TEAM PROJECT 01

태양광 발전량 예측 경진대회

_ DACON

[TEAM 빈SUN조]
_ 도준희 홍연정 이지수 한지민

CONTENTS

데이터 및 변수 소개

▶ 데이터 형태와 변수

	Day	Hour	Minute	DHI	DNI	WS	RH	T	TARGET
0	0	0	0	0	0	1.5	69.08	-12	0.0
1	0	0	30	0	0	1.5	69.06	-12	0.0
2	0	1	0	0	0	1.6	71.78	-12	0.0
3	0	1	30	0	0	1.6	71.75	-12	0.0
4	0	2	0	0	0	1.6	75.20	-12	0.0
...	...	...	...	...	...	...	...	...	...
52555	1094	21	30	0	0	2.4	70.70	-4	0.0
52556	1094	22	0	0	0	2.4	66.79	-4	0.0
52557	1094	22	30	0	0	2.2	66.78	-4	0.0
52558	1094	23	0	0	0	2.1	67.72	-4	0.0
52559	1094	23	30	0	0	2.1	67.70	-4	0.0

52560 rows × 9 columns

train

	Day	Hour	Minute	DHI	DNI	WS	RH	T	TARGET
0	0	0	0	0	0	2.7	34.42	0.0	0.0
1	0	0	30	0	0	2.7	34.17	0.1	0.0
2	0	1	0	0	0	2.7	34.23	0.2	0.0
3	0	1	30	0	0	2.7	33.99	0.3	0.0
4	0	2	0	0	0	2.8	33.97	0.4	0.0
...	...	...	...	...	...	...	...	...	...
331	6	21	30	0	0	3.6	56.09	-8.0	0.0
332	6	22	0	0	0	3.4	53.54	-8.2	0.0
333	6	22	30	0	0	3.4	53.89	-8.3	0.0
334	6	23	0	0	0	3.4	51.96	-8.4	0.0
335	6	23	30	0	0	3.4	51.96	-8.4	0.0

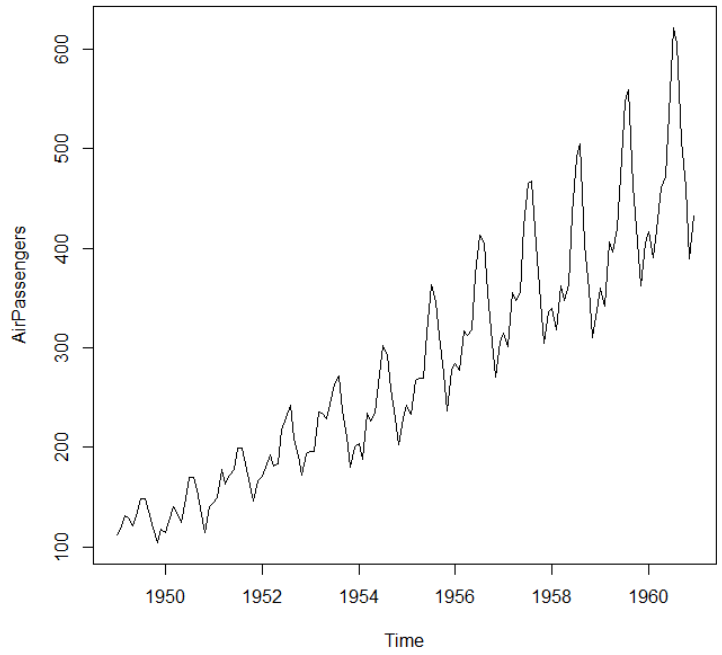
336 rows × 9 columns

test

변수	변수 소개
Hour	시간
Minute	분
DHI	수평면 산란일사량(Diffuse Horizontal Irradiance (W/m2))
DNI	직달일사량(Direct Normal Irradiance (W/m2))
WS	풍속(Wind Speed (m/s))
RH	상대습도(Relative Humidity (%))
T	기온(Temperature (Degree C))
Target	태양광 발전량 (kW)

시계열 데이터

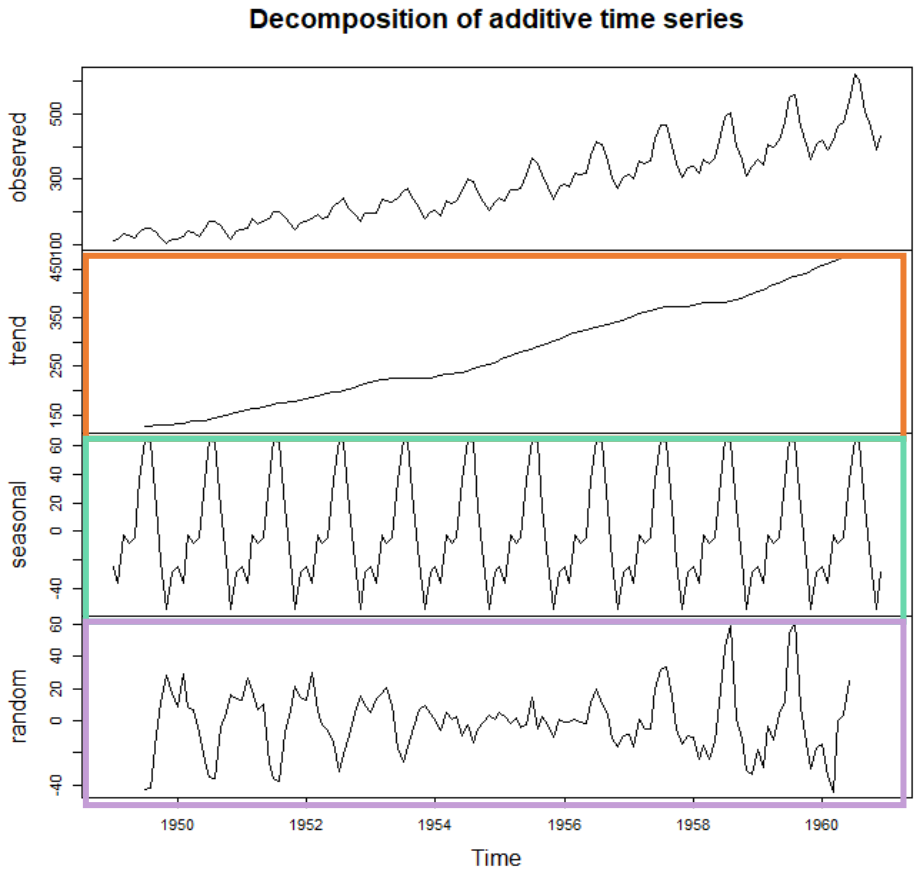
! 시계열 데이터란 !



