캡스톤 디자인 - 최종 발표

제목 : 시계열 분석을 통한 MSRC Trace 예측

16101666 - 안재현

목차

주제 선정 및 이점

주제 선정 및 이점

주제 : 시계열 분석을 통한 MSRC Trace 예측

다양한 SSD 시뮬레이터는 실험을 위해 Trace를 임의로 늘려 성능을 테스트하는 경우가 많이 존재함
이를 시계열 분석을 통해, 딥러닝으로 예측하는 결과를 보냄

1. 실험 중 Trace를 임의로 늘리는 경우는 실험환경 설정에 오류가 날 수 있으므로,
제대로 된 성능 측정이 불가함 -> 해결할 수 있음

2. MSRC 트레이스 파일의 데이터 분석 자료 활용

주제 선정 및 이점

주제 : 시계열 분석을 통한 MSRC Trace 예측

논문명 : Optimizing NAND Flash-Based SSDs via Retention Relaxation

requirements less than X within a time period of Y as $S_{X,Y}\%$. We first formulate $S_{T_1,T_1}\%$ in the write amount and the write working set size, where T_1 is the time span of the trace. Let N be the amount of data sectors written into the disk during T_1 and W be the write working set size (i.e., the number of distinct sector addresses being written) during T_1. We have the following formula (the proof is similar to the pigeonhole principle):

$$S_{T_1,T_1}\% = \frac{N-W}{N} = 1 - \frac{W}{N} \tag{13}$$

With this formula, the first projection we make is the percentage of writes that have retention time requirements less than T_1 in an observation period of T_2, where $T_2 = k \times T_1$, $k \in \mathbb{N}$. The projection is based on the assumption that for each T_1 period, the write amount and the write working set size remain N and W, respectively. We derive the lower bound on $S_{T_1,T_2}\%$ as follows:

$$S_{T_1,T_2}\% = \frac{k(N-W) + \sum_{i=1}^{k-1} u_i}{kN} \geq S_{T_1,T_1}\% \tag{14}$$

6 System Evaluation

We conduct simulation-based experiments using SSDsim [5] and Disksim-4.0 [10] to evaluate the RR-10week and RR-2week designs. SSDs are configured to

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

MSRC Trace 파일 분석

MSRC Trace 파일 분석

MSRC의 전체 재기록 기간 : **7**일

T1 : 지정된 시간 **T2** : 재기록 시간

총 5가지의 변수로 Trace를 분석

N : T1 동안의 **All write sector** 개수

W : T1 동안의 **All write sector**의 **unique sector**의 개수

ui : 2 x T1 걸치면서 **T2** 이하의 **write sector** 개수

T2_N : N 중 **T2** 이하의 재기록된 **write sector** 횟수

T2_W : T2_N에서 사용된 **unique sector**의 개수

MSRC Trace 파일 분석

```
/* T1 시간 내에 재기록되는 섹터의 수 및 시간 입력 방법 */

while( End of trace )
{
   for(int i = 1; i <= LENGTH_NUM; i++)
   {
        If( T1 * ( i - 1 )  <  Time  <   T1 *  i )
        {
            Sector[num]'s count ++;
            if ( Sector[num]'s count  == 1 )
            {
                Sector[num]'s First and Last Time = Time
            }

            else if ( sector[num]'s count > 1)
            {
                Sector[num]'s Last Time = Time
            }

            if( Sector[num]'s count >= 2  and  ( Time - Sector[num]'s Last Time ) < T2 )
            {
                T2_in_T1_count[ i-1 ] ++ ;
            }
        }
    }
}
```

```
/*   N, W 구하기     */

for(num = 0; num <= MAXSECTOR ; num++)
{
     for( i = 0; i < LENGTH_NUM; i++)
     {

      if( Sector[num]'s count  != 0 )
         {
         Unique Sector[num] Count ++ ;
         }

      All sector count += Sector[num]'s count ;
     }
}
```

```
/*   ui 구하기      */

for(int i = 1; i <= LENGTH_NUM; i++)
{
     if( Sector[num]'s First Time - Sector[num-1]'s Last Time <= T2 )
     {
         Sector[num]'s ui ++;
     }
}
```

