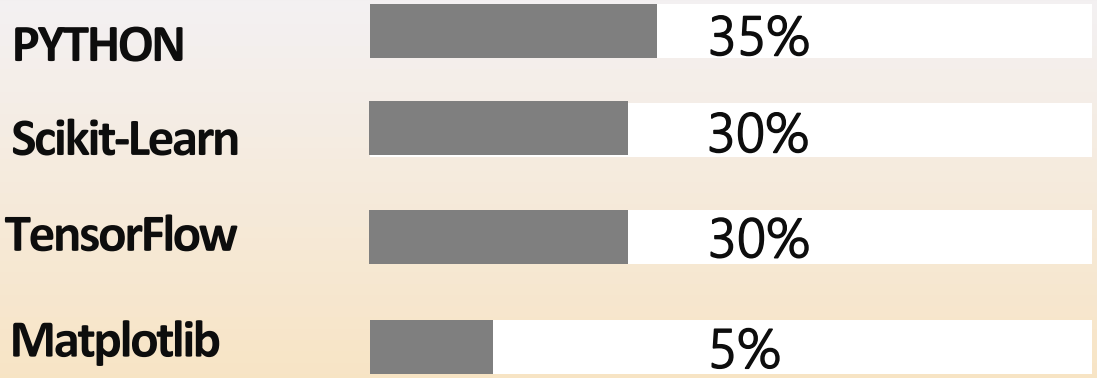
2022.01.07

LSTM을 활용한 비트코인 실시간 시세 예측



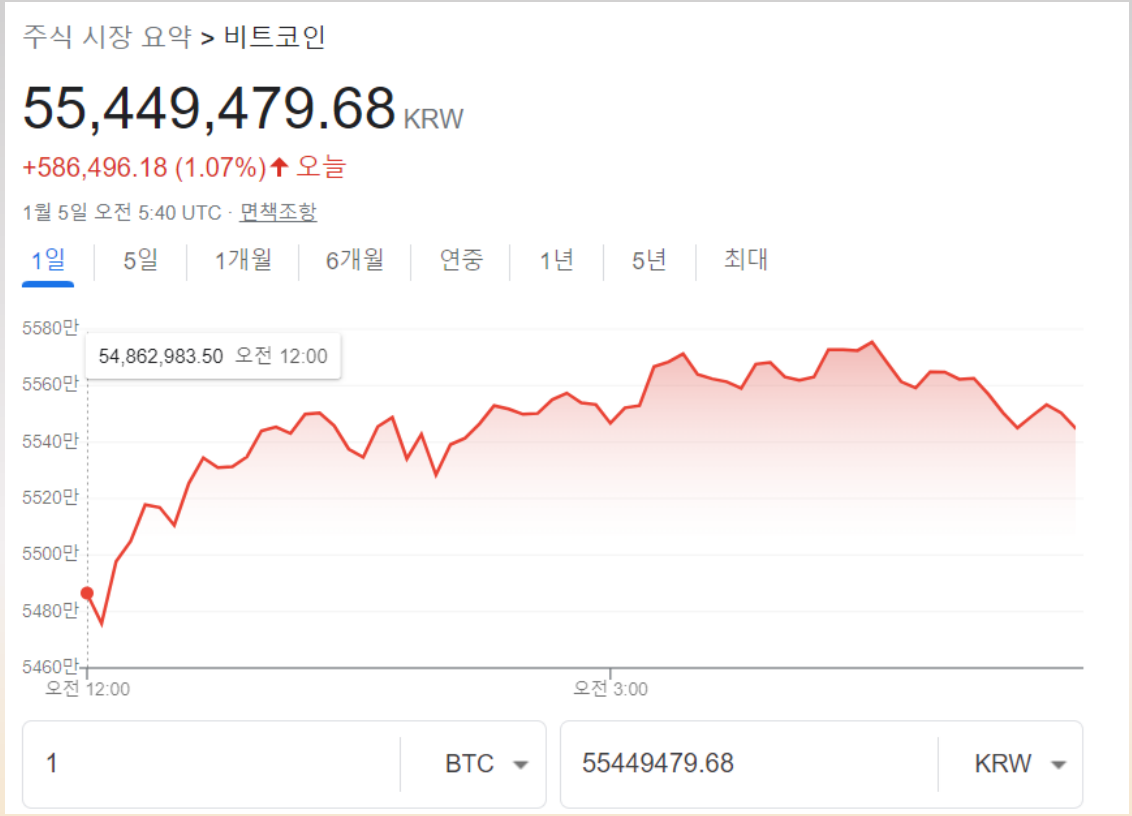
3팀 (삼삼오오)
김호준 김찬희 박유정 정새하 정한슬

목차

프로젝트 목표

비트코인 실시간으로 시세 분석하기

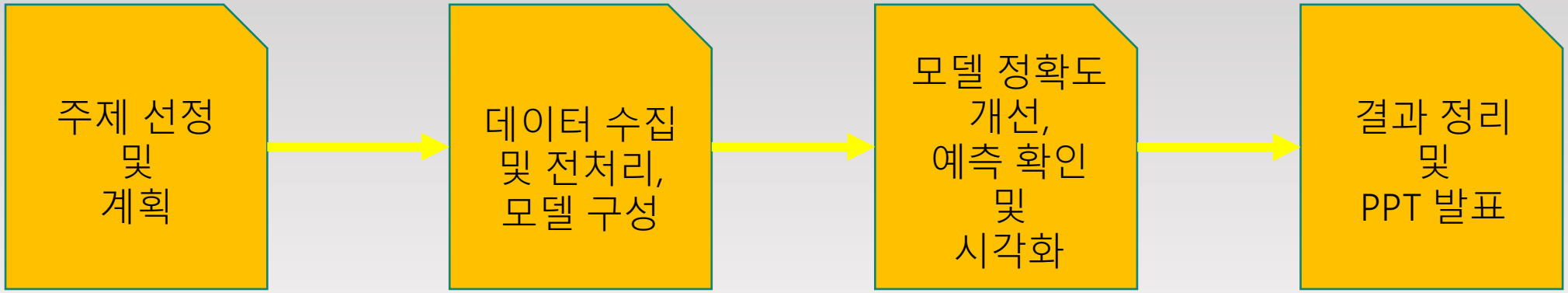
30일 동안의 데이터를 학습시켜
실시간 데이터를 예측하기



역할 분담

- 데이터 수집 — 정한슬
- 데이터 전처리 및 학습/테스트 데이터 분리 — 김호준
- LSTM 모델 설계 및 학습 — 정새하
- 예측 및 시각화 — 김찬희
- 모델 평가 및 결과 정리 — 박유정

일정



1/3 1/4 1/5~6 1/7

사용 기술

개발 환경 : Google Colaboratory

사용한 라이브러리 :

pandas

TensorFlow

NumPy

matplotlib

python

scikit learn

UPbit

사용 기술

디지털 자산 거래소 (코인 거래소)

