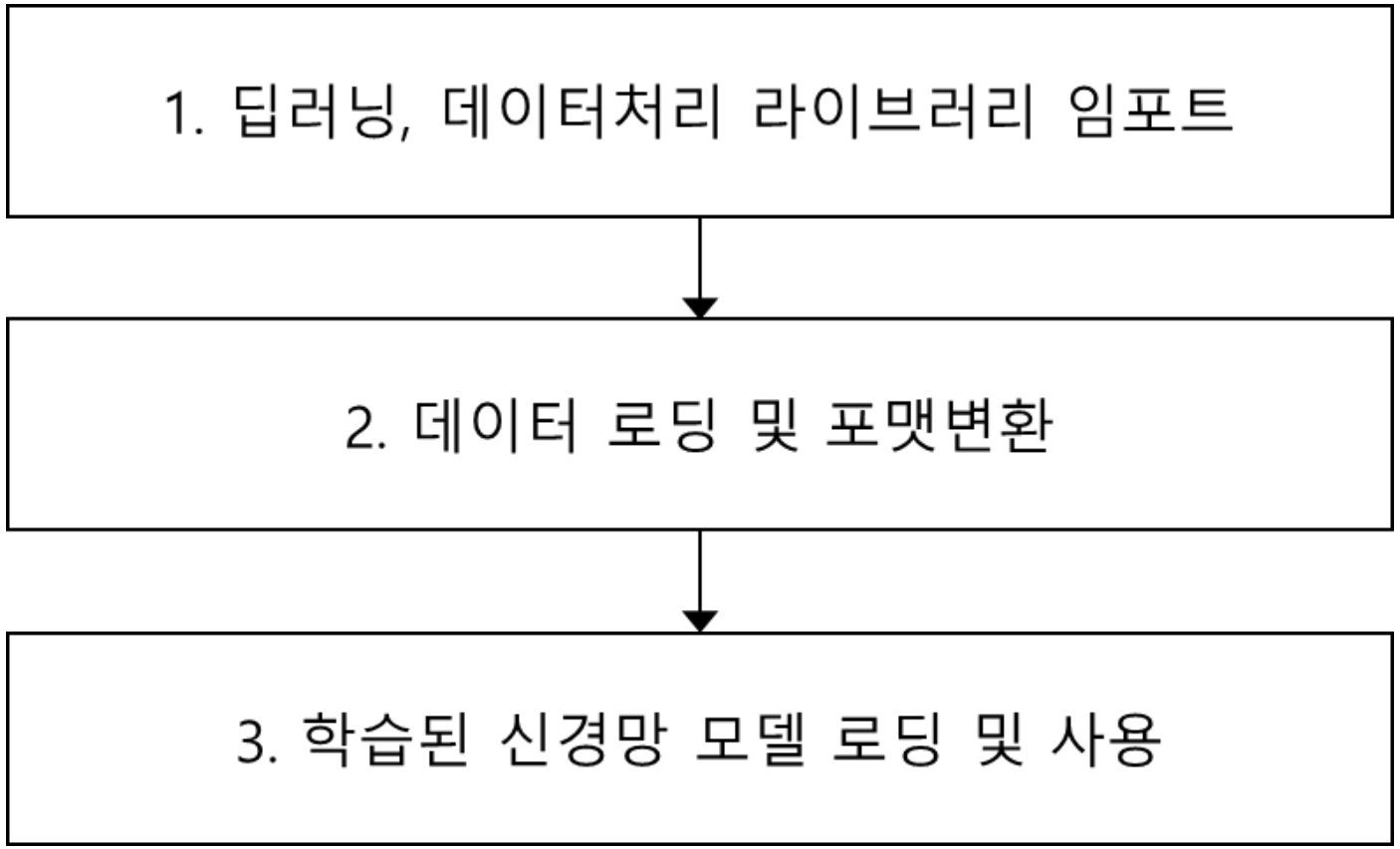
레이블 값 (실제값)

y_hat (모델이 예측한 값)

예측모델 개발 방법


1. 데이터분석, 딥러닝 라이브러리 임포트
2. 시계열 데이터 로드
3. 탐색적 데이터 분석
(EDA : Exploratory Data Analysis)
4. 시계열 데이터 정규화
5. 시계열 데이터를 시퀀스 데이터로 변환
6. 딥러닝 모델 구성 및 학습과정 설정
7. 훈련 데이터셋으로 딥러닝 모델 학습 시키기
8. 테스트 데이터셋으로 딥러닝 모델을 평가
9. 예측결과 시각화
10. 딥러닝 모델/가중치 저장

예측모델 사용 방법



실습 데이터셋

■ 데이터 파일 : e_usage_train.csv, e_usage_test.csv

- 데이터 설명 : ABC 빌딩의 15분 전기에너지 사용량 데이터
- e_usage_train.csv : 모델 학습(Train) 데이터, 70,000개
- e_usage_test.csv : 모델 성능 테스트(Test) 데이터, 35,040개
- 빌딩의 전기에너지 검침 주기가 15분으로,
- 1시간에는 4개의 데이터, 하루에는 96개(4개x24시) 데이터,
- 1년 기간에는 35,040개(4개x24시x365일)의 데이터가 있습니다.