- 관측치가 시간적 순서를 가진 데이터
- 변수간의 상관성(correation) 존재
- 독립항등분포(i.i.d), 연속(continous)하거나 불규칙적(irregular) 데이터는 다루지 않음

시계열 분석의 바탕이 되는 논리 : **과거의 특정 시간(구간)의 데이터를 통해 미래를 예측**

! 시계열 데이터란 !



- **추세(Trend)** ■
  경향. 데이터를 전체적으로 보았을 때 대략적인 증감 정보를 나타냄

- **계절성(Seasonality)** ■
  특정 기간마다 반복되는 패턴 확인 가능. 앞으로의 변화를 예측

- **랜덤(Random)** ■
  추세, 계절성 등으로 설명되지 않은 데이터
  예측오차 증가 문제 발생
  → 전처리를 통해 최대한 예측에 관여하지 않도록 해야함

▶ 데이터 전처리 과정

- Train, Test set 나누기

```
from sklearn.model_selection import train_test_split
x_train, x_valid, y_train, y_valid = train_test_split(train_x, train_y,
                                                      test_size=0.2)
x_train.shape, x_valid.shape
```

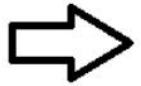
- 정규화 처리

각 컬럼 별 숫자 데이터의 **상대적인 크기 차이를 제거**

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
sc = MinMaxScaler() #정규화
test_data1=sc.fit_transform(test_data)
test_data=pd.DataFrame(test_data1,columns=test_data.columns)
```

	DHI	DNI	WS	RH	T
0	0	0	1.5	69.08	-12
1	0	0	1.5	69.06	-12
2	0	0	1.6	71.78	-12
3	0	0	1.6	71.75	-12
4	0	0	1.6	75.20	-12
...	...	...	...	...	...
52555	0	0	2.4	70.70	-4
52556	0	0	2.4	66.79	-4
52557	0	0	2.2	66.78	-4
52558	0	0	2.1	67.72	-4
52559	0	0	2.1	67.70	-4



	DHI	DNI	WS	RH	T
0	0.0	0.0	0.125000	0.665404	0.129630
1	0.0	0.0	0.125000	0.665188	0.129630
2	0.0	0.0	0.133333	0.694622	0.129630
3	0.0	0.0	0.133333	0.694297	0.129630
4	0.0	0.0	0.133333	0.731631	0.129630
...	...	...	...	...	...
52555	0.0	0.0	0.200000	0.682935	0.277778
52556	0.0	0.0	0.200000	0.640623	0.277778
52557	0.0	0.0	0.183333	0.640515	0.277778
52558	0.0	0.0	0.175000	0.650687	0.277778
52559	0.0	0.0	0.175000	0.650471	0.277778

▶ 슬라이딩 윈도우 (Sliding Window)

시계열 데이터를 활용한 예측 모델을 만들기 위한 방법 중 하나

- 윈도우 사이즈 만큼 데이터가 묶이고 스텝의 크기만큼 데이터가 이동
- 모델에 학습에 들어가게 되는 data set의 수 증가

➡ **더 정교한 예측을 통한 정확도 증가**

윈도우 : 빨간색 네모 스텝 : 옆으로 이동하는 칸의 수

```
def make_dataset(data, label, window_size):
feature_list = []
label_list = []
for i in range(len(data) - window_size-96):
    feature_list.append(np.array(data.iloc[i:i+window_size]))
    label_list.append(np.array(label.iloc[i+window_size :i+window_size+96 ]))
return np.array(feature_list), np.array(label_list)
```

>>> 과거 7일의 데이터로 미래 2일을 예측해야 함

윈도우 사이즈 : 336(7일* 48개/일), 스텝 크기 : 1 로 지정

변수 선택 과정

▶ 다중공선성 (Multicollinearity)

입력변수(독립변수)들 간의 상관관계가 존재해 회귀 계수의 분산을 크게 하기 때문에
회귀 분석 시 추정 회귀 계수를 믿을 수 없게 되는 문제 발생.

설명 변수들 사이에 유의한 상관관계가 존재하면 설명변수를 다른 설명변수와의 함수관계로 표현 가능

↓

회귀 계수의 분산 증가 및 회귀 계수 추정치 불안

↓

추정 회귀 계수를 믿을 수 없어 데이터 분석 시 부정적인 영향을 미침

변수 선택 과정

▶ **다중공선성** (Multicollinearity)

- VIF 수치 확인

→ $VIF_i \frac{1}{1-R_i^2}$ (R_i^2 : i 번째 독립변수의 결정계수) : i번 째 변수의 다중공선성의 정도

- 상관계수 확인

→ $r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)s_x s_y}$ (s_x, s_y : 각각 x, y의 표준편차)

VIF 수치가 10 이상 혹은 **상관계수가 0.8 이상**이면 다중공선성이 있다고 판단.

변수 선택 과정

▶ 다중공선성 (Multicollinearity)

- VIF 수치 확인

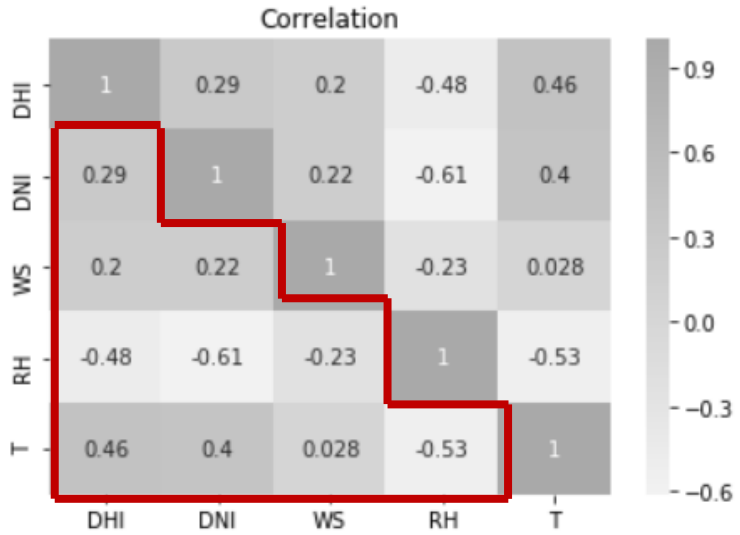
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif['Features'] = data.columns
vif['VIF Factor'] = [variance_inflation_factor(data.values, i)
                     for i in range(data.shape[1])]
vif
```