Trace 파일

24가지의 Trace 파일 분석

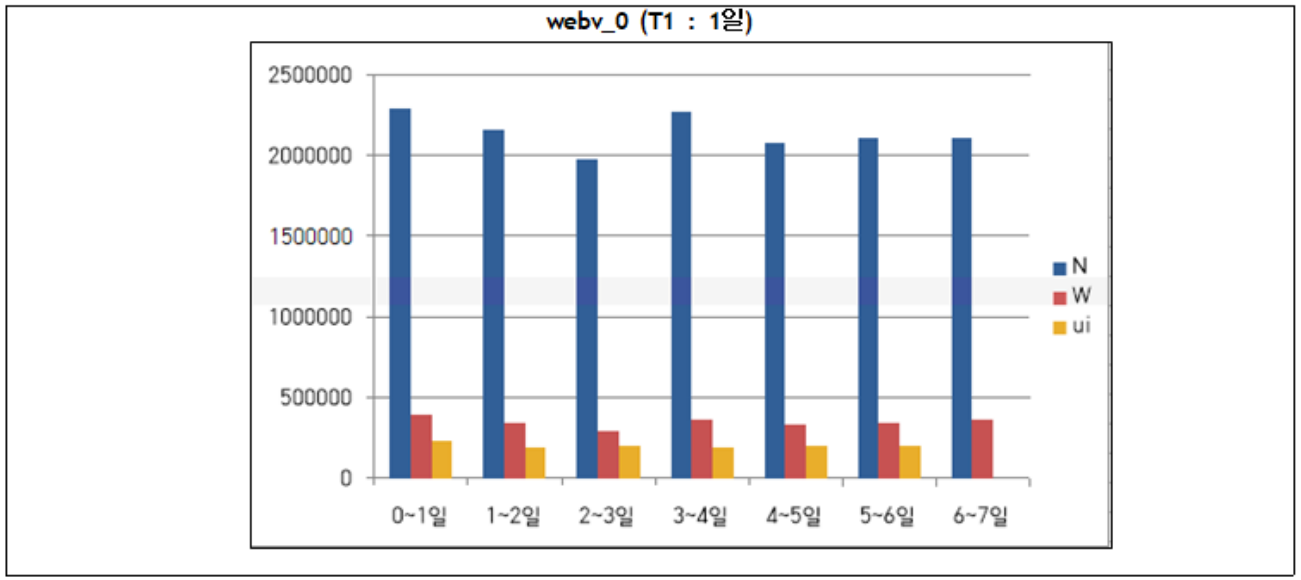
N1	W1	T2_N	T2_W	ui	N1(24)	W1(24)	T2_N(24)	T2_W(24)	ui(24)
210200	97584	112616	33456	33456	419544	161080	258464	33456	33456
209344	96952	112392	33456	33456	418792	160168	258624	33456	33456
208488	95992	112496	33456	33456	419920	161296	258624	33456	33456
210304	97632	112672	33456	33456	421784	163264	258520	33456	53096
209152	96608	112544	33456	33456	420728	162000	258728	33488	33464
210768	98144	112624	33456	33456	418232	159744	258488	33456	33560
210024	97608	112416	33456	33456	430951	165894	265057	35153	
211760	99112	112648	33456	33456					
209872	97408	112464	33456	33456					
210856	98048	112808	33488	33456					
209296	96688	112608	33456	33456					
208936	96512	112424	33456	33456					
208704	96360	112344	33456	33456					
222247	103454	118793	34689						

	N1	W1	T2_N	T2_W	ui
wdev_2	0.12	0.14	0.11	0.07	0.00
wdev_1	0.48	1.01	0.37	0.00	0.00
wdev_3	0.76	1.79	0.64	0.00	0.00
wdev_0	0.66	0.80	0.71	0.95	0.18
					0.75
					0.66
					1.43
					1.08
					1.16
					0.12
					0.32
					1.64
					5.39
					7.84
					1.36
					7.74
					1.39
					5.87
					6.35
					7.57
					14.36
					18.17
					23.80
proj_3	13.21	14.18	10.31	12.35	20.51