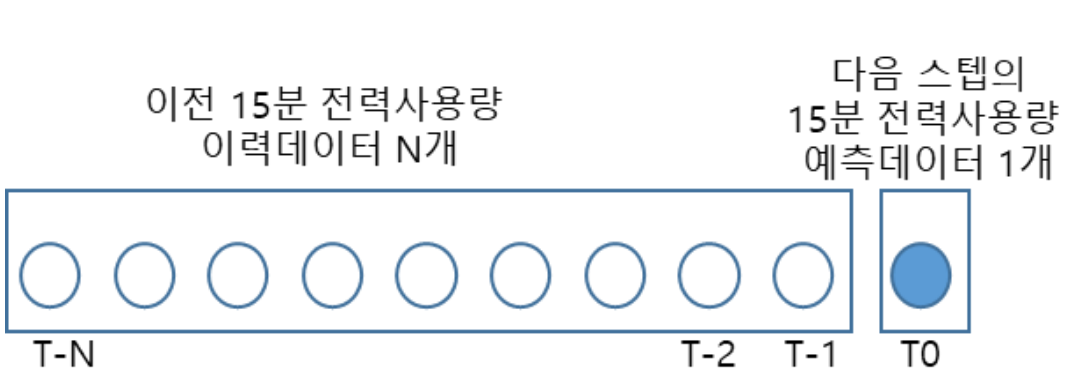
■ 데이터 컬럼명

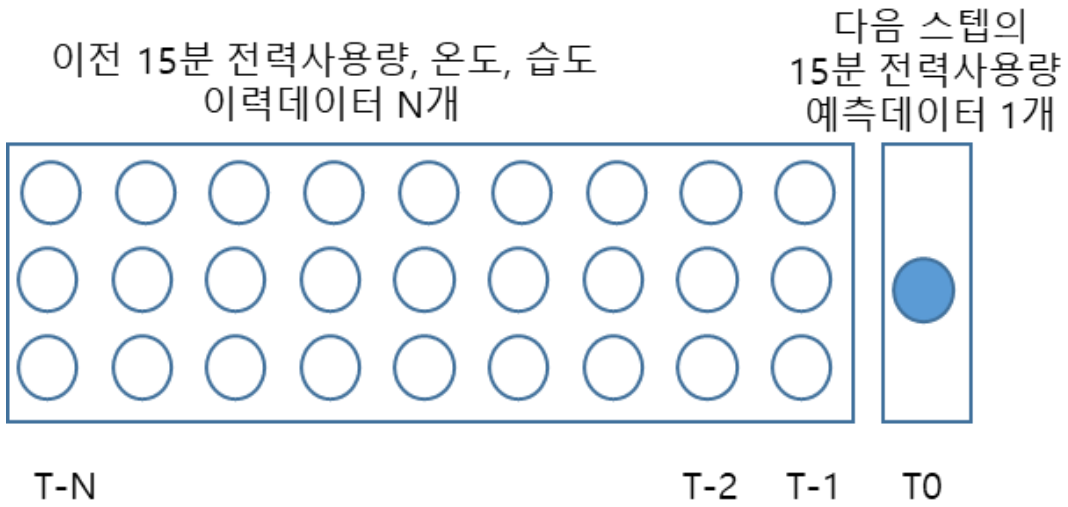
- b_name : 빌딩 이름
- daq_time : 데이터 수집 시간
- wday : 요일 구분
- day_type : 일 구분
  1 - 평일, 2 - 토요일, 3 - 일요일, 휴일
- temp : 온도(℃)
- rh : 상대습도(%)

b_name	daq_time	wday	day_type	hour	temp	rh	p_usage
ABC	2016-01-01 0:15	5	3	1	-2.5	99	229
ABC	2016-01-01 0:30	5	3	1	-2.5	99	231
ABC	2016-01-01 0:45	5	3	1	-2.5	99	231
ABC	2016-01-01 1:00	5	3	1	-3.1	100	226
ABC	2016-01-01 1:15	5	3	2	-3.1	100	229
ABC	2016-01-01 1:30	5	3	2	-3.1	100	223
ABC	2016-01-01 1:45	5	3	2	-3.1	100	233
ABC	2016-01-01 2:00	5	3	2	-3.1	100	234
ABC	2016-01-01 2:15	5	3	3	-3.1	100	230
ABC	2016-01-01 2:30	5	3	3	-3.1	100	228
ABC	2016-01-01 2:45	5	3	3	-3.1	100	224
ABC	2016-01-01 3:00	5	3	3	-2.9	100	226
ABC	2016-01-01 3:15	5	3	4	-2.9	100	234

