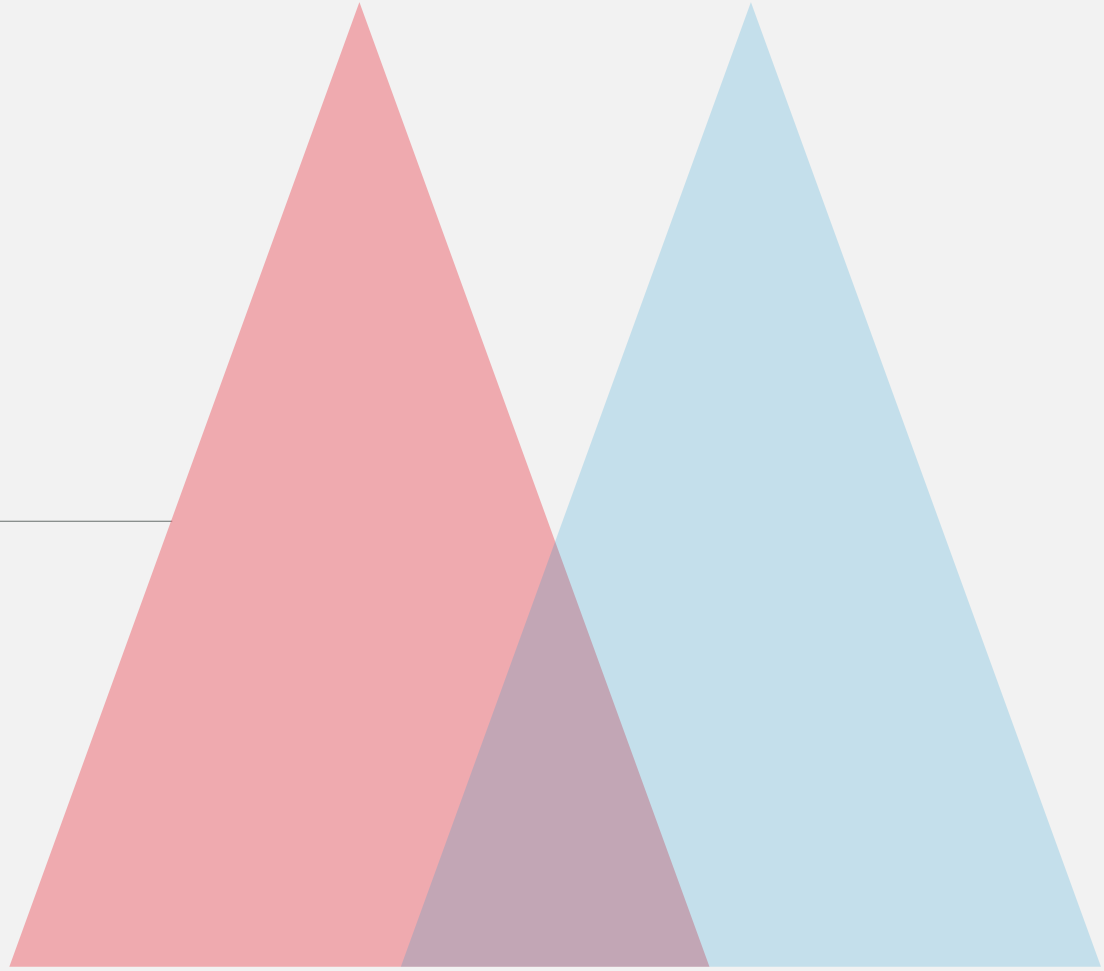
RNN을 이용한 감자 가격 예측

문찬호

Contents

001

프로젝트 소개



1

001 선정 이유

프로젝트 소개

한국경제 구독 PICK

'金자' 된 감자, 겨울 이상저온에 씨 말라…가격 더 뛴다

입력 2022.05.06. 오후 5:24 수정 2022.05.07. 오전 2:08 기사원문

한경제 기자 >

1 2

팜에어·한경 농산

도매가격 2900원

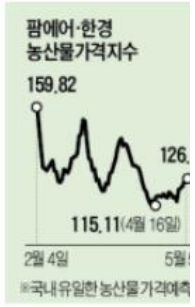
작황 부진·글로벌

감자 가격이 급등

한 저장감자와 올

6일 팜에어·한경 농

감자는 kg당 평균



SBS Biz 구독 PICK

패스트푸드점 감자튀김 품귀…코로나 여파 글로벌 물류난 이 원인

입력 2021.10.05. 오후 4:32 · 수정 2021.10.05. 오후 4:37 기사원문

류정훈 기자 >

2 8 가가

[감자튀김. 세계보건기구(WHO) 제공 (사진=연합뉴스)]

신종 코로나바이러스 감염증(코로나19) 사태로 인한 전 세계 물류대란이 국내 식품업계에영향을 미치고 있습니다.

5일 식품업계에 따르면 일부 패스트푸드점에서 감자튀김 품귀 현상이 벌어지고 있습니다.