	Features	VIF Factor
0	DHI	1.982141
1	DNI	2.202730
2	WS	3.672702
3	RH	4.246125
4	T	6.909959

➡ VIF 수치가 모두 9이하

- 상관계수 확인

```
import seaborn as sns

data.corr()
cmap = sns.light_palette('darkgray', as_cmap=True)
sns.heatmap(data.corr(), annot=True, cmap=cmap)
plt.title("Correlation")
plt.show()
```



➡ 상관계수가 모두 0.7 이하

변수 선택 과정

▶ 회귀분석 – 변수선택법, 최소제곱추정법

회귀분석에 변수선택법과 최소제곱추정법을 이용해 각 변수가 종속변수인 TARGET 변수에 유의미한 것이 있는지 확인

- 변수선택법 : 변수가 여러 개일 때 최적의 변수 조합을 찾아내는 기법

- 최소제곱추정법 : 실제로 관측된 값과 가정된 기댓값의 편차 제곱 합이 최소가 되도록 하는 방법

✓ R_squared(결정계수)가 1에 가까울 수록 회귀식의 정확도 높음

✓ p- value(유의확률)이 유의수준(0.05)보다 작으면 '통계적으로 유의하다 ' 라고 표현

변수 선택 과정

▶ 회귀분석 – 변수선택법, 최소제곱추정법

- 변수선택법

	numb_features	RSS	R_squared	features
0	1	1.950206e+03	0.270676	(DHI,)
1	1	1.202729e+03	0.550212	(DNI,)
2	1	2.421336e+03	0.094487	(WS,)
3	1	1.440117e+03	0.461436	(RH,)
4	1	1.859436e+03	0.304622	(T,)
5	2	7.190989e+02	0.731077	(DHI, DNI)
6	2	1.764083e+03	0.340281	(DHI, WS)
7	2	1.104647e+03	0.586892	(DHI, RH)
8	2	1.473515e+03	0.448946	(DHI, T)
9	2	1.045667e+03	0.608949	(DNI, WS)
10	2	7.171009e+02	0.731824	(DNI, RH)
11	2	7.735259e+02	0.710722	(DNI, T)
12	2	1.284529e+03	0.519621	(WS, RH)
13	2	1.618542e+03	0.394709	(WS, T)
14	2	1.137882e+03	0.574463	(RH, T)
15	3	5.741148e+02	0.785297	(DHI, DNI, WS)
16	3	3.815486e+02	0.857311	(DHI, DNI, RH)
17	3	4.254369e+02	0.840898	(DHI, DNI, T)
18	3	9.521400e+02	0.643926	(DHI, WS, RH)
19	3	1.275614e+03	0.522955	(DHI, WS, T)
20	3	8.433639e+02	0.684605	(DHI, RH, T)
21	3	5.696482e+02	0.786967	(DNI, WS, RH)
22	3	6.124694e+02	0.770953	(DNI, WS, T)
23	3	4.269723e+02	0.840324	(DNI, RH, T)
24	3	9.829646e+02	0.632398	(WS, RH, T)
25	4	2.372193e+02	0.911286	(DHI, DNI, WS, RH)
26	4	2.858943e+02	0.893083	(DHI, DNI, WS, T)
27	4	1.337821e+02	0.949969	(DHI, DNI, RH, T)
28	4	6.965171e+02	0.739522	(DHI, WS, RH, T)
29	4	2.832205e+02	0.894083	(DNI, WS, RH, T)
30	5	1.074908e-24	1.000000	(DHI, DNI, WS, RH, T)

➡ 5개 변수일 때 결정계수 가장 높음

- 최소제곱추정법

OLS Regression Results

Dep. Variable:	TARGET	**R-squared (uncentered):**	0.931
Model:	OLS	**Adj. R-squared (uncentered):**	0.931
Method:	Least Squares	**F-statistic:**	1.428e+05
Date:	Tue, 18 May 2021	**Prob (F-statistic):**	0.00
Time:	22:47:03	**Log-Likelihood:**	56849.
No. Observations:	52560	**AIC:**	-1.137e+05
Df Residuals:	52555	**BIC:**	-1.136e+05
Df Model:	5		
Covariance Type:	nonrobust		