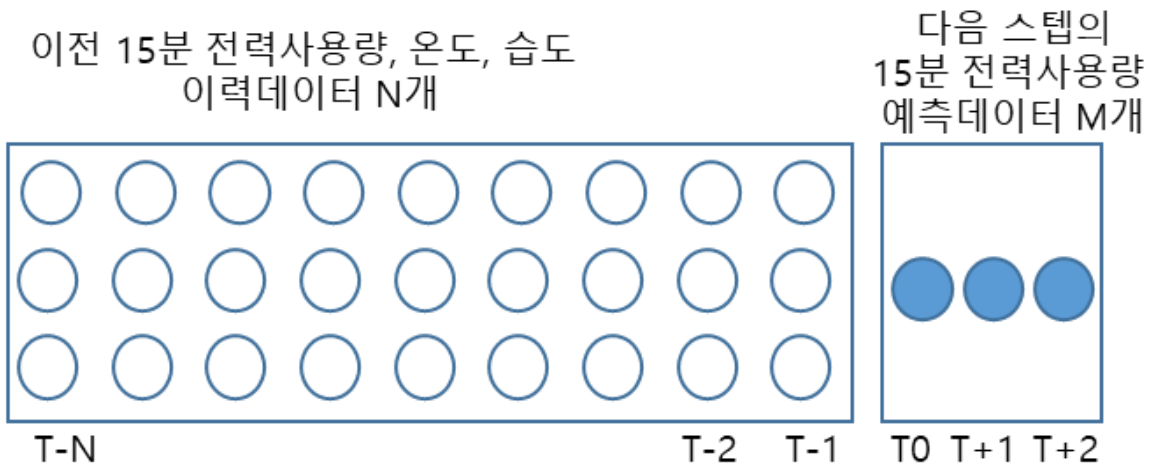
시퀀스 데이터 구성방법

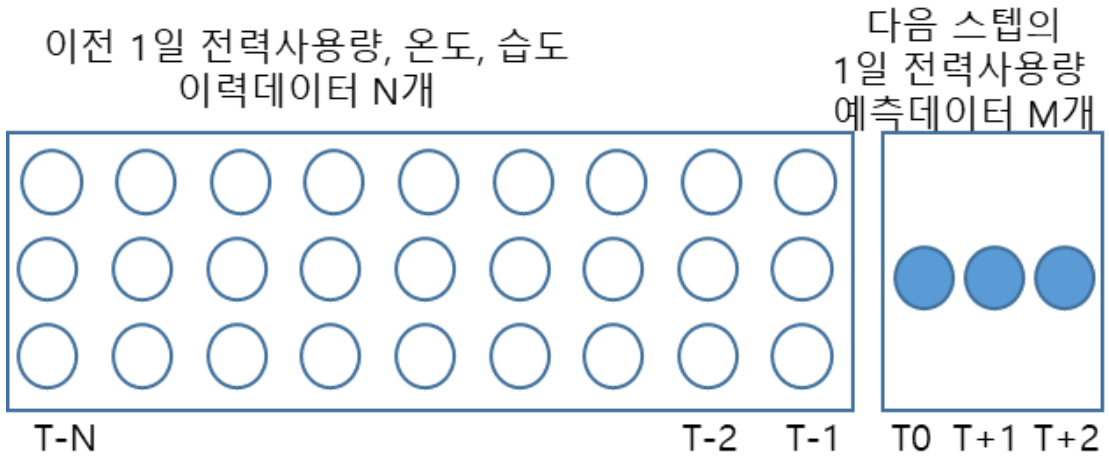
■ 싱글스텝





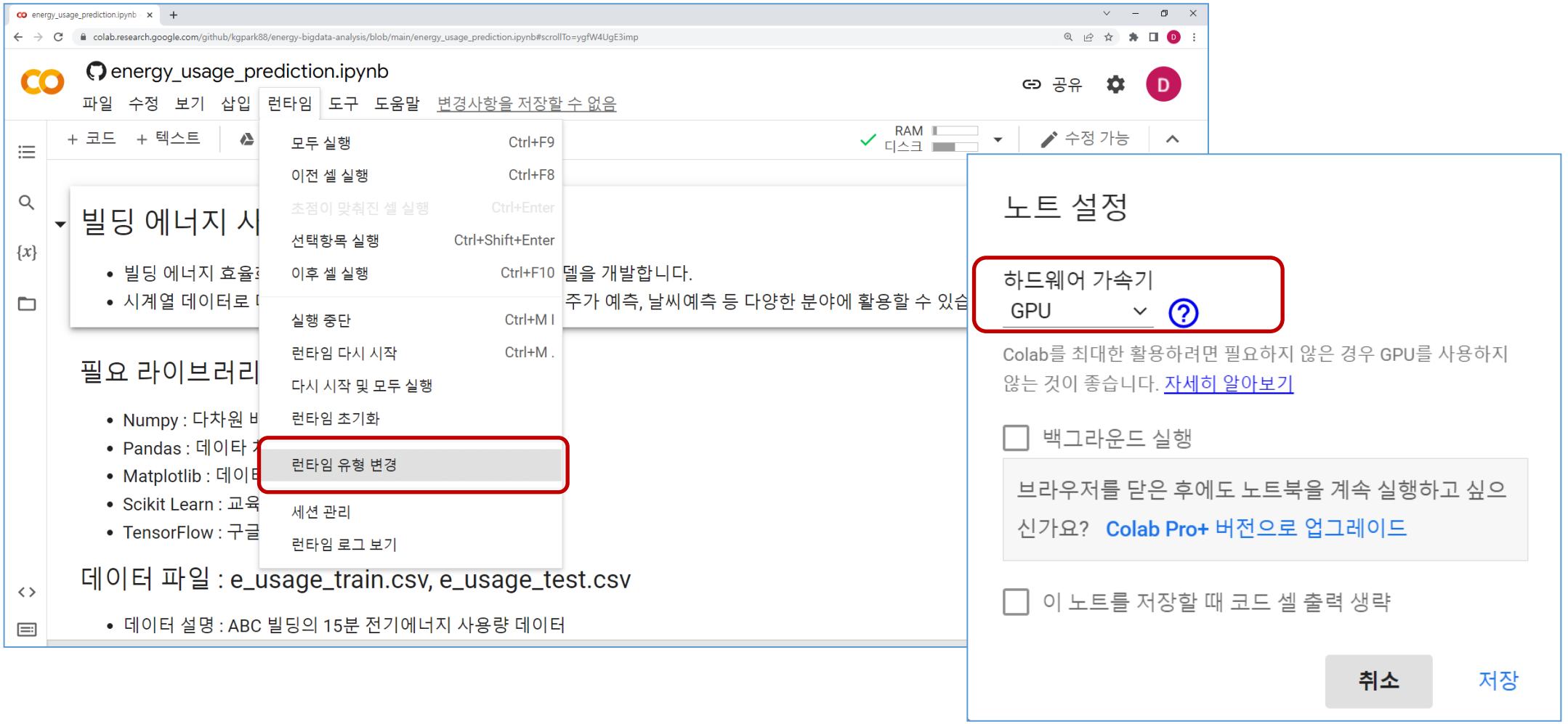
■ 멀티스텝





에너지 사용량 예측모델 개발

energy_usage_prediction.ipynb



에너지 사용량 예측모델 개발

STEP 1. 라이브러리 import

```
[1]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from sklearn.preprocessing import MinMaxScaler

     import tensorflow as tf
     from tensorflow.keras.models import Sequential, load_model
     from tensorflow.keras.layers import Dense, LSTM, Activation, Dropout
     from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

에너지 사용량 예측모델 개발

STEP 2. 시계열 데이터 처리

csv 파일에서 Train 데이터를 로드합니다.

```
[2]  df = pd.read_csv('e_usage_train.csv', header = 0, delimiter = ',')
```

데이터를 확인합니다.

```
[3]  df.head()
```

	b_name	daq_time	wday	day_type	hour	temp	rh	p_usage
0	ABC	2016-01-01 0:15	5	3	1	-2.5	99.0	229
1	ABC	2016-01-01 0:30	5	3	1	-2.5	99.0	231
2	ABC	2016-01-01 0:45	5	3	1	-2.5	99.0	231

에너지 사용량 예측모델 개발

데이터셋을 입력시퀀스데이터와 타깃데이터로 분리하는 함수입니다.

- 시계열 데이터를 시퀀스 데이터로 변환
- 입력데이터는 시퀀스이고, 출력은 고정크기의 벡터나 스칼라인 다대일(many-to-one) 구조로 데이터 변환

```
[7]  def split_multivariate_data(dataset, target, start_index, end_index, hist_data_size, target_size, step, single_step=False):
         data = []
         labels = []

         start_index = start_index + hist_data_size
         if end_index is None:
             end_index = len(dataset) - target_size

         for i in range(start_index, end_index):
             indices = range(i-hist_data_size, i, step)
             data.append(dataset[indices])

             if single_step:
                 labels.append(target[i+target_size])
             else:
                 labels.append(target[i:i+target_size])

         return np.array(data), np.array(labels)
```

에너지 사용량 예측모델 개발

입력시퀀스데이터, 타깃데이터, 예측데이터를 그래프에 출력하는 함수입니다.