- 코로나 시즌과 기후 이상으로 감자 수급이 점점 어려운 상황

- 머신러닝(시계열 데이터) 예측을 통해 이 동향을 예측

- RNN을 학습하고 머신 러닝이 감자 가격을 학습해서 예측할 수 있는 지 알아보는 프로젝트

- 활성화 함수 등 여러가지를 시도하면서 가장 예측이 잘 맞는 것을 학습

1

001 DataSet

프로젝트 소개

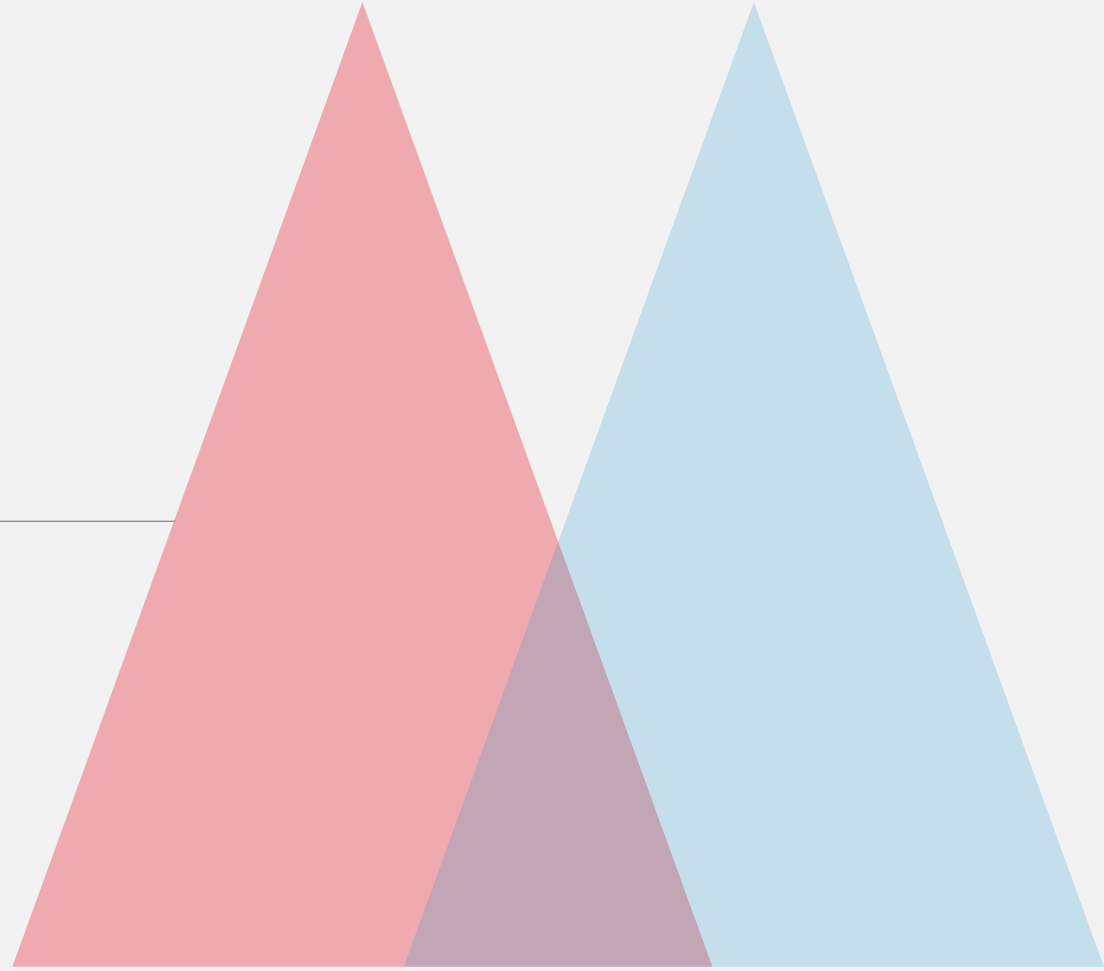
- KAMIS 농산물 유통정보의 데이터셋을 활용

- 1996년 1월부터 2021년 12월까지
총 312 개월의 감자 가격의 시계열 데이터를 활용

- https://www.kamis.or.kr/customer/main/main.do

002

RNN



2

001 시계열 데이터

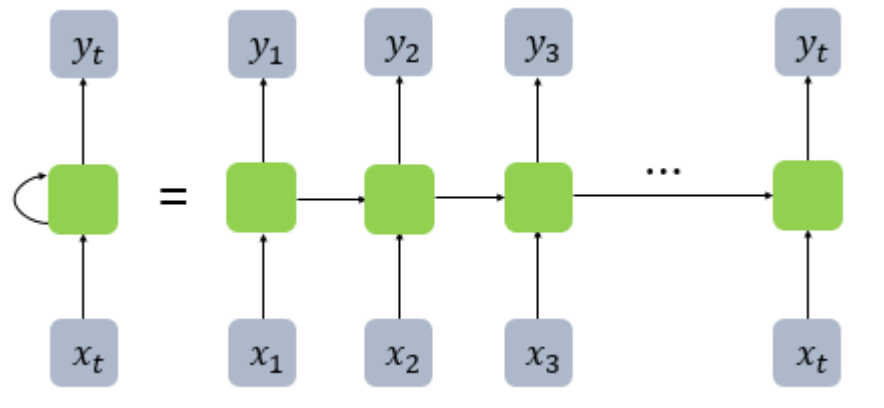
RNN

시계열 데이터를 학습하는 딥 러닝 기술

기준 시점 (t)와 다음 시점(t +1)을 네트워크 연결

AI 번역, 음성 인식, 주가 예측 등에 활용되는 기술

RNN

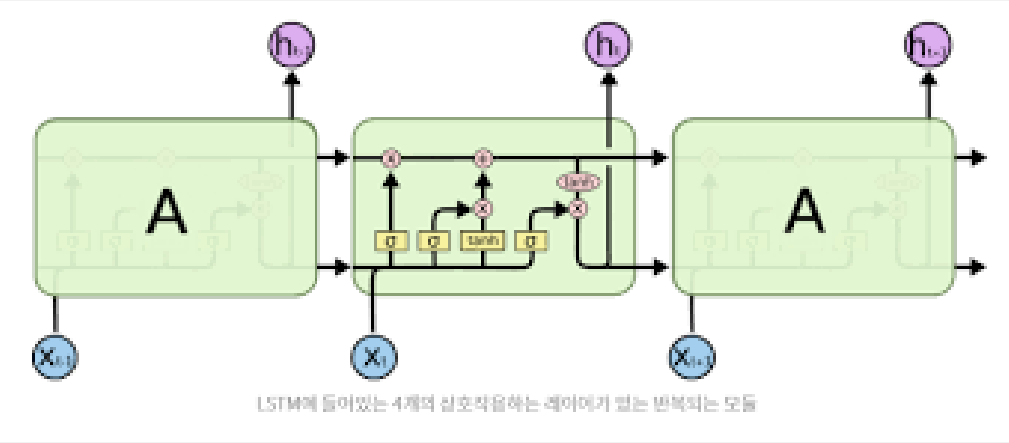


전 시점의 데이터를 다음 시점으로 넘겨 사용(unfold)

기울기 유실(vanishing gradient) 문제가 있음
(오래전 데이터의 영향력이 줄어드는 문제)

가까이 있는 계층이 더 강한 영향력을 발휘함

RNN



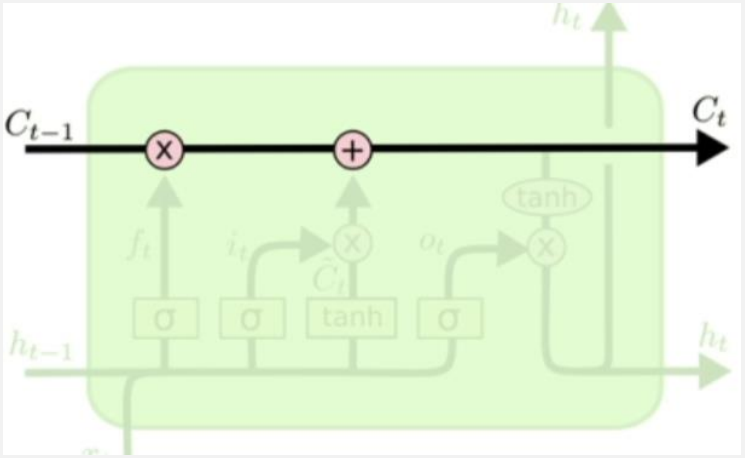
LSTM에 들어있는 4개의 상호작용하는 레이어가 있는 반복되는 모듈

LSTM

RNN의 기울기 유실 문제를 해결

2가지 종류의 정보를 다음 단에 전달

RNN



LSTM cellState

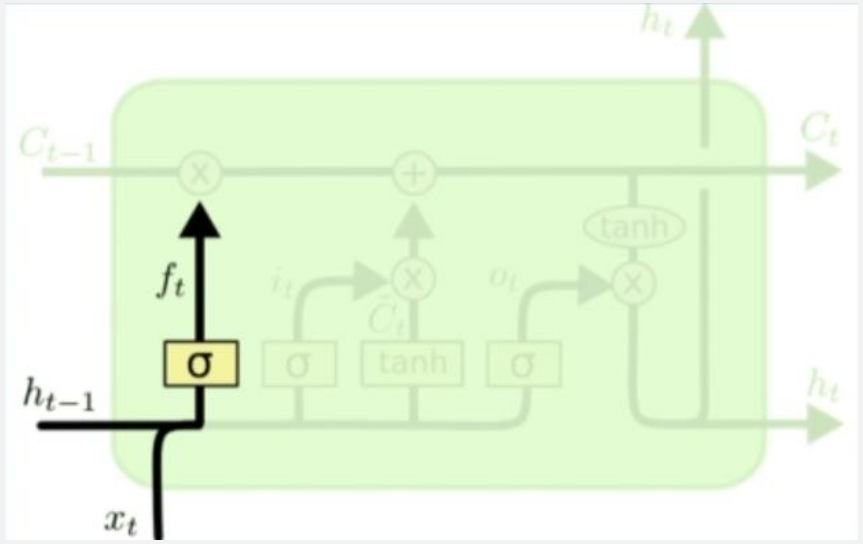
위쪽 부분을 나타냄

하나의 컨베이어 벨트처럼 전체 체인을 통과

- 현 단계 정보를 수정해서 계속 다음 단계에 전달
- 게이트를 활용해서 정보를 더하고 제거 하는데 사용
- CellState Line 에는 활성화 함수가 존재하지 않음

-> 기울기 유실 문제에서 벗어나는 이유

RNN



LSTM gating

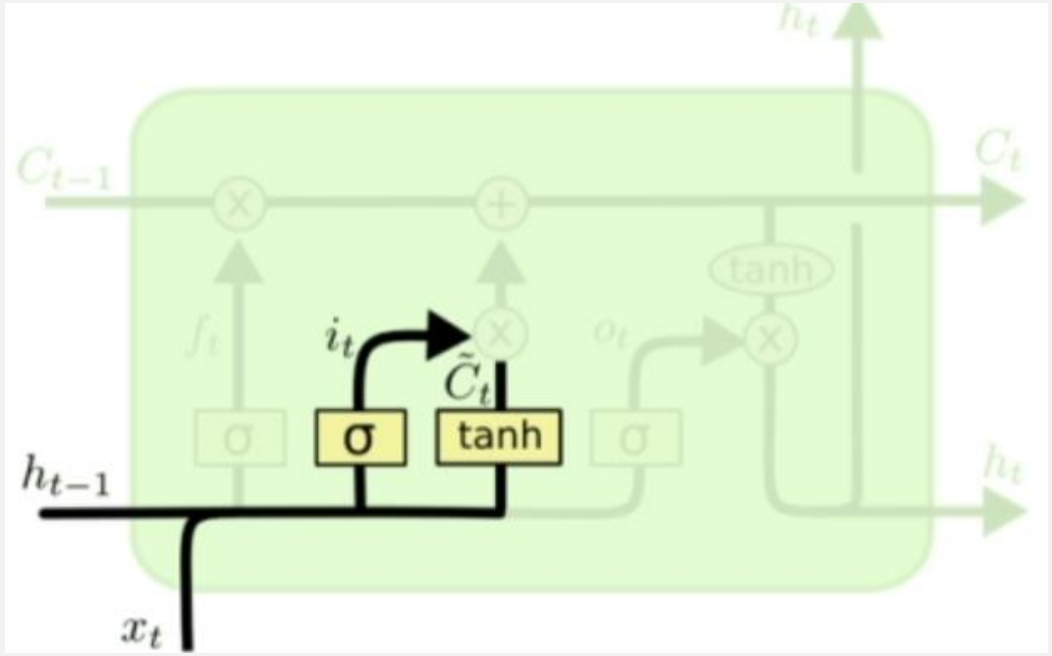
Cell State 라인의 추가/삭제 양을 조절

게이트는 총 4가지로 구성 됨

1. 망각 Gate
 - 전 단계에서 오는 정보를 얼마나 잊어버리나 결정 [0, 1]