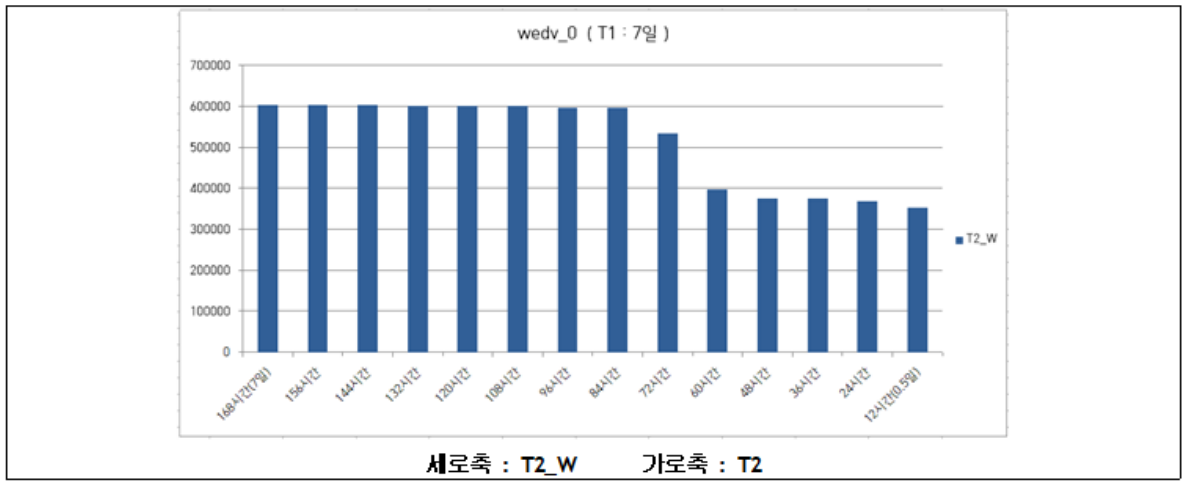
	mean(sd)	All trace	Write Only
wdev_2	0.09	9.14MB	6.37MB
wdev_1	0.37	55KB	38KB
wdev_3	0.64	34.8KB	23.6KB
wdev_0	0.66	57.4MB	32.2MB
mds_0	0.87	60.9MB	38.7MB
stg_0	0.87	100MB	61.1MB
usr_0	1.49	112MB	145MB
web_0	1.61	57.4MB	51.9MB
ts_0	1.80	88.9MB	54.2MB
src1_1	2.20	2.36GB	79.0MB
stg_1	2.35	110MB	28.4MB
src2_0	3.26	78.6MB	48.9MB
hm_1	4.51	28.3MB	0.99MB
hm_0	4.57	192MB	91.1MB
prxy_0	4.57	628MB	428MB
web_3	5.28	1.57MB	787KB
web_2	5.30	263MB	1.4MB
web_1	6.53	8.1MB	2.71MB
src1_2	8.01	96.5MB	51.3MB
proj_0	9.87	222MB	141MB
mds_1	12.07	84.2MB	4.19MB
proj_4	12.63	333MB	3.4MB
src2_1	13.52	33.9MB	523KB
proj_3	14.11	114MB	4.25MB

Trace 파일 시각화

Server	Function	#volumes
usr	User home directories	3
proj	Project directories	5
prn	Print server	2
hm	Hardware monitoring	2
rsrch	Research projects	3
prxy	Firewall/web proxy	2
src1	Source control	3
src2	Source control	3
stg	Web staging	2
ts	Terminal server	1
web	Web/SQL server	4
mds	Media server	2
wdev	Test web server	4