```
[8]  def plot_series(series, y=None, y_pred=None, x_label="$t$", y_label="$x(t)$"):
         n_steps = len(series)
         plt.plot(series, ".-")
         if y is not None:
             plt.plot(n_steps, y, "bx", markersize=10)
         if y_pred is not None:
             plt.plot(n_steps, y_pred, "ro")
         plt.grid(True)
         if x_label:
             plt.xlabel(x_label, fontsize=16, rotation=90)
         if y_label:
             plt.ylabel(y_label, fontsize=16, rotation=0)
```

에너지 사용량 예측모델 개발

전력사용량, 온도, 상대습도를 입력데이터(Feature)로 사용합니다.

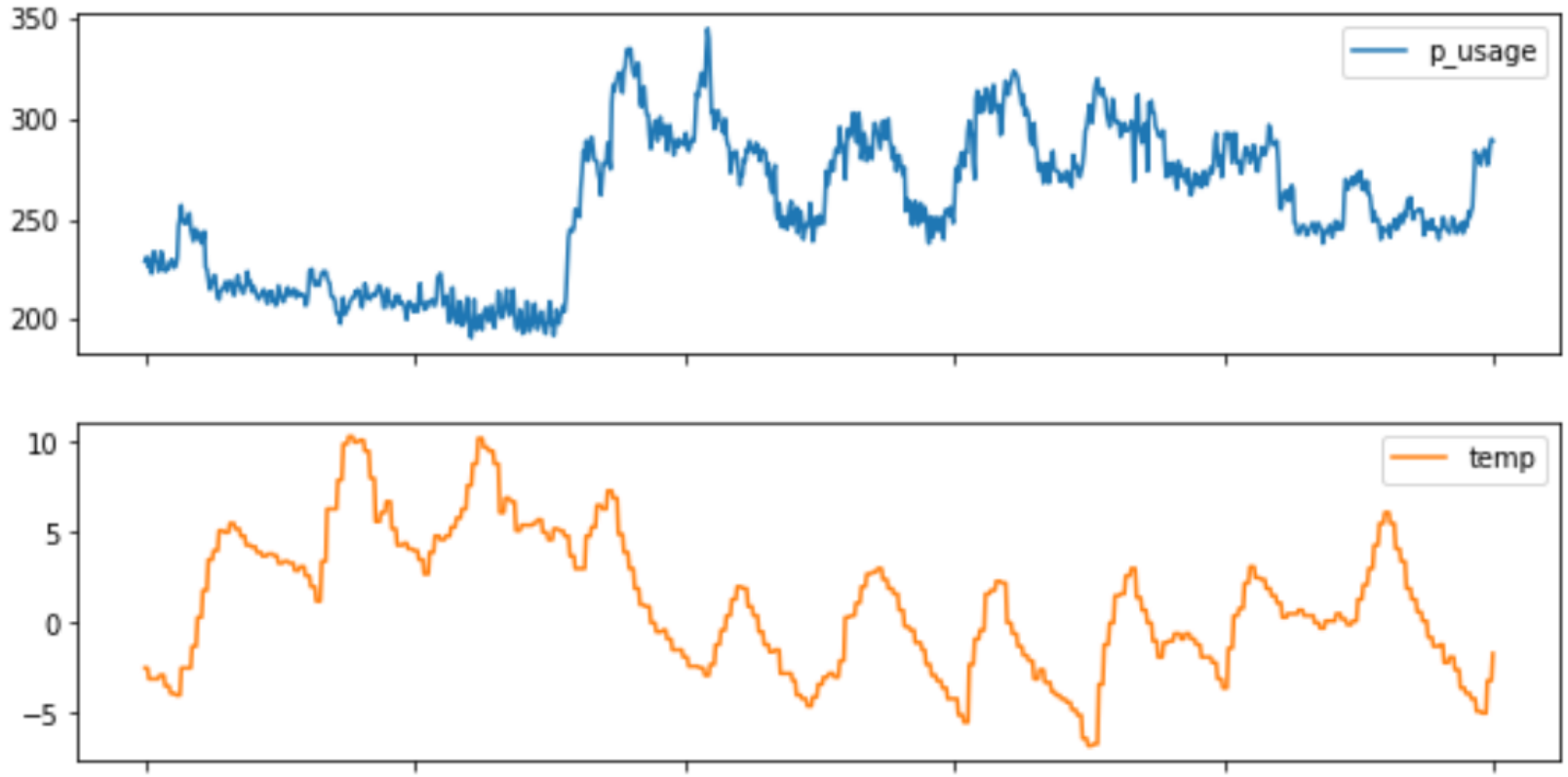
```
[9]  features = ['p_usage', 'temp', 'rh']
     features_data = df[features]
     features_data.index = df['daq_time']
     features_data.head()
```

daq_time	p_usage	temp	rh
2016-01-01 0:15	229	-2.5	99.0
2016-01-01 0:30	231	-2.5	99.0
2016-01-01 0:45	231	-2.5	99.0

에너지 사용량 예측모델 개발

시계열 데이터의 패턴을 확인합니다.

```
[10]  features_data[0:1000].plot(subplots=True, figsize=(10, 8))
```



에너지 사용량 예측모델 개발

데이터셋의 피처(Feature)를 정규화(Scaling)합니다.

```
[12]  TRAIN_SPLIT = 60000
      HISTORY_DATA_SIZE = 20
      FUTURE_TARGET = 0
      STEP = 1
```

```
[13]  scaler = MinMaxScaler()
      dataset  = scaler.fit_transform(dataset)
```

```
[14]  dataset
```

```
array([[0.43126177, 0.26245211, 0.98876404],
       [0.43502825, 0.26245211, 0.98876404],
       [0.43502825, 0.26245211, 0.98876404],
       ...,
```

에너지 사용량 예측모델 개발

Train 데이터셋과 Validation 데이터셋을 만듭니다.