 - 전 단계 출력값 (h_{t-1})과 현 단계 입력값 (X_t)를 합성 후 시그모이드 활성화를 거쳐 CellState에 올림

 - 두 숫자를 적절한 비율로 곱하고 편향치를 더하는 방식



LSTM gating

Cell State 라인의 추가/삭제 양을 조절

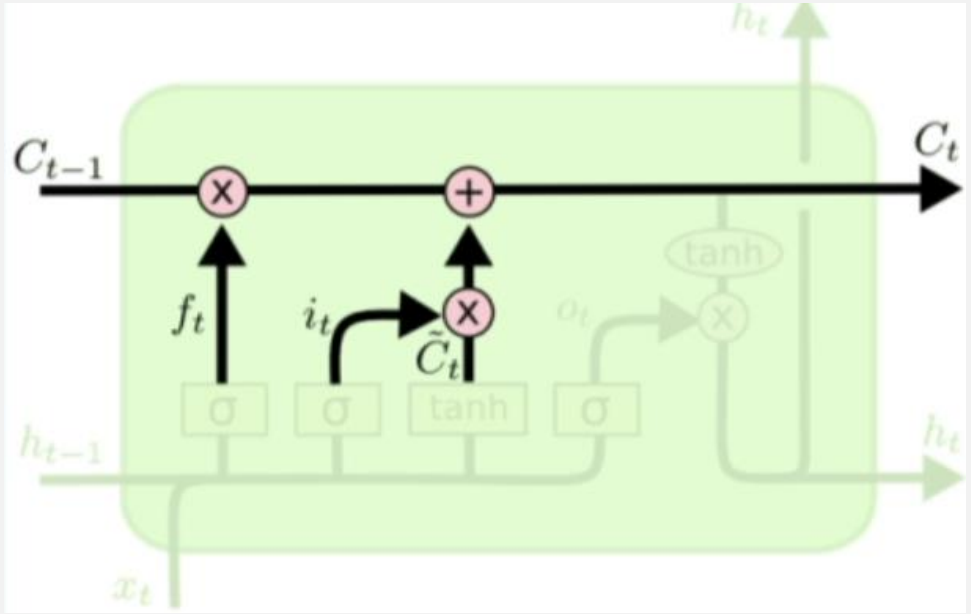
게이트는 총 4가지로 구성 됨

2. 새 정보 Gate
- 새로 들어오는 입력 값이 얼마나 받을까 결정
[-1, 1]

- 전 단계 출력값 (h_{t-1})과 현 단계 입력값 (X_t)를 합성 후 시그모이드 활성화를 거침 (i_t)

- 전 단계 출력값 (h_{t-1})과 현 단계 입력값 (X_t)를 합성 후 Tanh 활성화를 거침

- 이 두 과정을 합성해서 Cell State에 올리는 과정



LSTM gating

Cell State 라인의 추가/삭제 양을 조절

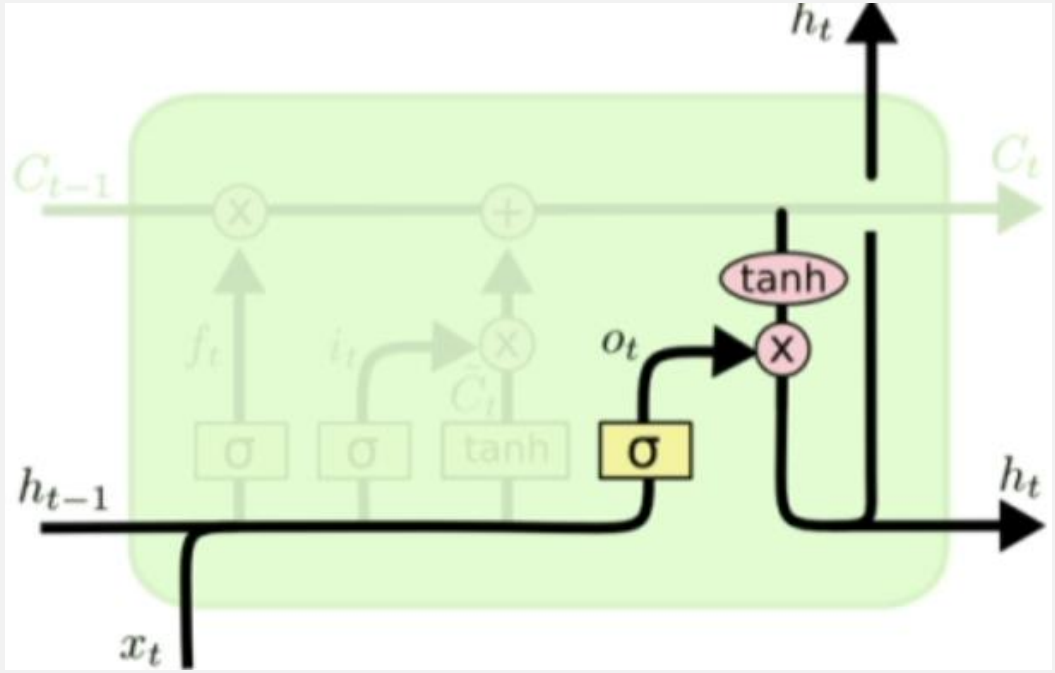
게이트는 총 4가지로 구성 됨

3. Cell State 출력 Gate
 - 앞서 계산한 망각 Gate, 새 정보 Gate를 합성해서 다음 단으로 값을 전달 함

 - 활성화 함수가 사용되지 않음

 - 망각과 새 정보 게이트를 합성할 지 결정하는 가중치도 존재하지 않음

RNN



LSTM gating

Cell State 라인의 추가/삭제 양을 조절

게이트는 총 4가지로 구성 됨

4. 현재 단 출력 Gate
- 현재 단의 출력을 결정 (망각 게이트와 방식은 동일)

- 세번째 Gate의 Cell State 출력 값을 활성화 진행

- 최종적으로 이 두가지를 합성한 것이 현재 단 출력 Gate

2

003 DataSet

RNN

- KAMIS 농산물 유통정보의 데이터셋을 활용

- 1996년 1월부터 2022년 1월까지
총 313 개월의 감자 가격의 시계열 데이터를 활용

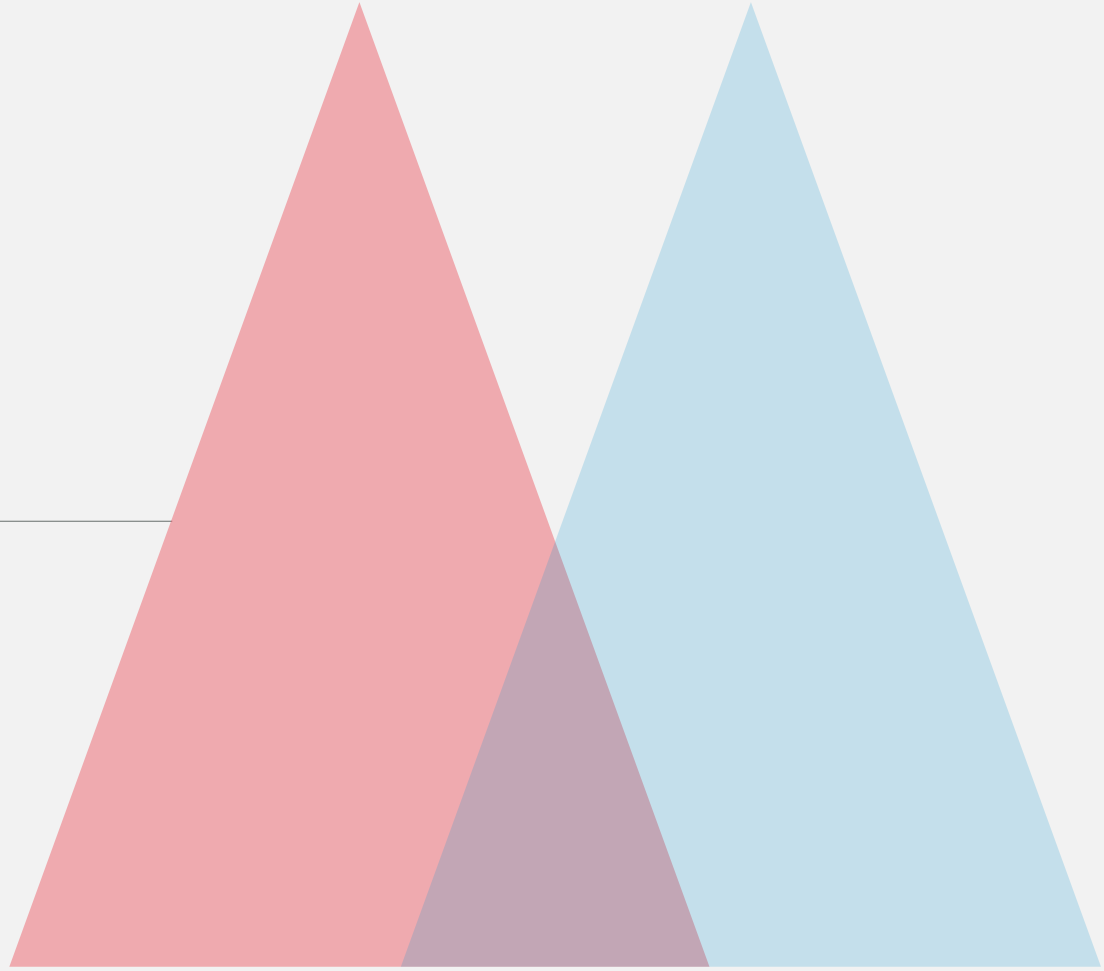
- https://www.kamis.or.kr/customer/main/main.do