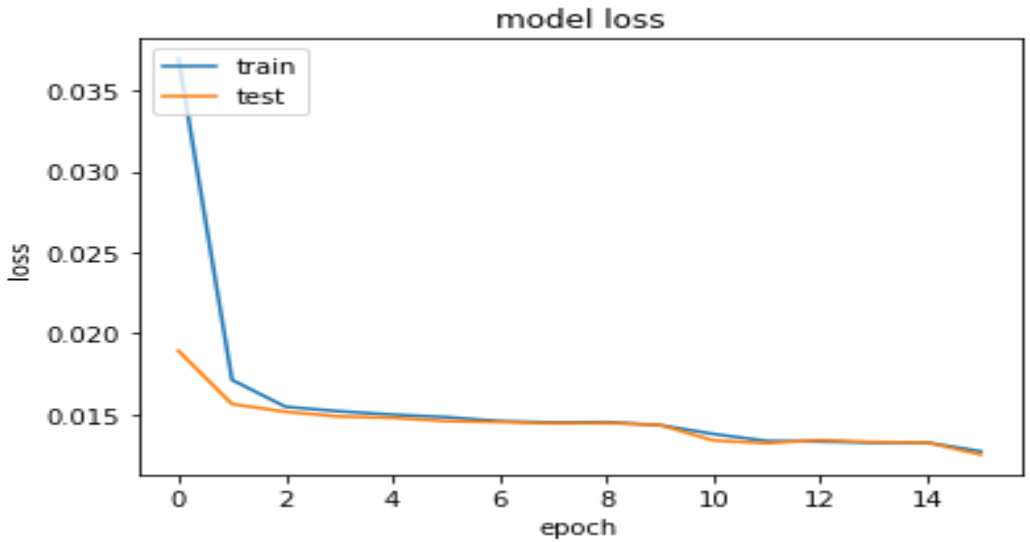
	coef	std err	t	P>|t|	[0.025	0.975]
DHI	0.5536	0.002	254.294	0.000	0.549	0.558
DNI	0.5072	0.001	379.776	0.000	0.505	0.510
WS	-0.0335	0.003	-11.546	0.000	-0.039	-0.028
RH	-0.0674	0.001	-53.323	0.000	-0.070	-0.065
T	0.0808	0.002	47.795	0.000	0.077	0.084

Omnibus:	6211.597	**Durbin-Watson:**	0.312
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	29641.663
Skew:	0.489	**Prob(JB):**	0.00
Kurtosis:	6.546	**Cond. No.**	7.07

➡ 유의확률이 유의수준보다 작으므로 유의함

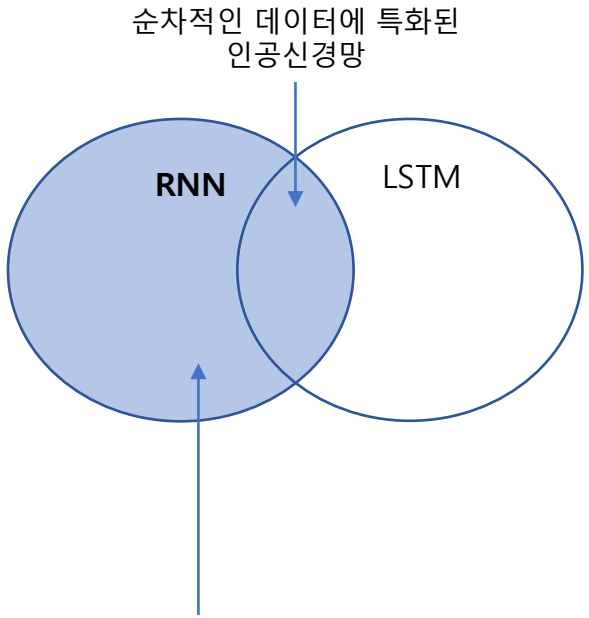
▶ 학습

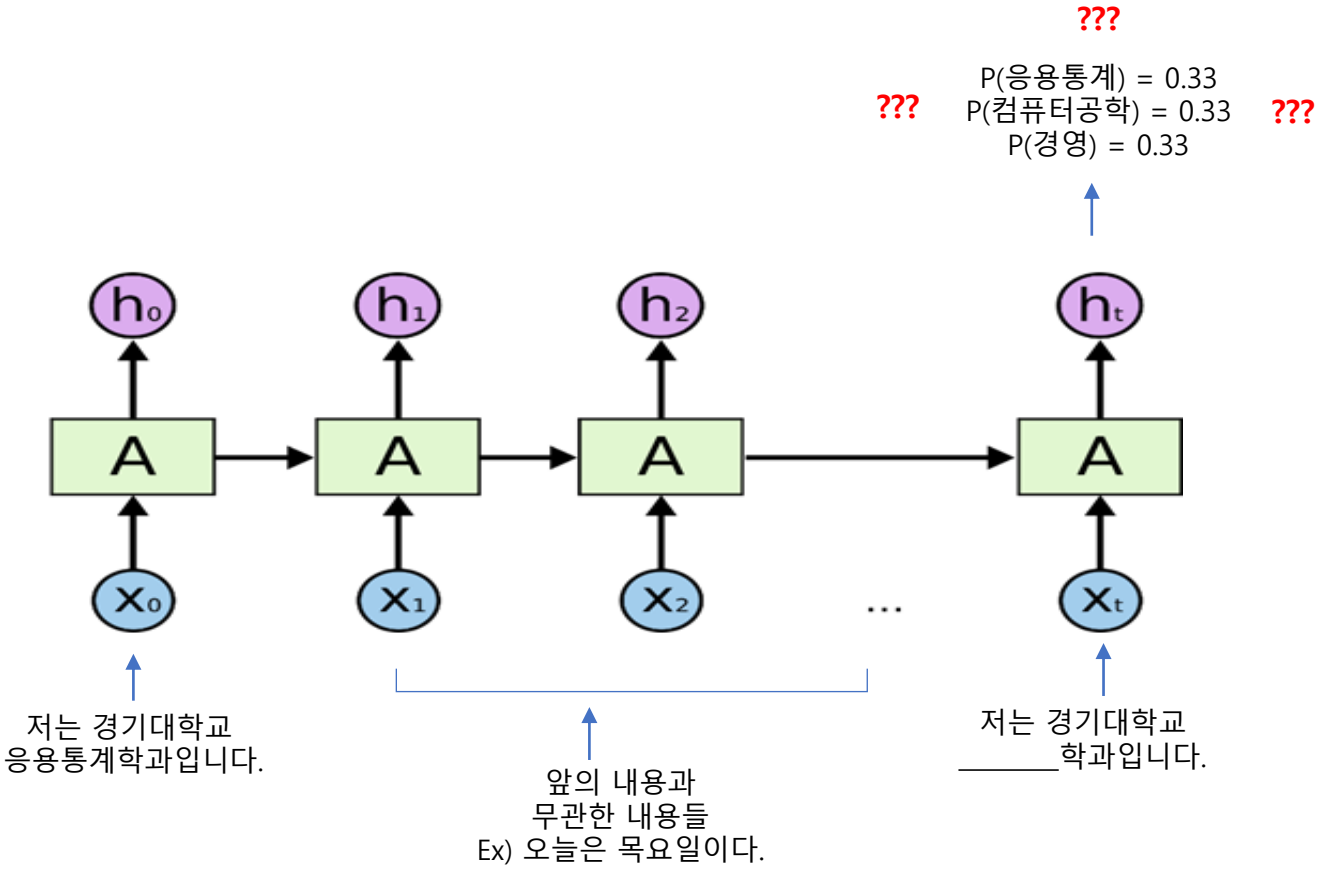
```
#모델 생성
model  = Sequential()
model.add(LSTM(48, activation='tanh',input_shape=(336,5)))
model.add(Dense(96 ,activation='relu'))
model.compile(loss='mse', optimizer='adam')
model.summary()

history = model.fit(x_train, y_train, batch_size=256, epochs=16, validation_data=(x_test, y_test))
```