```
[15]  X = dataset
      y = dataset[:,0]

      X_train, y_train = split_multivariate_data(X, y,
                                                 0, TRAIN_SPLIT,
                                                 HISTORY_DATA_SIZE, FUTURE_TARGET,
                                                 STEP, True)
      X_valid, y_valid = split_multivariate_data(X, y,
                                                 TRAIN_SPLIT, None,
                                                 HISTORY_DATA_SIZE, FUTURE_TARGET,
                                                 STEP, True)
```

에너지 사용량 예측모델 개발

split_multivariate_data 함수가 반환하는 내용입니다.

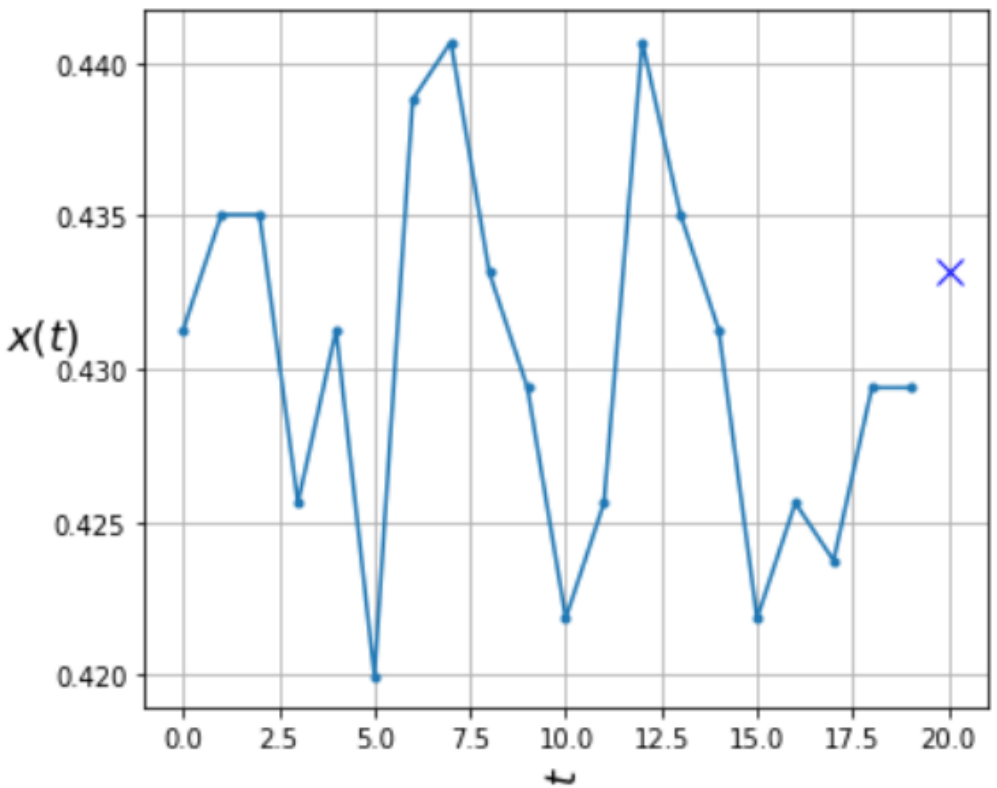
```
[16] print ('Single window of past history : {}'.format(X_train[0].shape))

     Single window of past history : (20, 3)
```

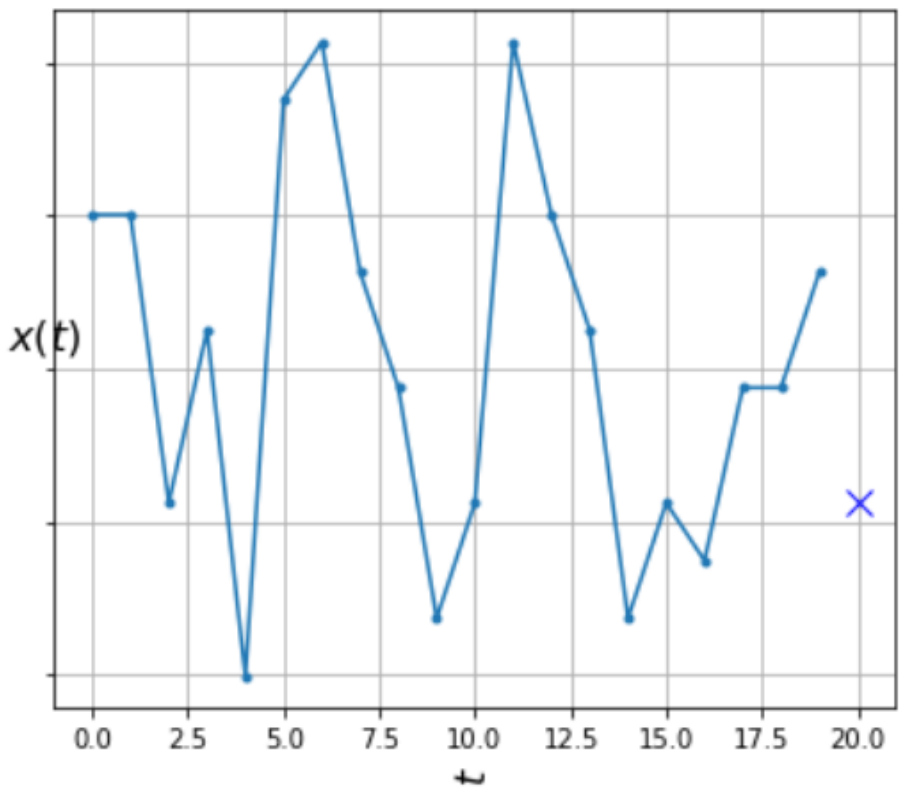
```
[17] print ('입력 데이터')
     print (X_train[0])
     print ('타겟 데이터')
     print (y_train[0])
```