REST API
Representational State Transfer API

API는 컴퓨터 기능을 실행시키는 방법/기계들의 통신방법을 의미

웹의 통신규약 HTTP를 이용한 API

HTTP 구성 = URL

```
url = "https://api.upbit.com/v1/candles/days?count=1"
```

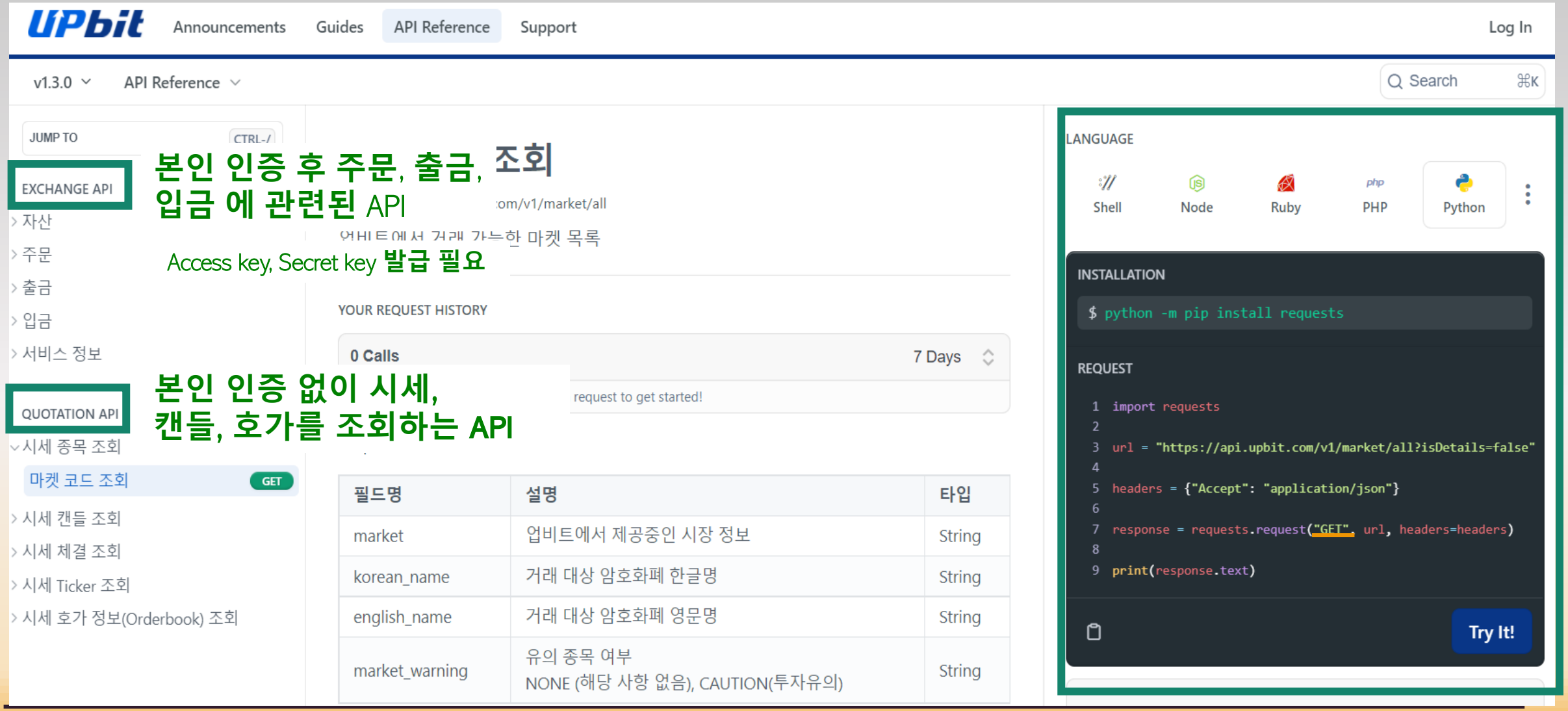
RESTful하다
: 정보가 주고받는 것을 인지하기 쉽게 표현 되어 있다.