- ▶ Lstm에서는 활성화 함수로 tanh를 사용했고 dense에서는 relu를 사용했다.
- ▶ 마지막 dense layer층은 2일간의 예측을 output값으로 출력해야 되기 때문에 96개의 층으로 구성
- ▶ 학습을 진행한 결과 낮은 편향과 낮은 분산을 보이며 잘 학습된 것을 볼 수 있다.

▶ 시계열 DATA 예측 모델 _ RNN

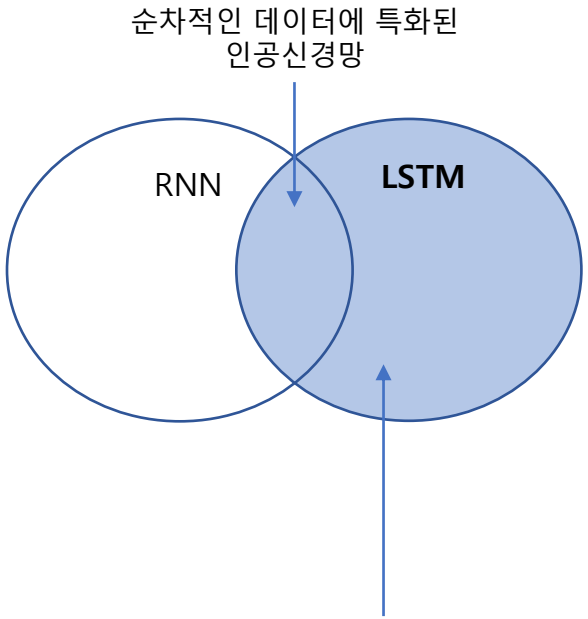


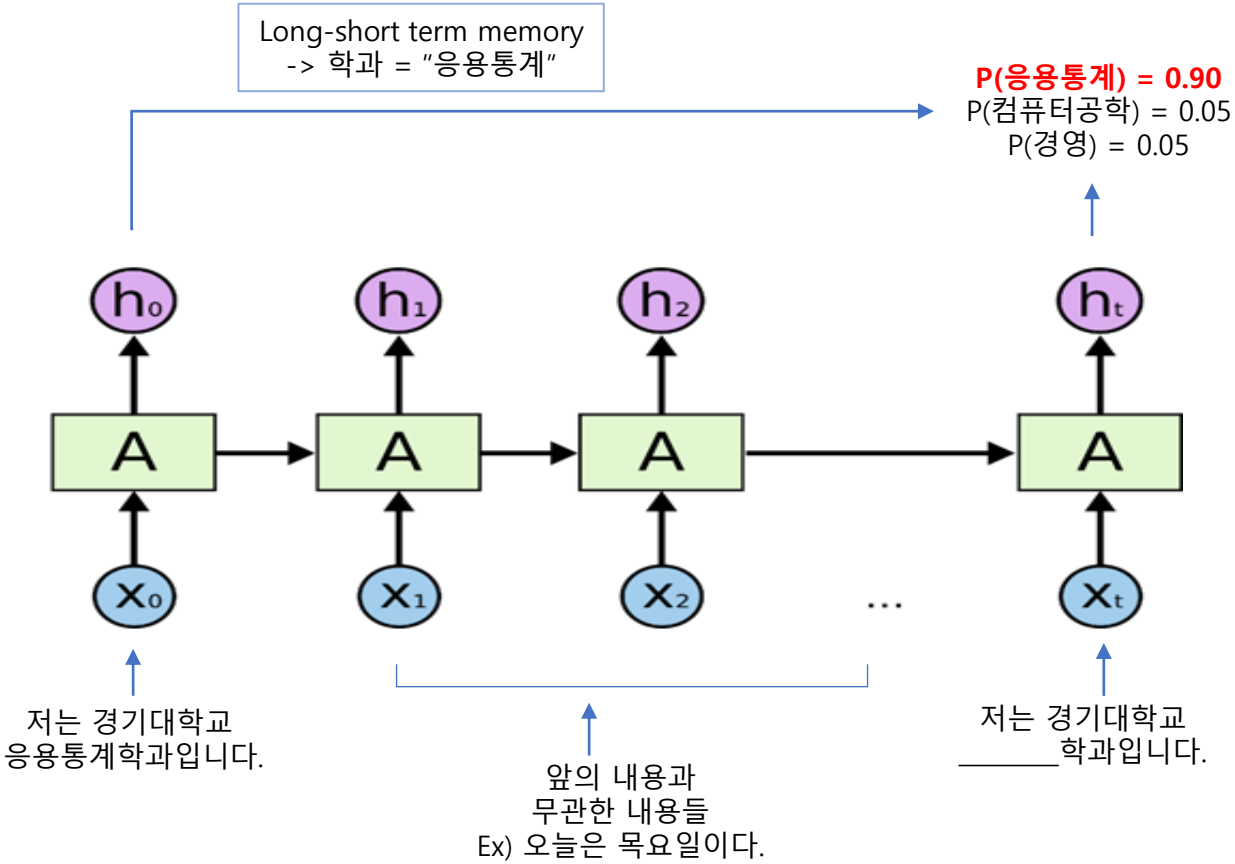


RNN (Recurrent Neural Network)