```
입력 데이터
[[0.43126177 0.26245211 0.98876404]
 [0.43502825 0.26245211 0.98876404]
 [0.43502825 0.26245211 0.98876404]
 [0.42561205 0.25095785 1.        ]
 [0.43126177 0.25095785 1.        ]
 [0.41996234 0.25095785 1.        ]
 [0.43879473 0.25095785 1.        ]
 [0.44067797 0.25095785 1.        ]
 [0.43314501 0.25095785 1.        ]
 [0.42937853 0.25095785 1.        ]
 [0.42184557 0.25095785 1.        ]
 [0.42561205 0.25478927 1.        ]
 [0.44067797 0.25478927 1.        ]
 [0.43502825 0.25478927 1.        ]
 [0.43126177 0.25478927 1.        ]
 [0.42184557 0.24329502 1.        ]
 [0.42561205 0.24329502 1.        ]
 [0.42372881 0.24329502 1.        ]
 [0.42937853 0.24329502 1.        ]
 [0.42937853 0.23563218 1.        ]]
타겟 데이터
0.4331450094161958
```

에너지 사용량 예측모델 개발

```
[18]  cols = 3
      fig, axes = plt.subplots(nrows=1, ncols=cols, sharey=True, figsize=(20, 5))
      for i in range(cols):
          plt.sca(axes[i])
          plot_series(X_train[i, :, 0], y_train[i])
      plt.show()
```





에너지 사용량 예측모델 개발

STEP 3. 딥러닝 모델 구현

데이터가 순차데이터(Sequence Data)인 시계열(Time Series) 이므로
다양한 길이의 순차데이터 처리에 적합한 RNN 기반의 LSTM 모델을 사용합니다.

```
[19] HIDDEN_SIZE = 10
     DROP_OUT = 0.3

     model = Sequential()
     model.add(LSTM(HIDDEN_SIZE, input_shape=[20, 3], return_sequences=False))
     model.add(Dropout(DROP_OUT))
     model.add(Dense(1))
```

에너지 사용량 예측모델 개발

모델 구성 확인

```
[20]  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10)                560

 dropout (Dropout)           (None, 10)                0

 dense (Dense)               (None, 1)                 11

=================================================================
Total params: 571
Trainable params: 571
Non-trainable params: 0
_________________________________________________________________
```

에너지 사용량 예측모델 개발

모델 컴파일

```
[22]  model.compile(optimizer='adam', loss='mse')
```

모델 학습(Train) 조기종료, 체크포인트 설정

```
[23]  early_stop = EarlyStopping(monitor='val_loss', mode='min',
                                 verbose=1, patience=5)
      check_point = ModelCheckpoint('best_model.h5', verbose=1,
                                    monitor='val_loss', mode='min', save_best_only=True)
```

에너지 사용량 예측모델 개발

모델 학습(Train)

전력사용량 데이터는 일(Day) 단위 패턴이 있으므로 BATCH_SIZE를 96(15분*24시간=96)

```
[24]  EPOCHS = 50
      BATCH_SIZE=96

      history = model.fit(x=X_train, y=y_train,
                epochs=EPOCHS,
                batch_size=BATCH_SIZE,
                verbose=1,
                validation_data=(X_valid, y_valid),
                callbacks=[early_stop, check_point])
```

에너지 사용량 예측모델 개발

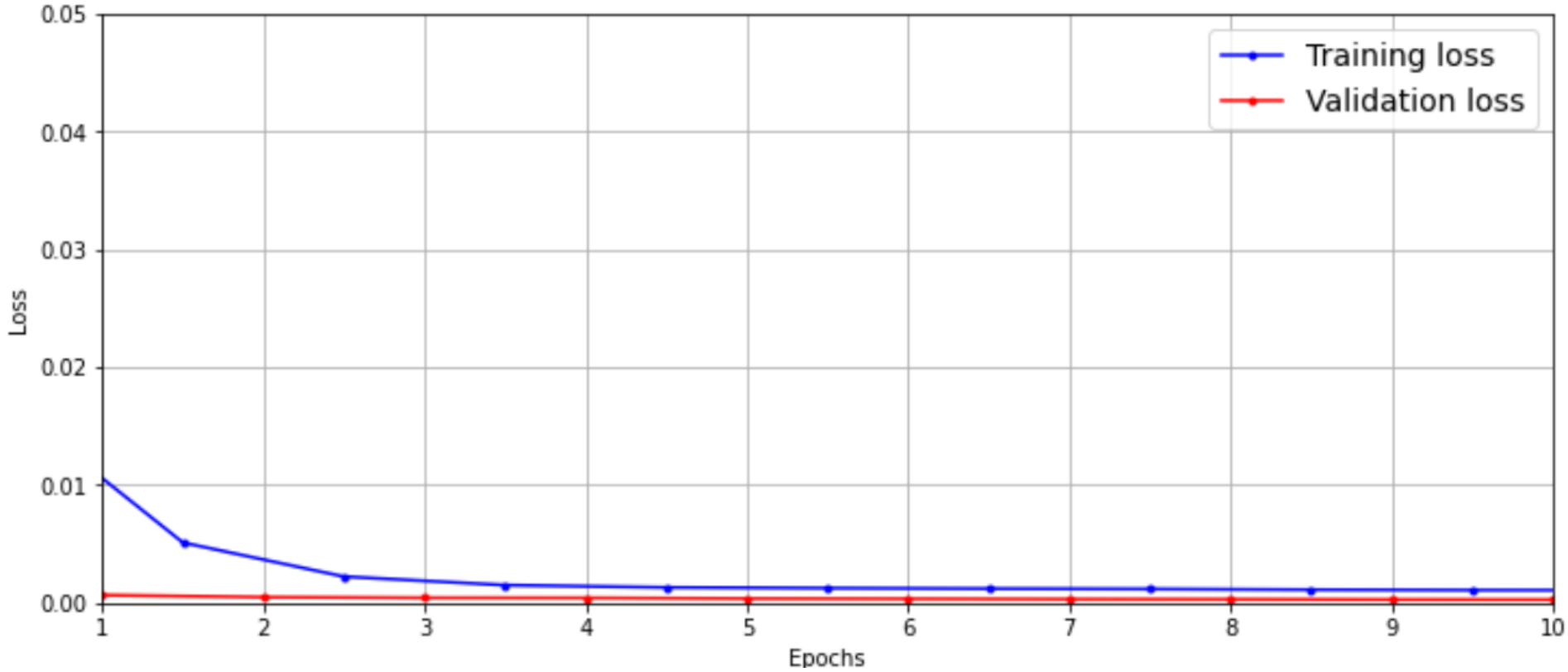
```
Epoch 1/50
616/625 [============================>.] - ETA: 0s - loss: 0.0161
Epoch 1: val_loss improved from inf to 0.00067, saving model to best_model.h5
625/625 [==============================] - 6s 5ms/step - loss: 0.0160 - val_loss: 6.6692e-04
Epoch 2/50
618/625 [============================>.] - ETA: 0s - loss: 0.0051
Epoch 2: val_loss improved from 0.00067 to 0.00049, saving model to best_model.h5
625/625 [==============================] - 3s 4ms/step - loss: 0.0051 - val_loss: 4.8703e-04
Epoch 3/50
612/625 [============================>.] - ETA: 0s - loss: 0.0022
Epoch 3: val_loss improved from 0.00049 to 0.00043, saving model to best_model.h5
625/625 [==============================] - 3s 4ms/step - loss: 0.0022 - val_loss: 4.2789e-04
Epoch 4/50
617/625 [============================>.] - ETA: 0s - loss: 0.0015
Epoch 4: val_loss improved from 0.00043 to 0.00042, saving model to best_model.h5
625/625 [==============================] - 3s 4ms/step - loss: 0.0015 - val_loss: 4.2062e-04
Epoch 5/50
616/625 [============================>.] - ETA: 0s - loss: 0.0013
Epoch 5: val_loss improved from 0.00042 to 0.00037, saving model to best_model.h5
```

에너지 사용량 예측모델 개발

모델의 Training Loss와 Validation Loss를 출력하는 함수입니다.