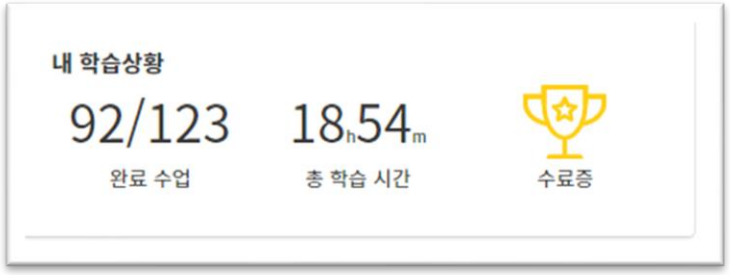
(T1 : 1일) N과 W의 양을 나타낸 그래프



T2를 12시간씩 줄여가면서 T2_W의 변화량을 관찰한 그래프

LSTM 시계열 예측모델

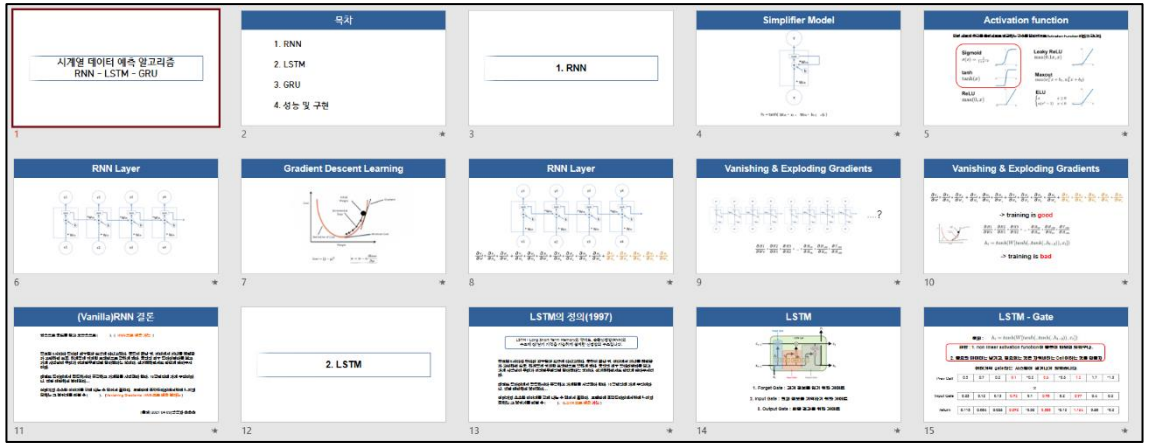
머신러닝 스터디

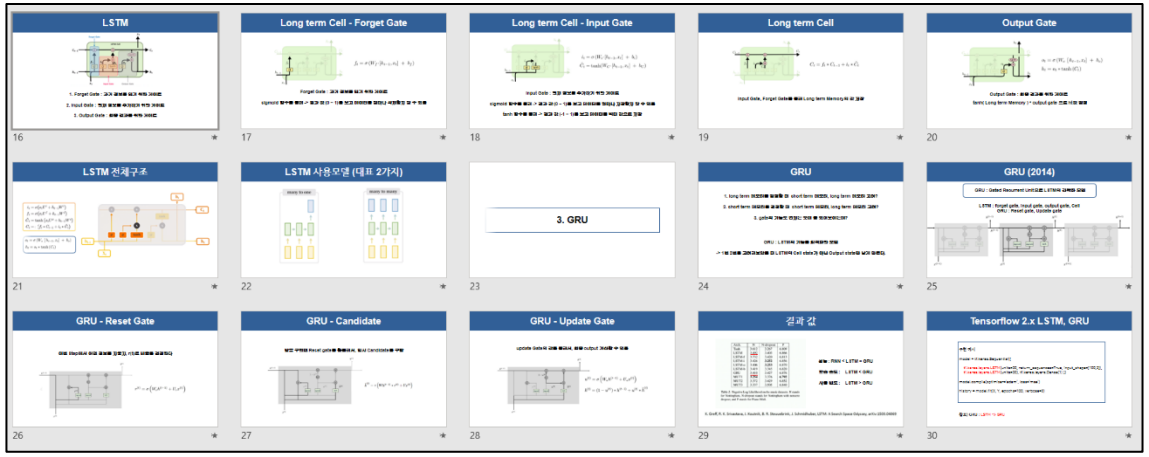


머신러닝의 개념	14 : 44
파이썬 기반 머신러닝의 특징및 장점과 구성요소	10 : 15
파이썬기반 머신러닝을 위한 SW의 설치	14 : 34
주피터 노트북 사용법과 넘파이/판다스의 필요성	19 : 45
강의에 사용될 예제 소스 코드 다운로드 받기	03 : 11
넘파이 배열 ndarray 소개	10 : 40
넘파이 배열 ndarray 초기화 방법과 ndarray차원과 크기를 변경하는 reshape()의 이해	08 : 14
넘파이 ndarray의 인덱싱(Indexing)을 통한 데이터 세트 선택하기 - 01	07 : 19
넘파이 ndarray의 인덱싱(Indexing)을 통한 데이터 세트 선택하기 - 02	09 : 29
넘파이 ndarray의 정렬과 선형대수 연산	13 : 51
판다스(Pandas) 개요와 기본 API - 01	10 : 38