- 단기기억을 사용
- 바로 직전 타입스텝의 정보만을 다음 셀로 전달
- 긴 길이의 시퀀스로 인해 기울기 소실이나 폭발과 같은 문제 발생 가능

▶ 시계열 DATA 예측 모델 _ LSTM

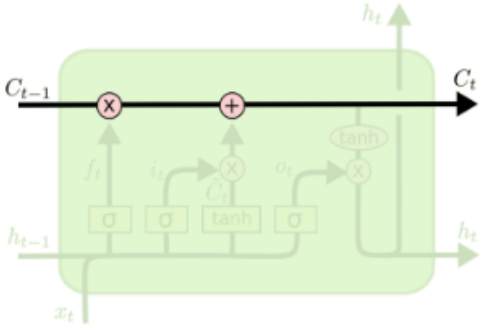




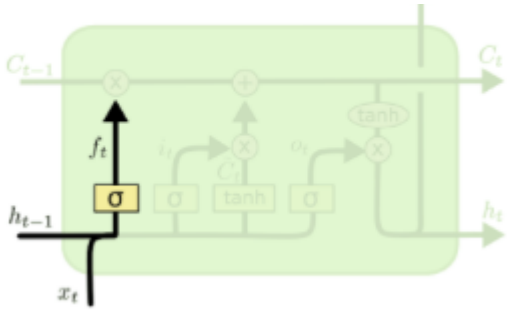
LSTM (Long-Short Term Memory)

- 장기기억 + 단기기억 모두 존재
- 오래된 내용에 패널티를 부여하는 방식

▶ 시계열 DATA 예측 모델 _ LSTM

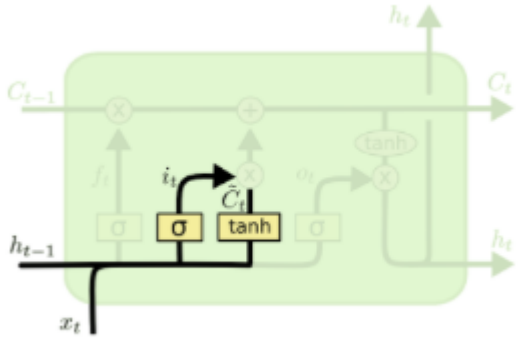


장기정보를 쭉 가져가는 역할로
컨베이어 벨트와 동일하다.



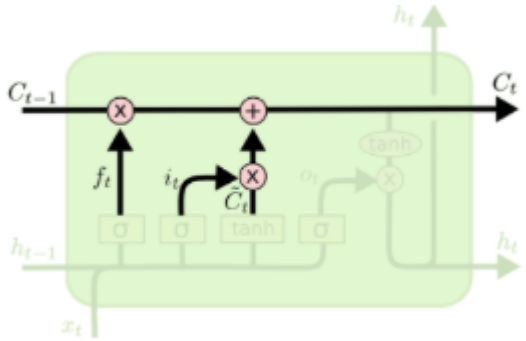
컨베이어 벨트에 어떠한 내용을 빨지,
유지할지 정하는 역할

f_t 는 시그모이드 함수로
0과 1의 값을 가진다.
0 – 정보 완전 제거
1 – 정보 완전 유지

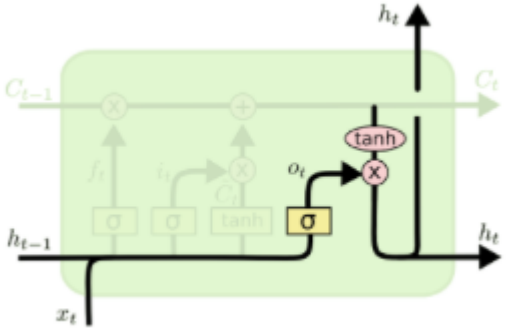


갱신과 관련된 단계로

i_t는 어떤 값을 갱신할지,
C_t는 더해질 수 있는 것들의
후보 값 벡터

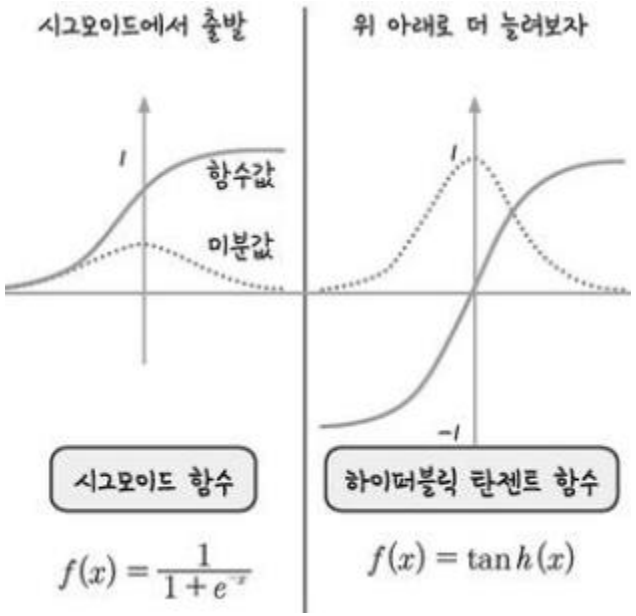


f_t 의 값은 곱하고,
i_t와C_t를 곱한 값은 더하여
컨베이어 벨트 위의 정보를
갱신한다.