```
[25]  def plot_learning_curves(loss, val_loss):
          plt.figure(figsize=(12, 5))
          plt.plot(np.arange(len(loss)) + 0.5, loss, "b.-", label="Training loss")
          plt.plot(np.arange(len(val_loss)) + 1, val_loss, "r.-", label="Validation loss")
          plt.axis([1, 10, 0, 0.05])
          plt.legend(fontsize=14)
          plt.xlabel("Epochs")
          plt.ylabel("Loss")
          plt.grid(True)

      plot_learning_curves(history.history["loss"], history.history["val_loss"])
      plt.show()
```

에너지 사용량 예측모델 개발



모델 저장

```
[26]  model.save('my_model.h5')
```

에너지 사용량 예측모델 사용

STEP 4. 딥러닝 모델 사용

csv 파일에서 Test 데이터를 로드합니다.

```
[27]  df = pd.read_csv('e_usage_test.csv', header = 0, delimiter = ',')
```

데이터를 확인합니다.

```
[28]  df.head()
```

	b_name	daq_time	wday	day_type	hour	temp	rh	p_usage
0	ABC	2018-01-01 0:15	1	3	1	-1.8	43	283
1	ABC	2018-01-01 0:30	1	3	1	-1.8	43	279

에너지 사용량 예측모델 사용

전력사용량, 온도, 상대습도를 입력데이터(Feature)로 사용합니다.

```
[31]  features = ['p_usage', 'temp', 'rh']
      features_data = df[features]
      features_data.index = df['daq_time']
      dataset = features_data.values
```

데이터를 정규화(Scaling) 합니다.

```
[32]  scaled_dataset = scaler.transform(dataset)
```

```
[33]  X = scaled_dataset
      y = scaled_dataset[:,0]

      X_test, y_test = split_multivariate_data(X, y,
                                               0, None,
                                               HISTORY_DATA_SIZE, FUTURE_TARGET,
                                               STEP, True)
```

에너지 사용량 예측모델 사용

저장한 모델을 로드합니다.

```
[34]  model = load_model('best_model.h5')
```

테스트 데이터셋으로 모델 성능을 평가합니다.

```
[35]  model.evaluate(X_test, y_test)
```

```
1095/1095 [==============================] - 3s 2ms/step - loss: 5.8708e-04
0.0005870836903341115
```

에너지 사용량 예측모델 사용

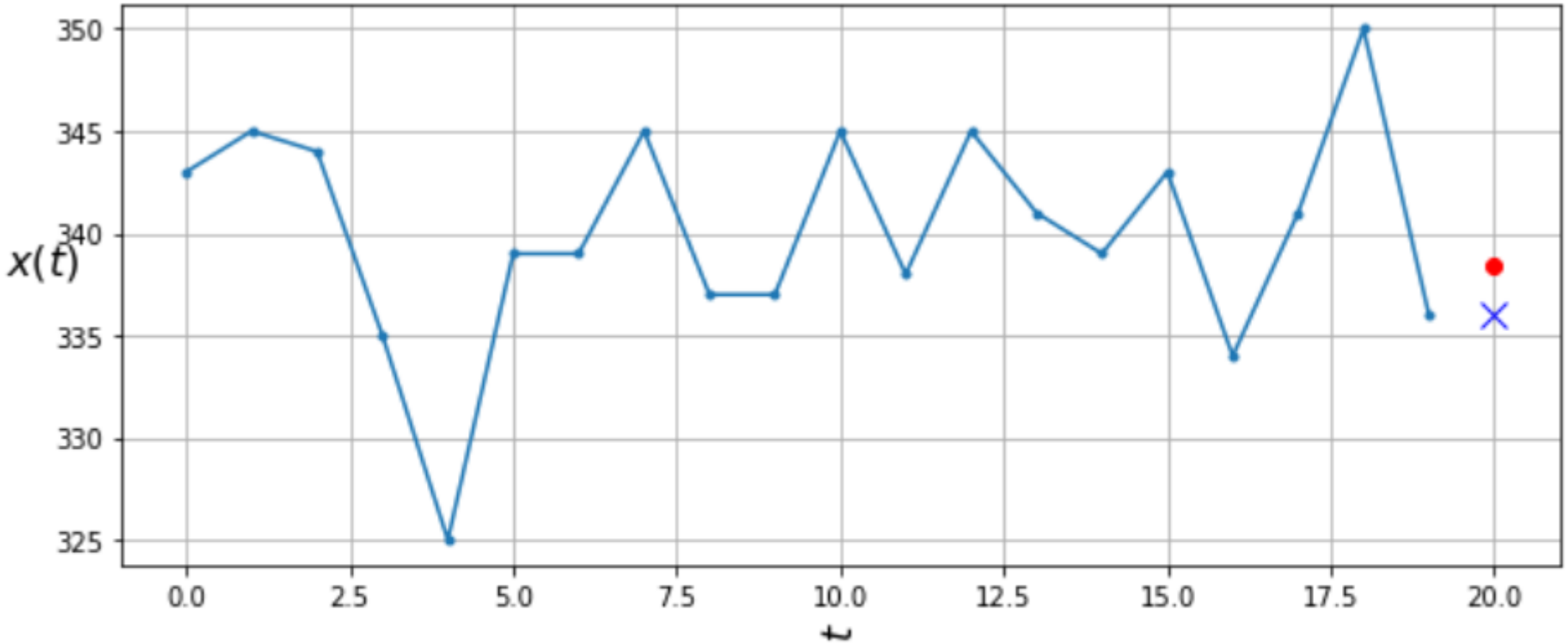
AI모델로 에너지 사용량을 예측합니다.