IDE, 라이브러리 버전

IDE	PyCharm 2022.1
Python	3.7
Tensorflow	2.3.0
Skikit-learn	1.01
Pandas	1.3.5
Numpy	1.19.2
Keras	2.4.3
Matplotlib	3.5.1
Seaborn	0.11.2

003

Code



Code

파라미터, 파일 읽기

```
# 하이퍼 파라미터
MY_PAST = 12  # 시 계열 데이터 중 몇 개의 숫자를 입력 값으로 사용할 지 정함 / 12달치의 감자가격을 RNN의 입력 값으로 사용,
# 13번째 달의 감자 가격 예측
MY_SPLIT = 0.8  # 데이터를 얼마만큼 학습용으로 사용할 것인지 결정, 0.8 == 80%
MY_UNIT = 300  # LSTM 안의 내부의 차원 수를 결정함
MY_SHAPE = (MY_PAST, 1)  # keras LSTM은 2차원 데이터, LSTM 입력으로 들어갈 데이터의 모양

MY_EPOCH = 300  # 반복 학습 수
MY_BATCH = 64  # 병렬 계산 데이터 수
np.set_printoptions(precision=3)  # 소수점 3번째 자리까지 출력

# 데이터 파일 읽기
# 결과는 pandas의 데이터 프레임 형식
raw = pd.read_csv('price d.csv',
                  header=None,
                  usecols=[1])  # 날짜는 사용하지 않고, 감자 가격만을 사용하기 위해 1번째 cols 사용
```

3

001 Data

Code

원본 데이터 통계

```python
# 데이터 원본 출력
print('원본 데이터 샘플 13개')
print(raw.head(13))

print('\n원본 데이터 통계')
print(raw.describe())
```

```
원본 데이터 통계
                1
count     313.000000
mean     2793.821086
std      1374.172469
min       861.000000
25%      1730.000000
50%      2626.000000
75%      3510.000000
max     10740.000000
```

데이터 개수	313
평균 값(mean)	2793.821086
표준 편차(std)	1374.172469
최소값(min)	861원
25%	1730원
50%	2626원
75%	3510원
최대값(max)	10740원

3

001 Data

Code

Min-Max 정규화 통계

```python
# MinMax 데이터 정규화
scaler = MinMaxScaler()
s_data = scaler.fit_transform(raw)

print('\n정규화 데이터 통계')
print(df.describe())
```

```
정규화 데이터 통계
                0
count  313.000000
mean     0.195649
std      0.139100
min      0.000000
25%      0.087964
50%      0.178662
75%      0.268145
max      1.000000
```

데이터 개수	313
평균 값(mean)	0.195649
표준 편차(std)	0.139100
최소값(min)	0.00000
25%	0.087964
50%	0.178662
75%	0.26145
최대값(max)	1.000000

데이터 사분할

```python
# 데이터를 입력과 출력으로 분할
x_data = bundle[:, 0:MY_PAST]
y_data = bundle[:, -1]

# 데이터를 학습용과 평가용으로 분할
split = int(len(bundle) * MY_SPLIT)
x_train = x_data[: split]
x_test = x_data[split:]

y_train = y_data[: split]
y_test = y_data[split:]

# 최종 데이터 모양
print('\n학습용 입력 데이터 모양:', x_train.shape)
print('학습용 출력 데이터 모양:', y_train.shape)

print('평가용 입력 데이터 모양:', x_test.shape)
print('평가용 출력 데이터 모양:', y_test.shape)
```

```
학습용 입력 데이터 모양: (240, 12, 1)
학습용 출력 데이터 모양: (240, 1)
평가용 입력 데이터 모양: (61, 12, 1)
평가용 출력 데이터 모양: (61, 1)
```

3

001 Data

Code

```
model = Sequential()  # 순차적으로 데이터를 불러온다.
model.add(InputLayer(input_shape=MY_SHAPE))  # 입력층을 지정한다. MY_SHAPE 2차원 데이터

model.add(LSTM(MY_UNIT))  # LSTM을 모델에 추가 , MY_UNIT 하이퍼 파라미터 사용, LSTM의 차원
model.add(Dense(1, activation='sigmoid'))

print('\nRNN 요약')
model.summary()
```

```
RNN 요약
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 300)               362400
_________________________________________________________________
dense (Dense)                (None, 1)                 301
=================================================================
Total params: 362,701
Trainable params: 362,701
Non-trainable params: 0
_________________________________________________________________
```

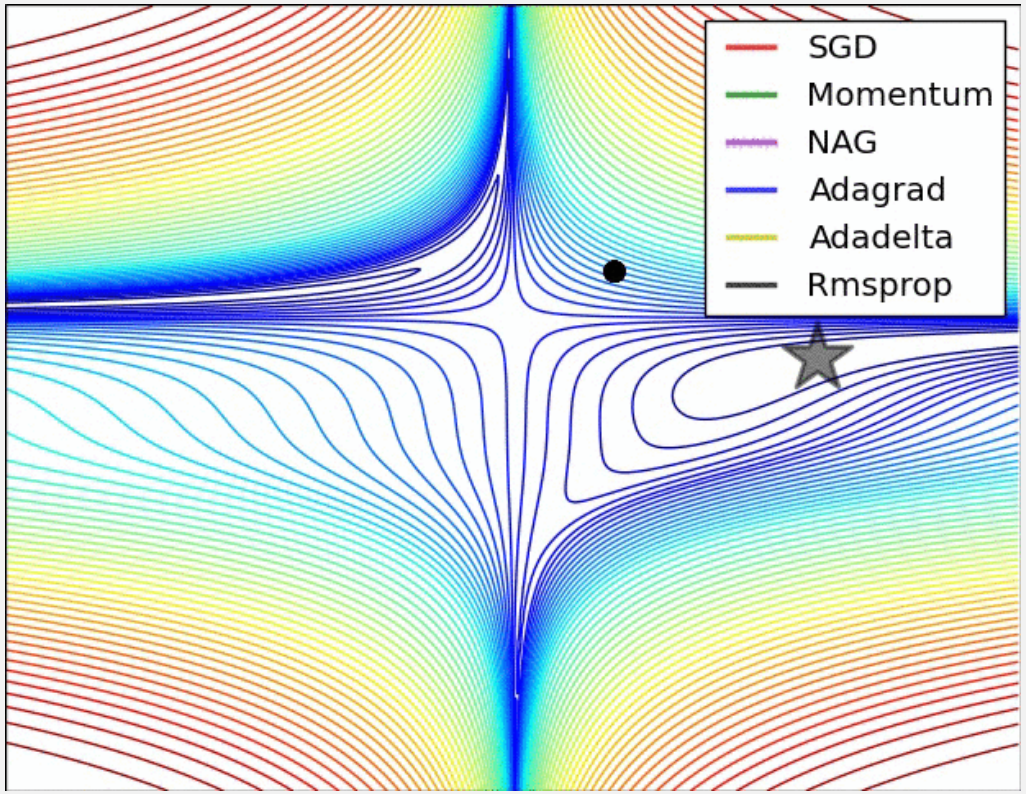
RNN 설계
- 입력 : 12달치 감자 가격
- 출력 : 13달째 감자 가격 (예측)