컨베이어 벨트 위의 정보들 중
어떤 것을 출력할지 결정한다.

▶ 하이퍼 블릭 탄젠트 함수



하이퍼 블릭 탄젠트 함수

시그모이드 함수를 위아래로 더 늘린 값으로

시그모이드 함수는 음수의 결과값을 가지지 못하지만 하이퍼 블릭 탄젠트 함수는 가능하다.

정리

▶ 전처리

	Day	Hour	Minute	DHI	DNI	WS	RH	T	TARGET
0	0	0	0	0	0	1.5	69.08	-12	0.0
1	0	0	30	0	0	1.5	69.06	-12	0.0
2	0	1	0	0	0	1.6	71.78	-12	0.0
3	0	1	30	0	0	1.6	71.75	-12	0.0
4	0	2	0	0	0	1.6	75.20	-12	0.0
...	...	...	...	...	...	...	...	...	...
52555	1094	21	30	0	0	2.4	70.70	-4	0.0
52556	1094	22	0	0	0	2.4	66.79	-4	0.0
52557	1094	22	30	0	0	2.2	66.78	-4	0.0
52558	1094	23	0	0	0	2.1	67.72	-4	0.0
52559	1094	23	30	0	0	2.1	67.70	-4	0.0

52560 rows × 9 columns

	DHI	DNI	WS	RH	T
0	0.0	0.0	0.125000	0.665404	0.129630
1	0.0	0.0	0.125000	0.665188	0.129630
2	0.0	0.0	0.133333	0.694622	0.129630
3	0.0	0.0	0.133333	0.694297	0.129630
4	0.0	0.0	0.133333	0.731631	0.129630
...	...	...	...	...	...
52555	0.0	0.0	0.200000	0.682935	0.277778
52556	0.0	0.0	0.200000	0.640623	0.277778
52557	0.0	0.0	0.183333	0.640515	0.277778
52558	0.0	0.0	0.175000	0.650687	0.277778
52559	0.0	0.0	0.175000	0.650471	0.277778

52560 rows × 5 columns

	TARGET
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
52555	0.0
52556	0.0
52557	0.0
52558	0.0
52559	0.0

52560 rows × 1 columns

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

▶ 데이터 분할 후 최대 최소 정규화 진행

정리

▶ 전처리

	DHI	DNI	WS	RH	T
0	0.0	0.0	0.125000	0.665404	0.129630
1	0.0	0.0	0.125000	0.665188	0.129630
2	0.0	0.0	0.133333	0.694622	0.129630
3	0.0	0.0	0.133333	0.694297	0.129630
4	0.0	0.0	0.133333	0.731631	0.129630
...	...	...	...	...	...
52555	0.0	0.0	0.200000	0.682935	0.277778
52556	0.0	0.0	0.200000	0.640623	0.277778
52557	0.0	0.0	0.183333	0.640515	0.277778
52558	0.0	0.0	0.175000	0.650687	0.277778
52559	0.0	0.0	0.175000	0.650471	0.277778

52560 rows × 5 columns

336 X 5

	TARGET
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
52555	0.0
52556	0.0
52557	0.0
52558	0.0
52559	0.0

52560 rows × 1 columns

96 X 1

```
# train set, test set 분할
x_train, x_test, y_train, y_test = train_test_split(data_ar, target_ar, test_size=0.2, shuffle=True, random_state = 2021)
```

```
# train set, test set 분할 후 shape
x_train_shape = np.shape(x_train)
y_train_shape = np.shape(y_train)
x_test_shape = np.shape(x_test)
y_test_shape = np.shape(y_test)

print('x_train:',x_train_shape, 'y_train:',y_train_shape)
print('x_test:',x_test_shape, 'y_test:',y_test_shape)
```

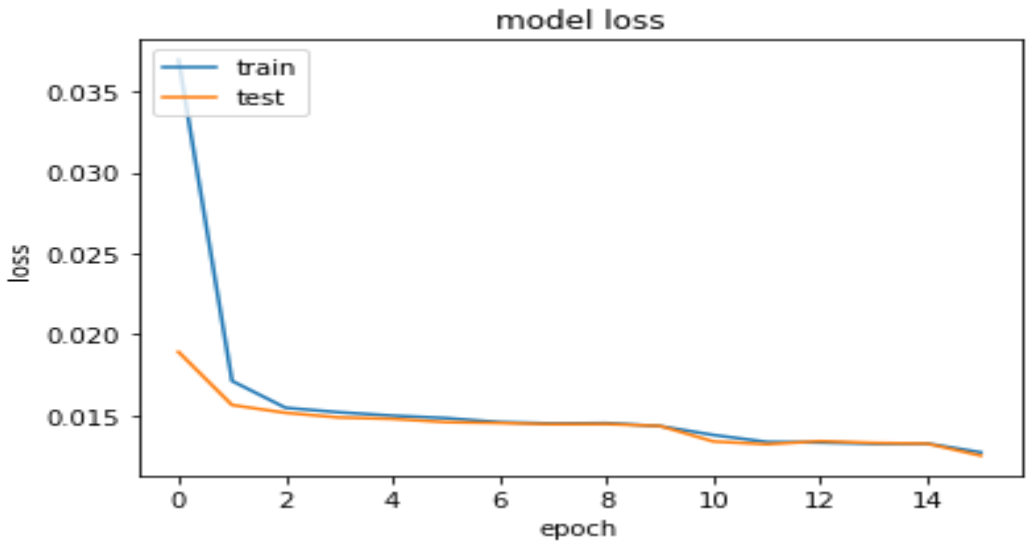
```
x_train: (41702, 336, 5) y_train: (41702, 96)
x_test: (10426, 336, 5) y_test: (10426, 96)
```

- ▶ Input data는 7일(48*7) output data는 2일 (48*2)으로 설정하여 슬라이딩 윈도우를 적용하여 데이터 생성
- ▶ Train set과 test set을 8:2의 비율로 랜덤하게 생성