```
[36]  for TIME_STEP in range(1000,1110):
          p_usage_hist = scaler.inverse_transform(X_test[TIME_STEP])
          p_usage_hist = p_usage_hist[:, 0]

          p_usage_real  = dataset[TIME_STEP + HISTORY_DATA_SIZE][0]

          pred = model.predict(
              X_test[TIME_STEP].reshape(1, HISTORY_DATA_SIZE, -1))
          pred = pred[0][0]
          p_usage_pred = scaler.inverse_transform([[pred, 0, 0]])[0][0]

          error = abs(p_usage_pred - p_usage_real)
          error_rate = error/p_usage_real*100
```

에너지 사용량 예측모델 사용

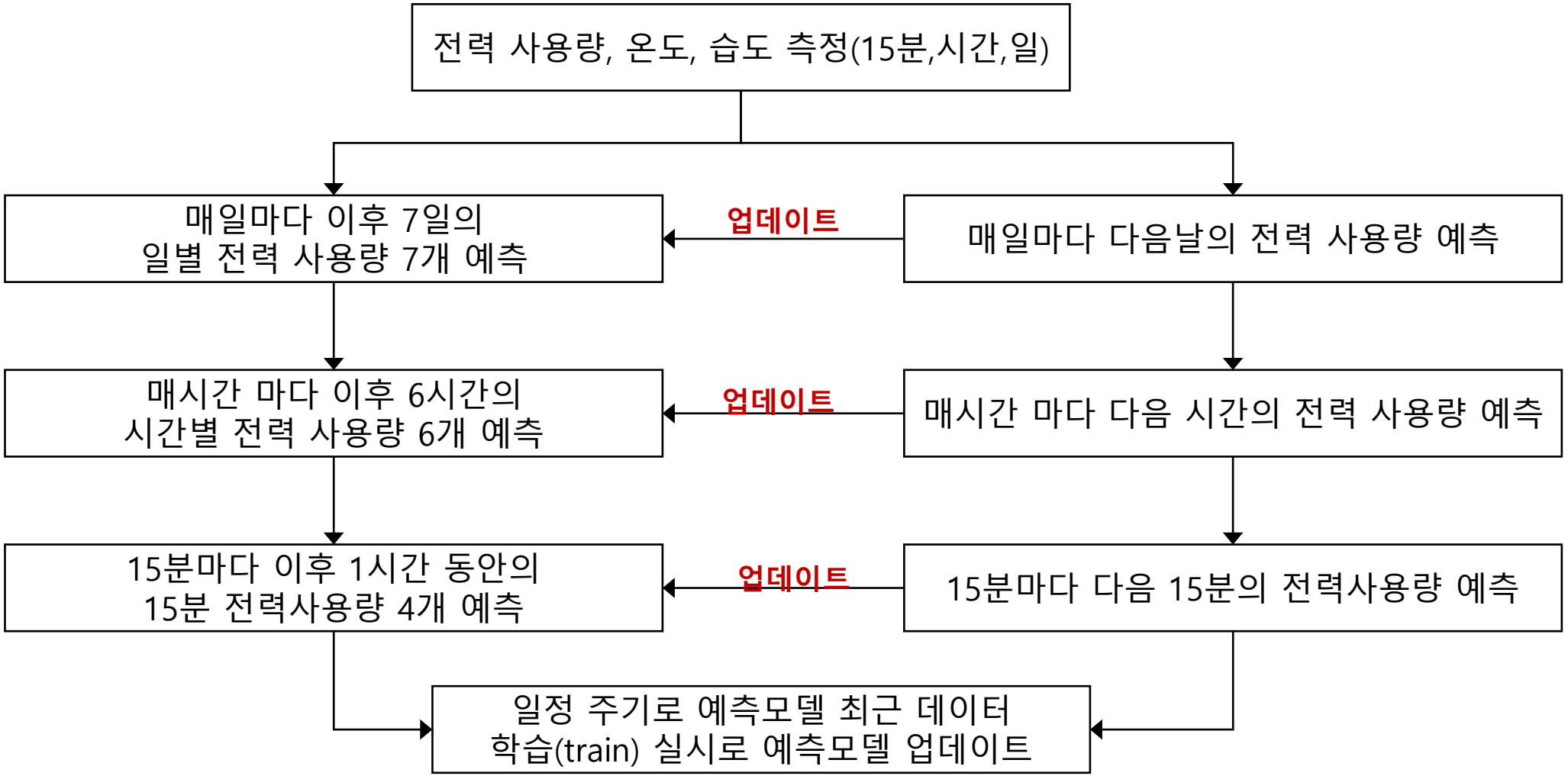
```
fig, axes = plt.subplots(nrows=1, ncols=1,
                         sharey=True, figsize=(10, 4))
plot_series(p_usage_hist, p_usage_real, p_usage_pred)
plt.show()

print(f'입력데이터 : {p_usage_hist}')
print(f'실제값 : {p_usage_real:.2f}')
print(f'예측값 : {p_usage_pred:.2f}')
print(f'오차 : {error:.2f}')
print(f'오차율 : {error_rate:.2f}%')
```

에너지 사용량 예측모델 사용



입력데이터 : [343. 345. 344. 335. 325. 339. 339. 345. 337. 337. 345. 338. 345. 341.
339. 343. 334. 341. 350. 336.]
실제값 : 336.00
예측값 : 338.46
오차 : 2.46
오차율 : 0.73%

에너지 사용량 예측모델 사용 전략



THANK YOU

kgpark88@gmail.com