판다스(Pandas) 개요와 기본 API - 02	16 : 07
판다스 DataFrame의 변환, 컬럼 세트 생성/수정, 삭제 및 Index 객체 소개	20 : 39
판다스 데이터 셀렉션과 필터링 - 01	07 : 59
판다스 데이터 셀렉션과 필터링 - 02	19 : 37
판다스 Aggregation함수와 Group by 수행	15 : 22
판다스 결손 데이터 처리하기	05 : 17
판다스 람다식 적용하여 데이터 가공하기	09 : 27
파이썬 기반의 머신러닝과 생태계 이해 Summary	04 : 04
섹션 2. 사이킷런으로 시작하는 머신러닝	8 강의 102 : 03
사이킷런 소개와 첫번째 머신러닝 애플리케이션 만들어 보기 - 붓꽃(Iris) 품종 예측	16 : 15
사이킷런의 기반 프레임 워크 익히기 - 주요 API/모듈 및 내장 예제 데이터 세트 소개	08 : 54
학습과 테스트 데이터 세트의 분리	09 : 55
교차검증 - K-Fold와 Stratified K-Fold	15 : 36
교차검증 성능평가 cross_val_score()와 하이퍼 파라미터 튜닝을 위한 GridSearchCV	13 : 28
데이터 전처리 - 인코딩과 스케일링 - 01	07 : 37