정리

▶ 학습

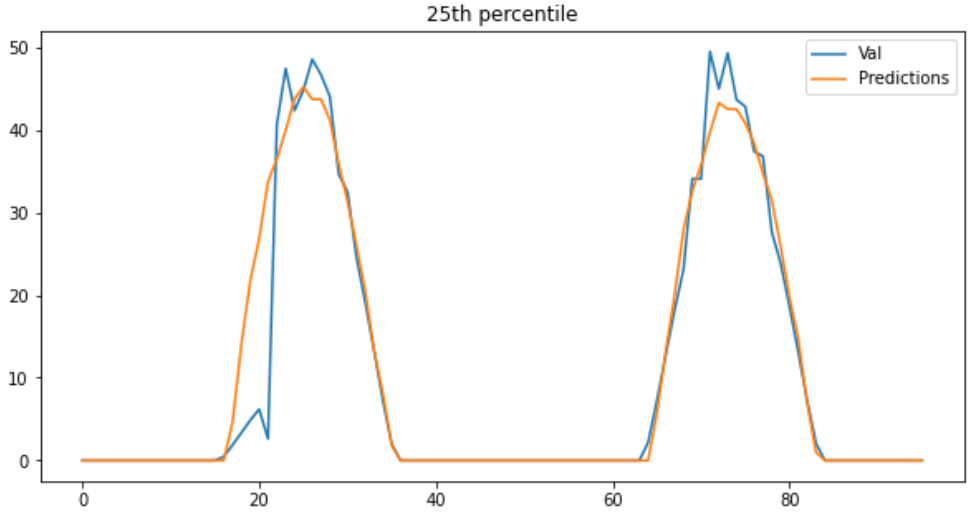
```
#모델 생성
model  = Sequential()
model.add(LSTM(48, activation='tanh',input_shape=(336,5)))
model.add(Dense(96 ,activation='relu'))
model.compile(loss='mse', optimizer='adam')
model.summary()

history = model.fit(x_train, y_train, batch_size=256, epochs=16, validation_data=(x_test, y_test))
```

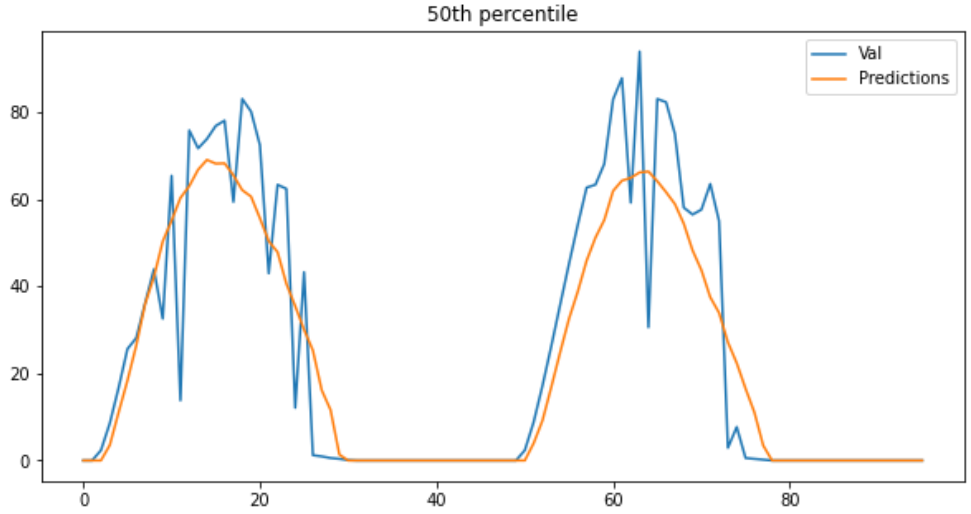


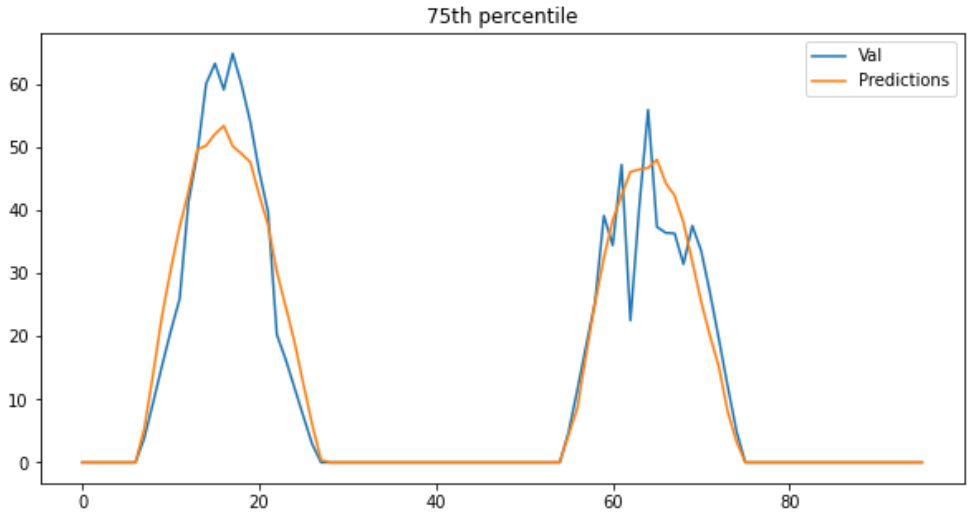
- ▶ Lstm에서는 활성화 함수로 tanh를 사용했고 dense에서는 relu를 사용했다.
- ▶ 마지막 dense layer층은 2일간의 예측을 output값으로 출력해야 되기 때문에 96개의 층으로 구성

예측 결과

▶ 결론







- Test set에서 25%, 50%, 70% 지점의 실제 값과 예측 값을 시각화한 그래프
- 파란색: 실제 값, 주황색: 예측 값

TEAM PROJECT 01

태양광 발전량 예측 경진대회

_ DACON

[TEAM 빈SUN조]
_ 도준희 홍연정 이지수 한지민