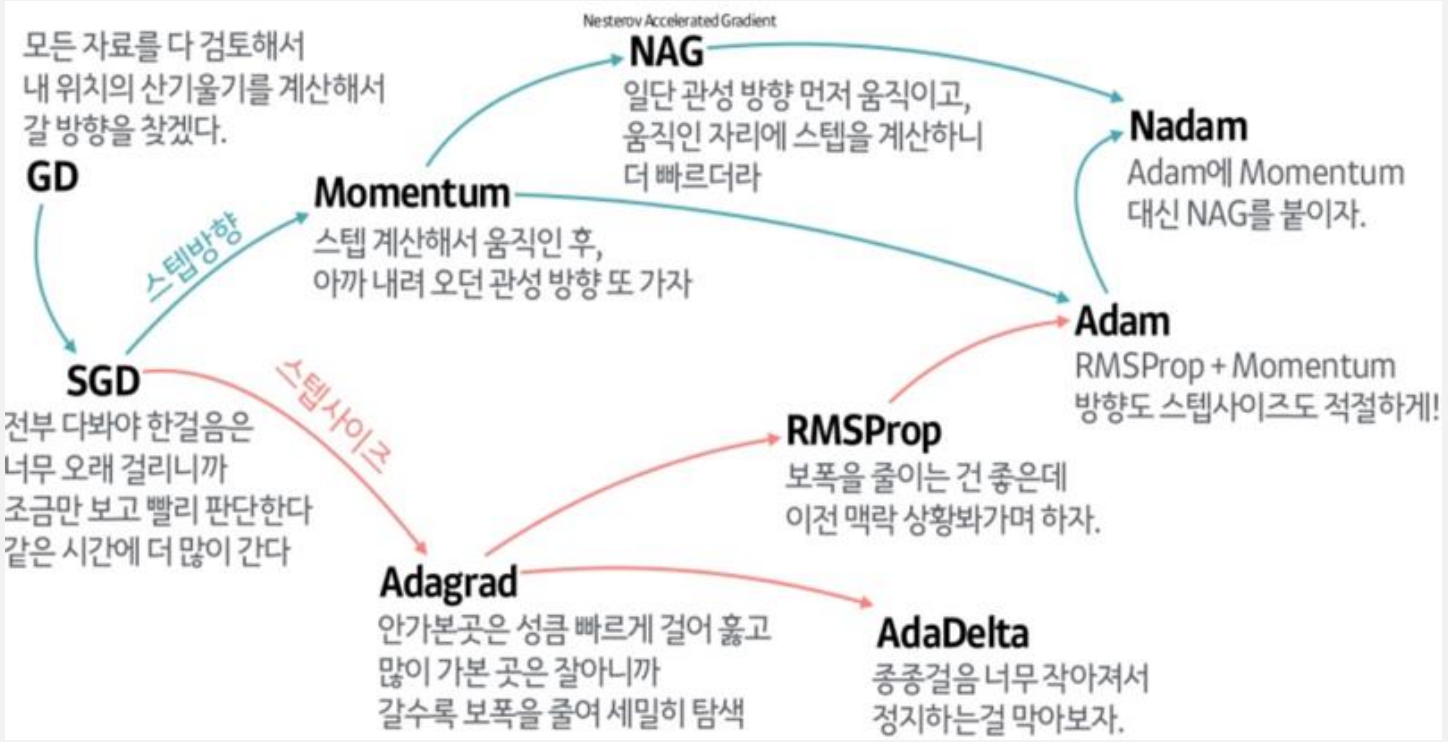
LSTM을 사용
활성화 함수로 sigmoid를 사용

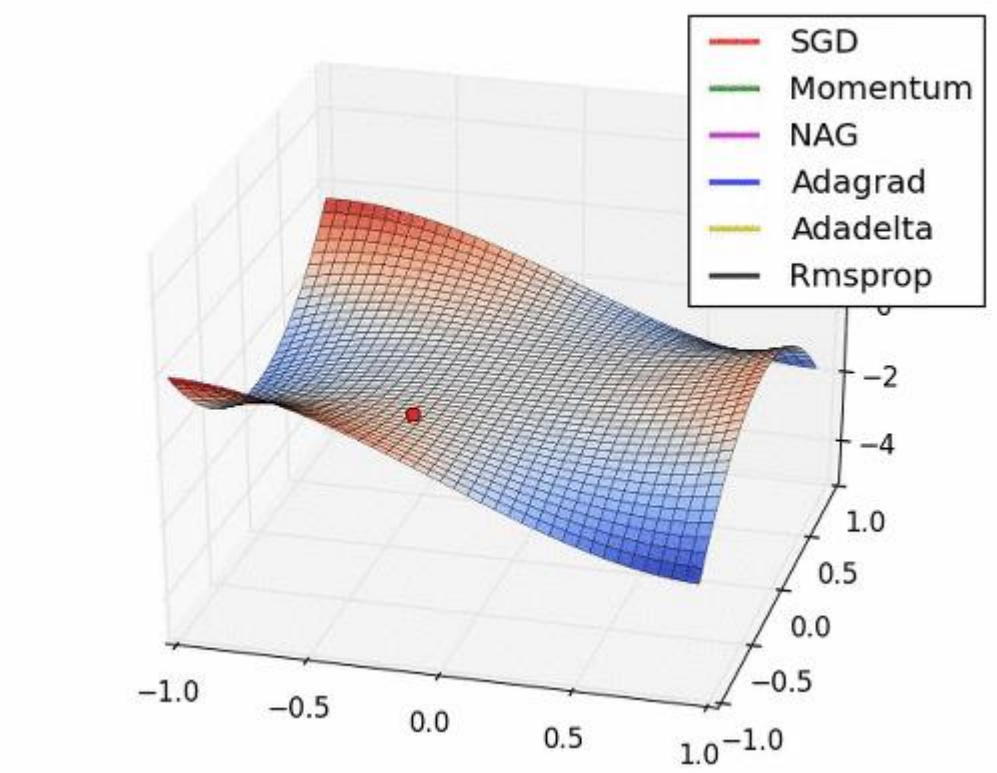
300개의 시냅스와 1개의 뉴런

LSTM의 규모는 **362,701개**

역전파 알고리즘
RMSProp(Root Mean Square Propagation)을 사용







역전파 알고리즘
RMSProp(Root Mean Square Propagation)을 사용

학습률을 상황에 맞게 조절하는 방법

학습률이 크면 가중치가 빠르게 변함
-> 학습 시간이 줄어듬, 양질의 가중치를 놓치기 쉬움

g^2_t는 기울기

For each Parameter w^j

(*j subscript dropped for clarity*)

$$\nu_t = \rho\nu_{t-1} + (1-\rho) * g_t^2$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*
ν_t : *Exponential Average of squares of gradients*
g_t : *Gradient at time t along* ω^j

역전파 알고리즘
RMSProp(Root Mean Square Propagation)

```python
# 최적화 함수와 손실 함수 지정
model.compile(optimizer='rmsprop',
              loss='mse') # 평균 제곱 오차를 사용

begin = time()
print('\nRNN 학습 시작')

model.fit(x_train,
          y_train,
          epochs=MY_EPOCH,
          batch_size=MY_BATCH,
          verbose=0)

end = time()
print('총 학습 시간: {:.1f}초'.format(end - begin))
```

```
RNN 학습 시작
총 학습 시간: 25.5초
```

Optimizer	RMSProp
Loss	MSE(평균 제곱 오차)
Time Use	25.5 Sec

3

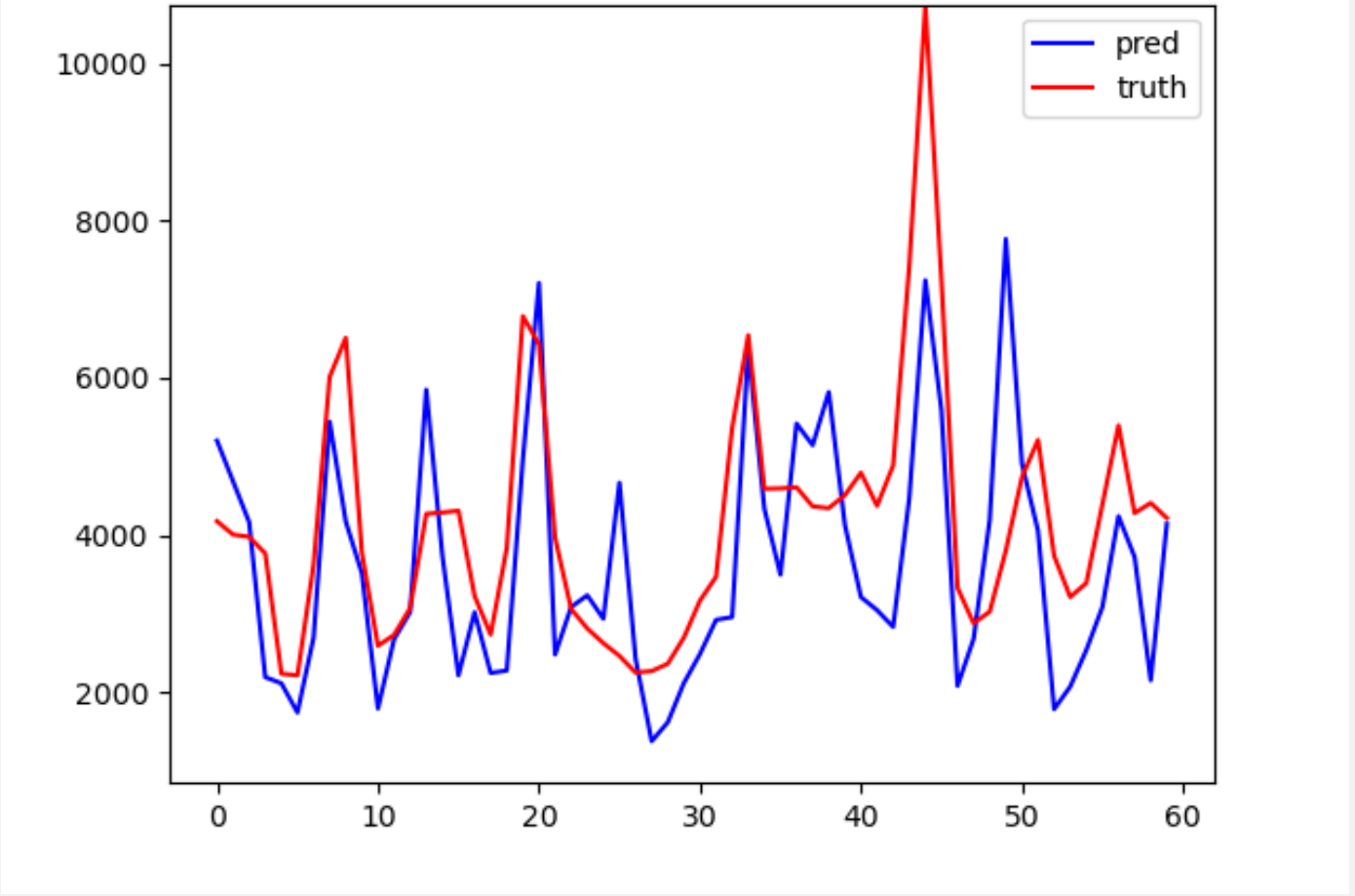
001 Data

Code

평가

```python
# RNN 평가
loss = model.evaluate(x_test,
                      y_test,
                      verbose=0)

print('최종 MSE 손실값: {:.3f}'.format(loss))

# RNN 추측
pred = model.predict(x_test)
pred = scaler.inverse_transform(pred)
pred = pred.flatten().astype(int)
print('\n추측 결과 원본:', pred)

# 정답 역전환 (정규화 값을 원래 수로 돌리는 전환)
truth = scaler.inverse_transform(y_test)
truth = truth.flatten().astype(int)
print('\n정답 원본:', truth)

# line plot 구성
axes = plt.gca()
axes.set_ylim([850, 10750])

sns.lineplot(data=pred, label='pred', color='blue')
sns.lineplot(data=truth, label='truth', color='red')

plt.show()
```