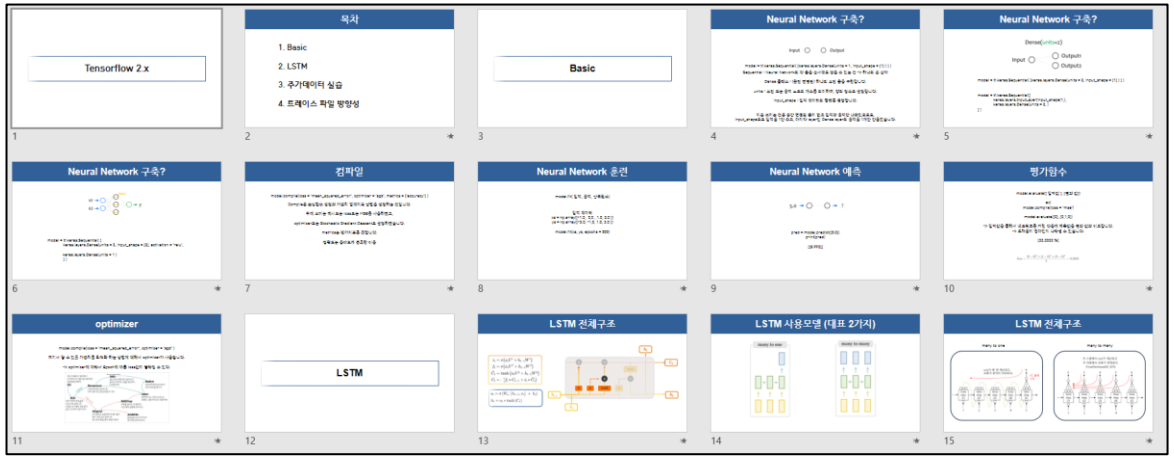
딥러닝 스터디

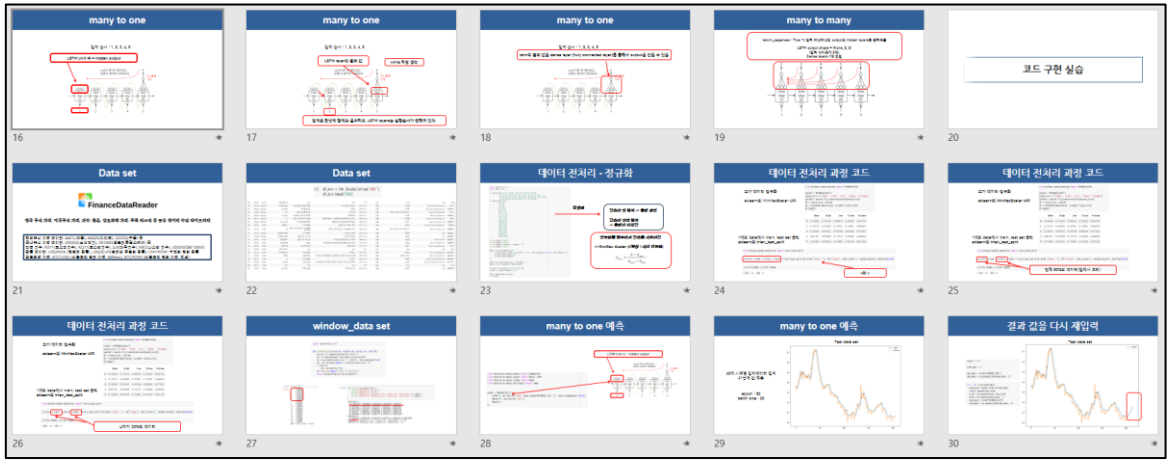
딥러닝 개념 이해





Tensorflow 구현 이해





시계열 데이터 실습

1.(주가데이터 셋) FinanceDataReader

```
df_krx = fdr.StockListing('KRX')
df_krx.head(7000)
```

6979	227840	KOSPI	현대코퍼레이션홀딩스	음식료품 및 담배 도매업	브랜드사업, 목류유통	2015-10-23	12월	정몽혁, 김원갑(각자대표이사)	http://www.hyundaicorpholdings.com
6980	126560	KOSPI	현대퓨처넷	텔레비전 방송업	광고업	2010-12-23	12월	류성택	http://www.hyundaifuturenet.co.kr
6981	001450	KOSPI	현대해상	보험업	손해보험(자동차보험)	1989-08-25	12월	조용일, 이성재	http://www.hi.co.kr
6982	057050	KOSPI	현대홈쇼핑	무점포 소매업	TV홈쇼핑 도소매/홈쇼핑프로그램 제작	2010-09-13	12월	정교선, 임대규(각자 대표이사)	http://www.hmall.com
6983	092300	KOSDAQ	현우산업	전자부품 제조업	인쇄회로판(PCB)	2007-10-24	12월	문병선, 문인식 (각자대표)	NaN
6984	053660	KOSDAQ	현진소재	기타 금속 가공제품 제조업	선박용 엔진부품	2002-02-15	12월	이도현	http://www.hjmco.co.kr
6985	011080	KOSDAQ	형지I&C	봉제의복 제조업	남성셔츠	2004-07-02	12월	최혁원	http://www.woosunginc.com
6986	093240	KOSPI	형지엘리트	봉제의복 제조업	학생복,스포츠의류,작업복 제조,도매	2009-09-28	06월	최병오	http://www.hyungji-elite.com
6987	003010	KOSPI	혜인	기계장비 및 관련 물품 도매업	건설기계,산업용 기계장비(진공흡인차,세정차),스태커크레인,비상용발전기,해상풍연진 도...	1988-09-20	12월	원경희	http://www.haein.com
6988	111110	KOSPI	호전실업	봉제의복 제조업	스포츠웨어 및 아웃도어 웨어 OEM	2017-02-02	12월	박종철 박진호	http://www.hojeon.com
6989	008770	KOSPI	호텔신라	기타 상품 전문 소매업	면세판매,관광숙박,외식사업,예식업	1991-03-12	12월	이부진	http://www.hotelshilla.net
6990	008775	KOSPI	호텔신라우	NaN	NaN	NaT	NaN	NaN	Na
6991	060560	KOSDAQ	홈센타홀딩스	기타 전문 도매업	건축자재,레미콘	2002-07-11	12월	박 병 윤	http://www.home-center.co.kr

```
STOCK_CODE = '035810'
```

```
stock = fdr.DataReader(STOCK_CODE, '2015', '2020')
```

```
stock.tail()
```