HTTP의 METHOD(Protocol)

* POST	새로운 데이터 추가	C
* GET	데이터 조회	R
* PUT	데이터 전체 수정	U
* PATCH	데이터 일부 수정	D
* DELETE	정보 삭제	

```python
import requests

url = "https://api.upbit.com/v1/candles/days?count=1"

headers = {"Accept": "application/json"}

response = requests.request("GET", url, headers=headers)

print(response.text)
```

사용 기술


UPbit
Announcements
Guides
API Reference
Support
Log In
v1.3.0
API Reference
Search ⌘K
JUMP TO CTRL-/
EXCHANGE API
자산
주문
출금
입금
서비스 정보
본인 인증 후 주문, 출금, 조회
입금 에 관련된 API
Access key, Secret key 발급 필요
QUOTATION API
시세 종목 조회
마켓 코드 조회 GET
시세 캔들 조회
시세 체결 조회
시세 Ticker 조회
시세 호가 정보(Orderbook) 조회
본인 인증 없이 시세,
캔들, 호가를 조회하는 API
:om/v1/market/all
업비트에서 거래 가능한 마켓 목록
YOUR REQUEST HISTORY
0 Calls
7 Days
request to get started!
필드명
설명
타입
market
업비트에서 제공중인 시장 정보
String
korean_name
거래 대상 암호화폐 한글명
String
english_name
거래 대상 암호화폐 영문명
String
market_warning
유의 종목 여부
NONE (해당 사항 없음), CAUTION(투자유의)
String
LANGUAGE
Shell
Node
Ruby
PHP
Python
INSTALLATION
$ python -m pip install requests
REQUEST
import requests
url = "https://api.upbit.com/v1/market/all?isDetails=false"
headers = {"Accept": "application/json"}
response = requests.request("GET", url, headers=headers)
print(response.text)
Try It!

사용 기술

```python
import requests

url = "https://api.upbit.com/v1/candles/days?count=1"

headers = {"Accept": "application/json"}

response = requests.request("GET", url, headers=headers)

print(response.text)
```

RESPONSE

200 Log


```
[
  {
    "market": "KRW-BTC",
    "korean_name": "비트코인",
    "english_name": "Bitcoin"
  },
  {
    "market": "KRW-ETH",
    "korean_name": "이더리움",
    "english_name": "Ethereum"
  },
  {
    "market": "BTC-ETH",
    "korean_name": "이더리움",
    "english_name": "Ethereum"
  },
  {
    "market": "BTC-LTC",
    "korean_name": "라이트코인",
    "english_name": "Litecoin"
  },
  {
    "market": "BTC-XRP",
    "korean_name": "리플",
    "english_name": "Ripple"
  },
```

사용 기술

pyupbit

업비트 open API를 wrapping한 파이썬 라이브러리

```
import pyupbit

df = pyupbit.get_ohlcv("KRW-BTC")
print(df)
```

```
                          open        high  ...        volume         value
2021-06-20 09:00:00  42088000.0  42443000.0  ...  13184.495902  5.427276e+11
2021-06-21 09:00:00  41843000.0  42048000.0  ...  23102.561338  9.008327e+11
2021-06-22 09:00:00  37397000.0  39068000.0  ...  29986.869584  1.100786e+12
2021-06-23 09:00:00  37784000.0  40099000.0  ...  17696.395168  6.903951e+11
2021-06-24 09:00:00  39077000.0  40705000.0  ...  11788.006430  4.603288e+11
...                         ...         ...  ...           ...           ...
2022-01-01 09:00:00  56784000.0  58271000.0  ...   2628.145965  1.510645e+11
2022-01-02 09:00:00  57915000.0  58300000.0  ...   3567.505712  2.052913e+11
2022-01-03 09:00:00  57540000.0  57749000.0  ...   6304.958564  3.600635e+11
2022-01-04 09:00:00  56640000.0  57685000.0  ...   6053.445802  3.425126e+11
2022-01-05 09:00:00  56023000.0  56677000.0  ...   1331.615319  7.509916e+10

[200 rows x 6 columns]
```

interval 지정 가능 값	month week day minute1, minute3, minute5, minute10, minute15, minute30, minute60, minute240
count 지정 가능 값	숫자 integer (default 값: 200)