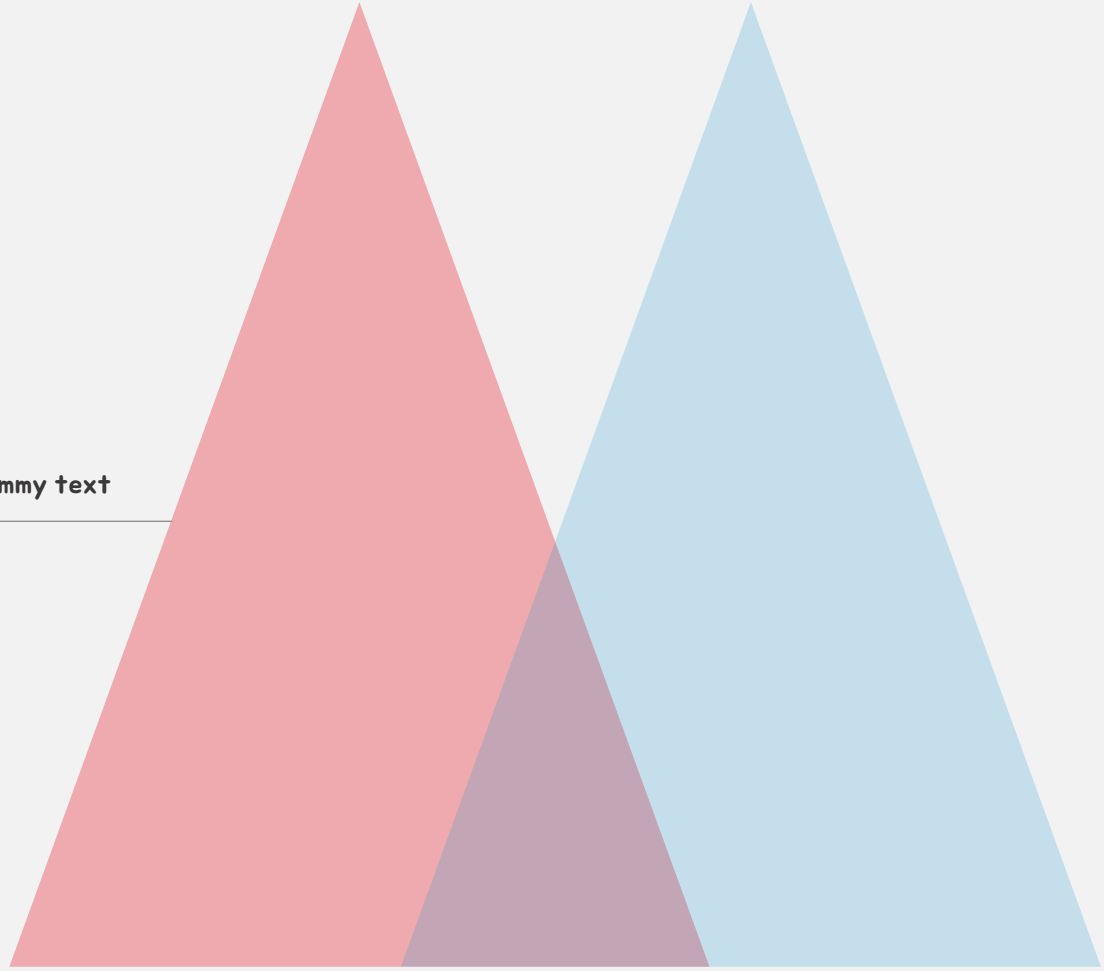
```
최종 MSE 손실값: 0.018
```



004

Lorem Ipsum is simply dummy text

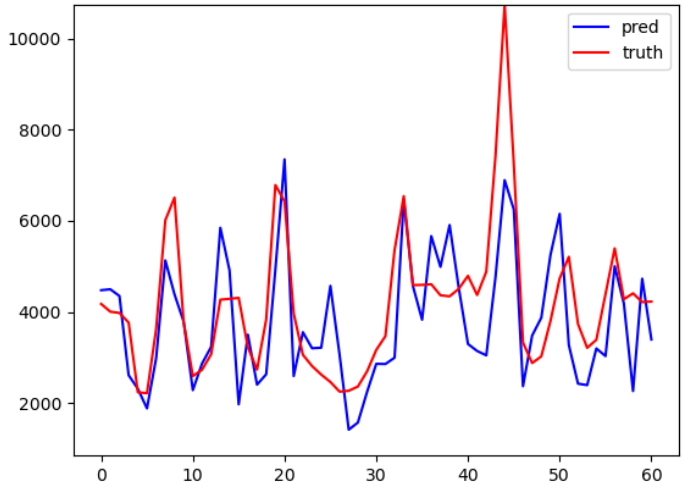
다양한 실험 결과



다양한 실험 결과

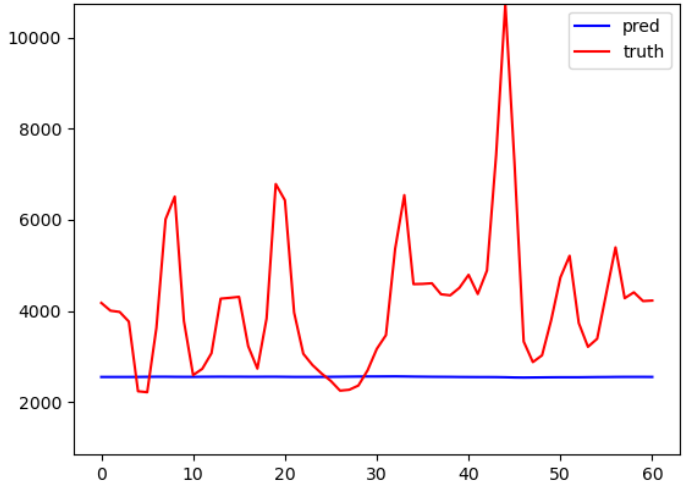
Optimizer 변경

RMSProp



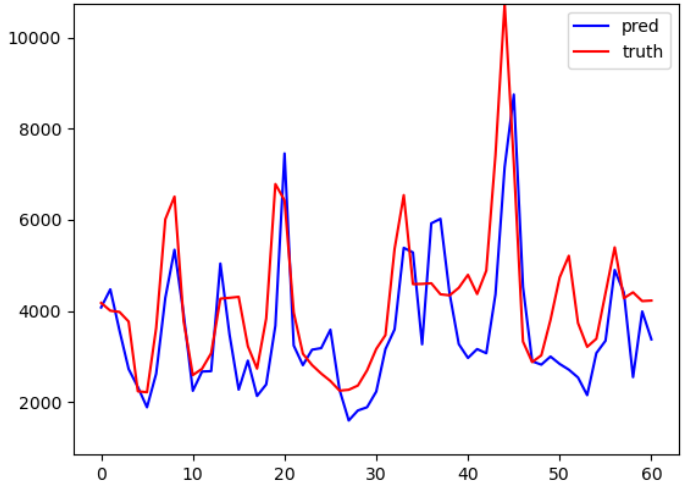
Optimizer	RMSProp
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

SGD(확률적 경사하강법)



Optimizer	SGD
Total Param	362,701
Loss(MSE)	0.052
Time Use	22.3초

Adam(가장 대중적으로 쓰이는 알고리즘)