Date	Open	High	Low	Close	Volume	Change
2019-12-23	4768	4863	4768	4834	374778	0.014694
2019-12-24	4813	4838	4788	4814	205932	-0.004137
2019-12-26	4813	4903	4798	4903	305143	0.018488
2019-12-27	4863	4918	4848	4918	248591	0.003059
2019-12-30	4918	4978	4893	4973	332172	0.011183

데이터는 **2015**년 ~ **2020**년까지의 데이터를 받아옴

2. 데이터 정규화

```
[208] from sklearn.preprocessing import MinMaxScaler

      scaler = MinMaxScaler()
      scale_cols = ['Open', 'High', 'Low', 'Close', 'Volume']
      scaled = scaler.fit_transform(stock[scale_cols])
      df = scale_cols, scaled
      df = pd.DataFrame(scaled, columns= scale_cols)
      df.head()
```

	Open	High	Low	Close	Volume
0	0.146393	0.112676	0.149615	0.130255	0.055703
1	0.129125	0.106888	0.130861	0.119901	0.084870
2	0.100345	0.091453	0.108148	0.082626	0.077422
3	0.081351	0.074088	0.112315	0.105405	0.048903
4	0.102264	0.070230	0.116483	0.103334	0.031352

시계열 데이터 실습

3. train, test data set 분리(종가 데이터만 사용) 20퍼센트의 비율로 분리

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(df.drop('Close', 1), df['Close'], test_size=0.2, random_state=0, shuffle=False)
```

```
x_train.shape, y_train.shape
```

```
((981, 4), (981,))
```

```
y_test.shape
```

```
(246,)
```

```
import tensorflow as tf
```

```
def windowed_dataset(series, window_size, batch_size, shuffle):
    series = tf.expand_dims(series, axis=-1)
    ds = tf.data.Dataset.from_tensor_slices(series)
    ds = ds.window(window_size + 1, shift=1, drop_remainder=True)
    ds = ds.flat_map(lambda w: w.batch(window_size + 1))
    if shuffle:
        ds = ds.shuffle(1000)
    ds = ds.map(lambda w: (w[:-1], w[-1])) #
    return ds.batch(batch_size).prefetch(1)
```

```
WINDOW_SIZE=40
BATCH_SIZE=30
```

```
train_data = windowed_dataset(y_train, WINDOW_SIZE, BATCH_SIZE, True)
test_data = windowed_dataset(y_test, WINDOW_SIZE, BATCH_SIZE, False)
```

```
for data in train_data.take(1):
    print(f'데이터셋(X) 구성(batch_size, window_size, feature갯수): {data[0].shape}')
    print(f'데이터셋(Y) 구성(batch_size, window_size, feature갯수): {data[1].shape}')
```

```
데이터셋(X) 구성(batch_size, window_size, feature갯수): (30, 40, 1)
데이터셋(Y) 구성(batch_size, window_size, feature갯수): (30, 1)
```

4. 모델 구축 - LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.losses import Huber
from tensorflow.keras.optimizers import Adam


model = Sequential([
    LSTM(30, activation='tanh',input_shape=[WINDOW_SIZE, 1], return_sequences= False),
    Dense(16, activation="relu"),
    Dense(1),
])
```

5. compile 및 fit

```
optimizer = Adam(0.0005)
model.compile(loss=Huber(), optimizer=optimizer, metrics=['mse'])
```