당일 하루 1분씩 조회

```
df1 = pyupbit.get_ohlcv("KRW-BTC", interval="minute1")
print(df1)
```

```
                          open        high  ...      volume         value
2022-01-05 10:40:00  56415000.0  56431000.0  ...    2.615698  1.475571e+08
2022-01-05 10:41:00  56425000.0  56483000.0  ...    3.137228  1.770816e+08
2022-01-05 10:42:00  56452000.0  56501000.0  ...    4.703617  2.655293e+08
2022-01-05 10:43:00  56500000.0  56509000.0  ...    5.590329  3.157452e+08
2022-01-05 10:44:00  56450000.0  56470000.0  ...    2.913548  1.644453e+08
...                         ...         ...  ...         ...           ...
2022-01-05 13:55:00  56420000.0  56443000.0  ...    0.973458  5.492561e+07
2022-01-05 13:56:00  56420000.0  56437000.0  ...    1.845705  1.041436e+08
2022-01-05 13:57:00  56417000.0  56437000.0  ...    2.890894  1.630872e+08
2022-01-05 13:58:00  56400000.0  56447000.0  ...    4.745861  2.677791e+08
2022-01-05 13:59:00  56446000.0  56477000.0  ...    3.087981  1.742442e+08

[200 rows x 6 columns]
```

코드

데이터 불러오기

```
import pandas as pd
import pyupbit

from datetime import datetime, time, date, timedelta
from calendar import monthrange
from time import sleep
```

```
year = 2021
month = 12
YYYYMM = str(year) + '{0:02d}'.format(month)

# 해당 년월 마지막 일(28일, 30일, 31일)
end_day = monthrange(year, month)[1]

my_ticker = "KRW-BTC"
my_interval = "minutes1"

now = datetime.now()
```

```
def get_upbit_ohlcv(now, ticker, year, month):
    df = pd.DataFrame(columns=['open', 'high', 'low', 'close', 'volume'])

    # 해당 년월 1일부터
    from_date = date(year, month, 1)

    # 해당 년월 마지막 일(28일, 30일, 31일)
    end_day = monthrange(year, month)[1]
    to_date = date(year, month, end_day)

    # 해당 년월 마지막 일자가 현재 프로그램 수행일자보다 큰 경우
    if to_date >= now.date():
        to_date = now.date()
        end_day = to_date.day

    temp_list = []
    # 해당 년월 1일부터 말일(또는 프로그램 수행일자)까지 데이터 수집 실시
    for day in range(1, end_day+1):
        cnt = 200 # default
        base_time = datetime.combine(from_date, time(3, 20, 0))

        for i in range(8):
            try:
                df_temp = pyupbit.get_ohlcv(ticker,
                                            interval='minute1', #1분당 데이터 수집
                                            count=cnt, to=base_time)

                df = pd.concat([df, df_temp], axis=0)
                if i == 6:
                    base_time += timedelta(hours=0, minutes=40)
                else:
                    base_time += timedelta(hours=3, minutes=20)
            except Exception as e:
                print('Exception:', e)

        from_date = from_date + timedelta(days=1)
        sleep(0.5)

    return df
```

코드

데이터 불러오기

```
df = get_upbit_ohlcv(now, ticker=my_ticker, year=year, month=month)
df
```

	open	high	low	close	volume	value
2021-12-01 09:00:00	70771000.0	70822000.0	70700000.0	70719000.0	9.234089	6.532091e+08
2021-12-01 09:01:00	70743000.0	70779000.0	70714000.0	70714000.0	9.732817	6.883343e+08
2021-12-01 09:02:00	70706000.0	70717000.0	70553000.0	70553000.0	17.325307	1.223836e+09
2021-12-01 09:03:00	70556000.0	70567000.0	70507000.0	70521000.0	15.076560	1.063433e+09
2021-12-01 09:04:00	70526000.0	70630000.0	70511000.0	70616000.0	12.208826	8.615437e+08
...	...	...	...	...	...	...
2022-01-01 08:55:00	56820000.0	56821000.0	56768000.0	56768000.0	2.444209	1.387907e+08
2022-01-01 08:56:00	56768000.0	56775000.0	56762000.0	56769000.0	2.030722	1.152749e+08
2022-01-01 08:57:00	56766000.0	56776000.0	56763000.0	56770000.0	4.879123	2.769724e+08
2022-01-01 08:58:00	56766000.0	56770000.0	56668000.0	56740000.0	12.165039	6.898896e+08
2022-01-01 08:59:00	56701000.0	56785000.0	56700000.0	56784000.0	5.362344	3.042829e+08