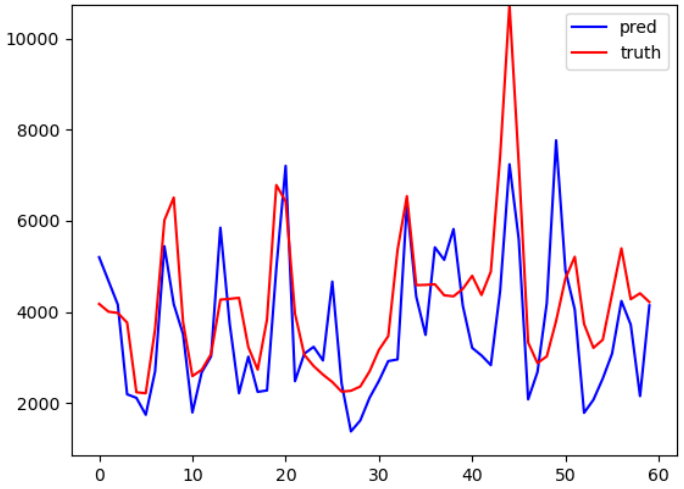


Optimizer	Adam
Total Param	362,701
Loss(MSE)	0.016
Time Use	23.0초

다양한 실험 결과

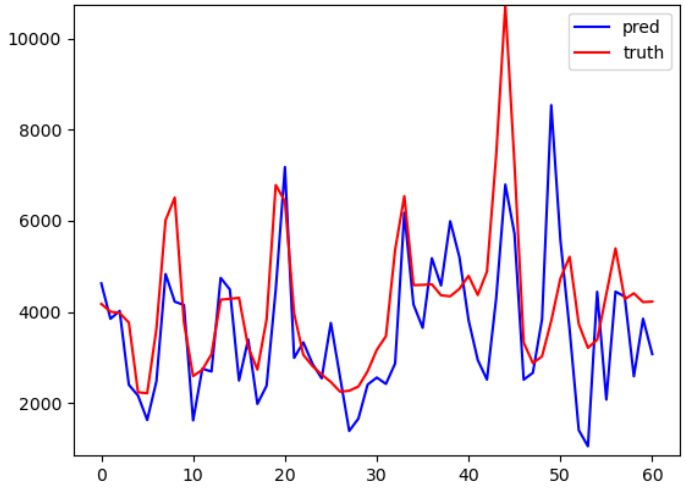
Activation(활성화 함수) 변경

Tanh



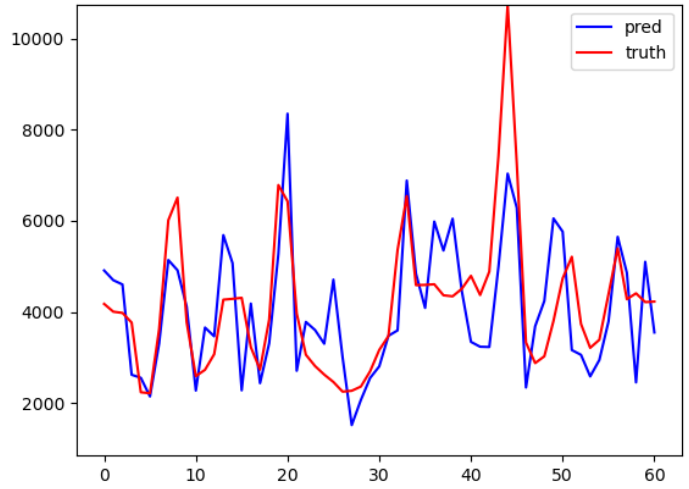
Activation	Tanh
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

None(삭제)



Activation	None
Total Param	362,701
Loss(MSE)	0.021
Time Use	23.7초

Linear(선형)

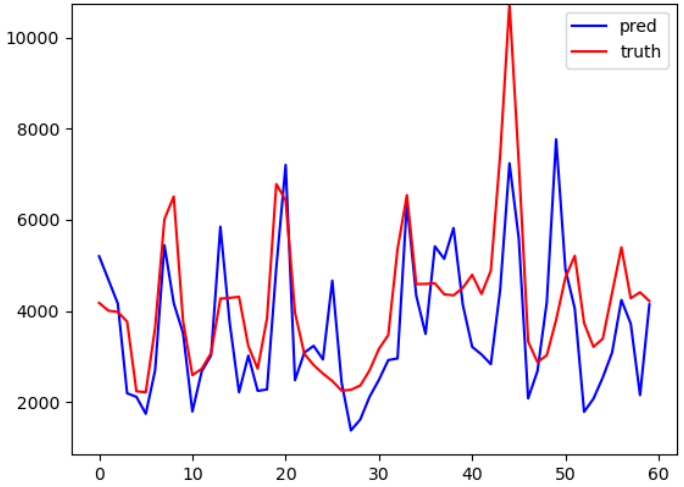


Activation	Linear
Total Param	362,701
Loss(MSE)	0.019
Time Use	24.0초

다양한 실험 결과

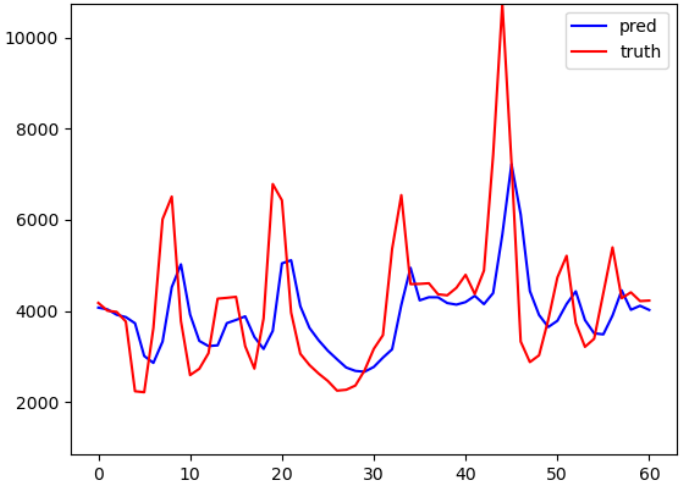
LSTM 규모 변경 (LSTM 셀의 차원 수 변경)

MY_UNIT = 300



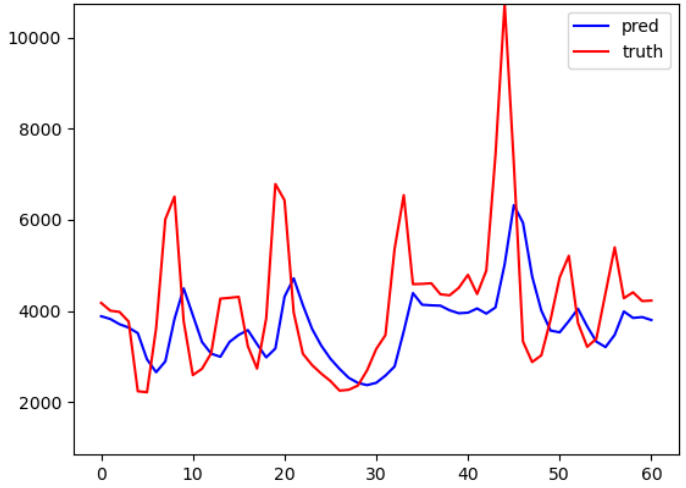
MY_UNIT	300
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

MY_UNIT = 50



MY_UNIT	50
Total Param	10,451
Loss(MSE)	0.017
Time Use	3.8초

MY_UNIT = 10

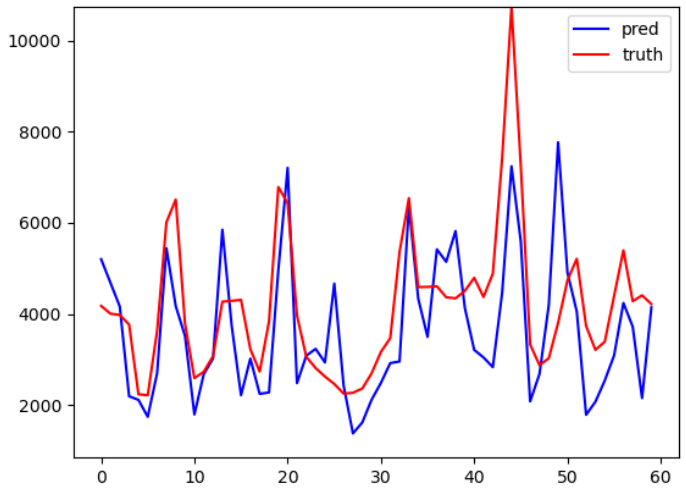


MY_UNIT	10
Total Param	491
Loss(MSE)	0.022
Time Use	3.2초

다양한 실험 결과

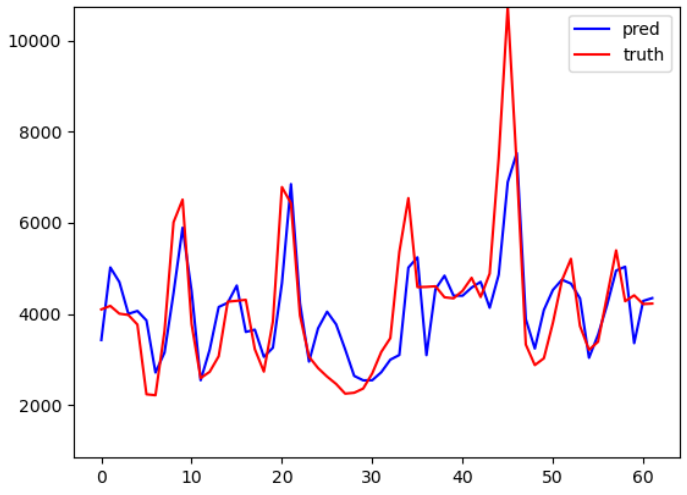
과거 데이터 규모 변경 (현재 12개월 단위)