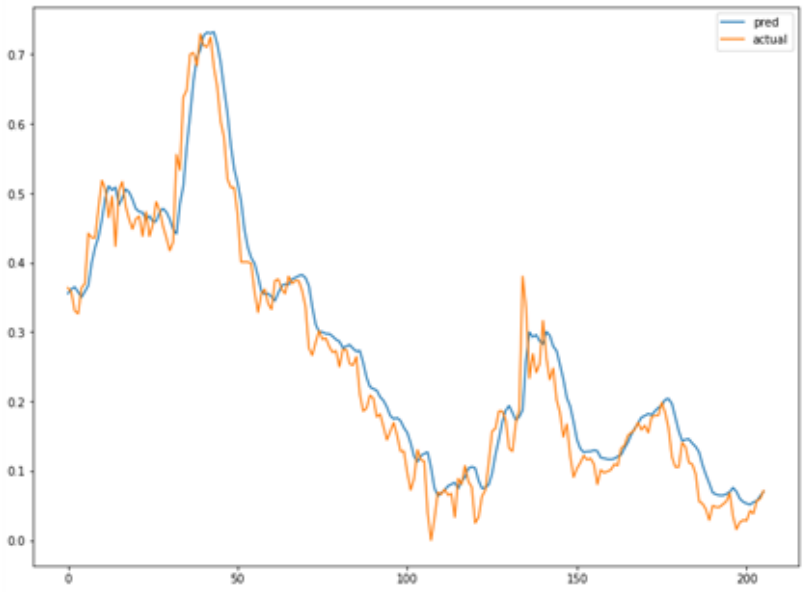
```
model.fit(train_data, validation_data=(test_data), epochs=50)
```

시계열 데이터 실습

6. 모델 학습 이후 **test_data set** 으로 확인

```
pred = model.predict(test_data)
```

7. 시각화



8. 예측한 데이터를 입력값으로 넣어서 다시 예측 **10**일 간 예측

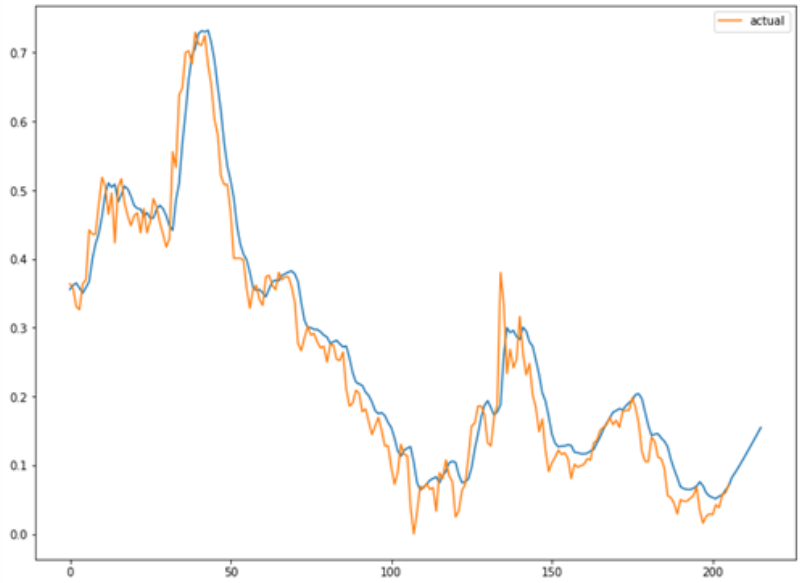
```
result = []

pred_day = 10

new_data = pred[-WINDOW_SIZE:]
new_data = np.expand_dims(new_data, 0)
```

```
for i in range(pred_day):
  new_pred = model.predict(new_data)
  result.append(new_pred)
  pred = np.append(pred, new_pred)
  pred = np.expand_dims(pred, 1)
  new_data = pred[-WINDOW_SIZE:]
  new_data = np.expand_dims(new_data, 0)
```

9. 시각화



결론 및 방향성

결론 및 방향성

발전성은 총 두가지가 보이게 되었습니다.

실제로 many to one에서 나오는 것은 상한 하한이 존재하기 때문에
test 데이터에서 비슷한 경향이 나오는 것은 당연하다.

예측 방법 발전 : 예측할 경우 실제 데이터를 포함하여 예측해야 한다.

1. 하루 뒤 예측 그래프의 상승 및 하강 추세가 맞는지 아닌지를 체크
2. 상승 및 하강의 양을 테스트 하며 실제와 비슷한지 아닌지를 체크

trace 데이터의 경우 각각의 데이터가 방대하기 때문에,
이를 머신이 받아드릴 수 있을까라는 의문
-> Trace 파일의 데이터 분석 측면에서 조금 더 자세하게 들어가 보는건?