MY_PAST = 12



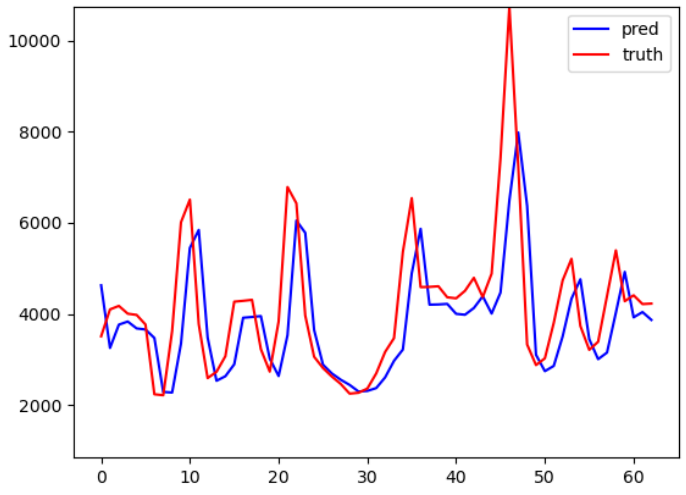
MY_PAST	12
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

MY_PAST = 24



MY_PAST	6
Total Param	362,701
Loss(MSE)	0.010
Time Use	13.7초

MY_PAST = 6

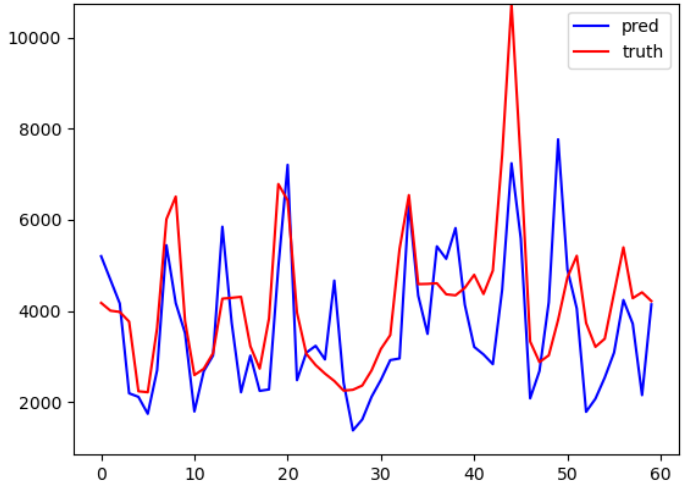


MY_PAST	3
Total Param	362,701
Loss(MSE)	0.008
Time Use	8.5초

다양한 실험 결과

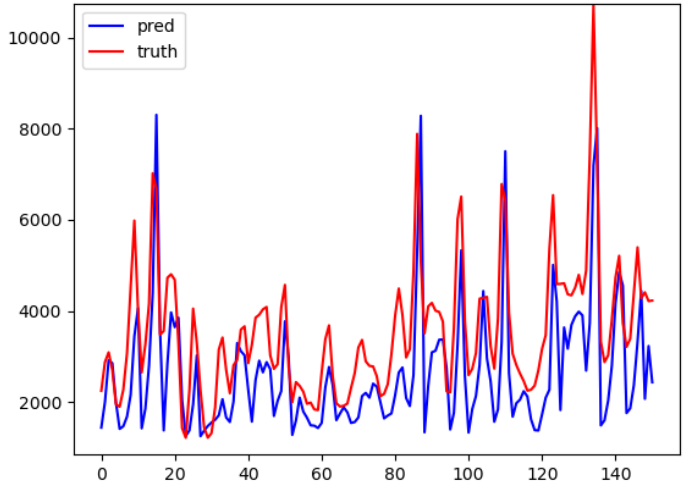
학습용 / 평가용 데이터 비율 조정

MY_SPLIT = 0.8 (80%)



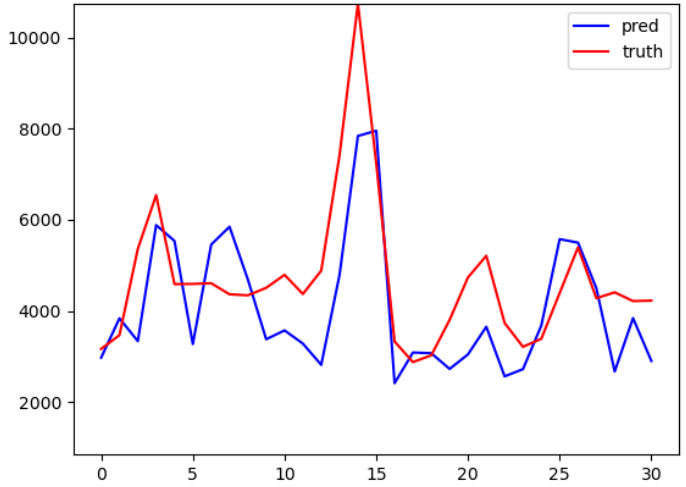
MY_SPLIT	0.8
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

MY_SPLIT = 0.5 (50%)



MY_SPLIT	0.5
Total Param	362,701
Loss(MSE)	0.018
Time Use	16.9초

MY_SPLIT = 0.9 (90%)

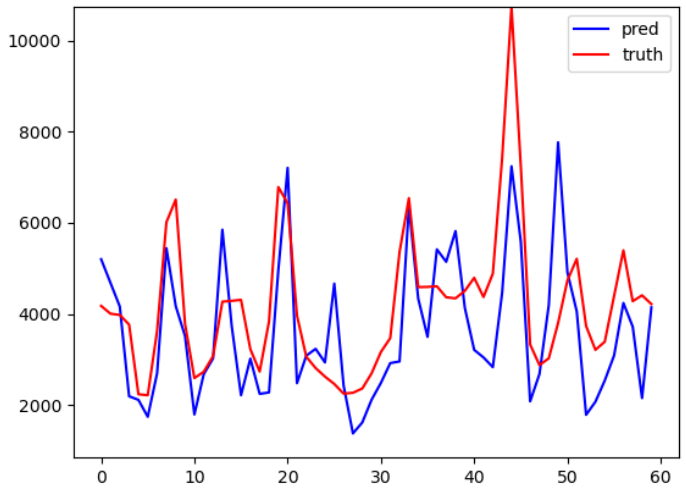


MY_SPLIT	0.9
Total Param	362,701
Loss(MSE)	0.016
Time Use	28.3초

다양한 실험 결과

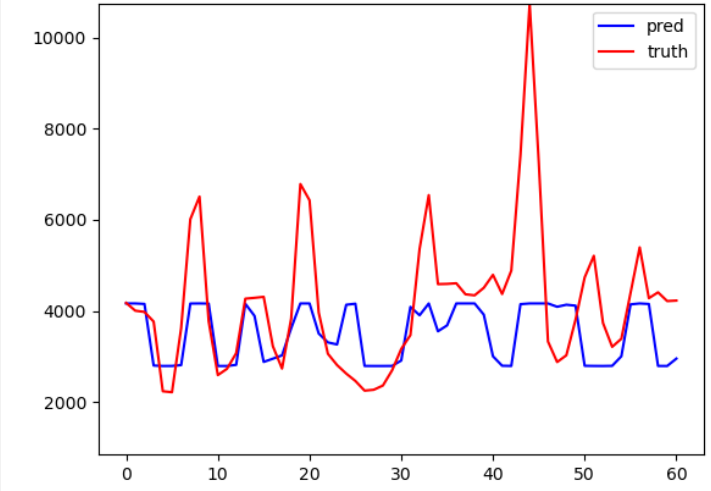
정규화 종류 변경

MinMaxScaler([0, 1] 범위)



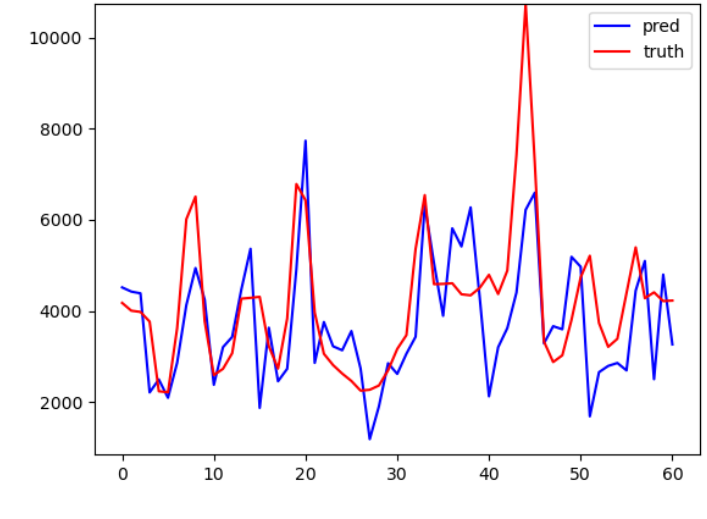
Scaler	MinMaxScaler
Total Param	362,701
Loss(MSE)	0.015
Time Use	24.2초

StandardScaler(평균 0, 표준편차 1)



Scaler	StandardScaler
Total Param	362,701
Loss(MSE)	1.207
Time Use	25.2초

MaxAbsScaler(최대 절대값 = 1, 0 = 0)



Scaler	MaxAbsScaler
Total Param	362,701
Loss(MSE)	0.015
Time Use	25.5초

Q&A

Thank You