49600 rows × 6 columns

코드

데이터 전처리 및 분리

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 8861 entries, 2021-12-01
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   open    8861 non-null   float64
 1   high    8861 non-null   float64
 2   low     8861 non-null   float64
 3   close   8861 non-null   float64
 4   volume  8861 non-null   float64
 5   value   8861 non-null   float64
dtypes: float64(6)
memory usage: 484.6 KB
```

```
from sklearn.preprocessing import MinMaxScaler
# feature = ['open', 'high', 'low', 'volume', 'value'] label = ['close']
scaler = MinMaxScaler()
# 스케일을 적용할 column을 정의합니다.
scale_cols = ['open', 'high', 'close', 'low', 'volume', 'value']
# 스케일 후 columns
scaled = scaler.fit_transform(df[scale_cols])
scaled
```

```
df2 = pd.DataFrame(scaled, columns= scale_cols) # x_train data
```

```
x_train = df2.drop('close',1) # feature = ['open', 'high', 'low', 'volume', 'value']
```

```
y_train = df2['close'] # label = ['close']
```

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

```
((8861, 5), (8861,))
```

코드

데이터 전처리 및 분리

```
df_valid = get_upbit_ohlcv(now, ticker=my_ticker, year=2022, month=1)
df_valid
```

	open	high	low	close	volume	value
2022-01-01 09:00:00	56784000.0	56784000.0	56762000.0	56784000.0	7.226950	4.102409e+08
2022-01-01 09:01:00	56784000.0	56876000.0	56764000.0	56811000.0	7.188775	4.081979e+08
2022-01-01 09:02:00	56876000.0	56941000.0	56814000.0	56890000.0	5.121608	2.913801e+08
2022-01-01 09:03:00	56881000.0	56916000.0	56848000.0	56869000.0	6.890893	3.919717e+08
2022-01-01 09:04:00	56892000.0	56895000.0	56833000.0	56868000.0	3.857810	2.194228e+08
...	...	...	...	...	...	...
2022-01-05 13:57:00	56417000.0	56437000.0	56400000.0	56402000.0	2.890894	1.630872e+08
2022-01-05 13:58:00	56400000.0	56447000.0	56383000.0	56447000.0	4.745861	2.677791e+08
2022-01-05 13:59:00	56446000.0	56477000.0	56384000.0	56449000.0	4.542710	2.563593e+08
2022-01-05 14:00:00	56448000.0	56474000.0	56403000.0	56449000.0	3.961444	2.236209e+08
2022-01-05 14:01:00	56449000.0	56478000.0	56449000.0	56449000.0	1.562240	8.820781e+07

8000 rows × 6 columns

```
# feature = ['open', 'high', 'low', 'volume', 'value'] label = ['close']
scaler = MinMaxScaler()
# 스케일을 적용할 column을 정의합니다.
scale_cols_f = ['open', 'high', 'close', 'low', 'volume', 'value']
# 스케일 후 columns
scaled_f = scaler.fit_transform(df_valid[scale_cols])
scaled_f
```

```
scaled_f = scaler.fit_transform(df_valid[scale_cols])

df2_f = pd.DataFrame(scaled_f, columns= scale_cols_f) # x_train data

x_test = df2_f.drop('close',1)

y_test = df2_f['close'] # label = ['close']

x_test.shape, y_test.shape
```

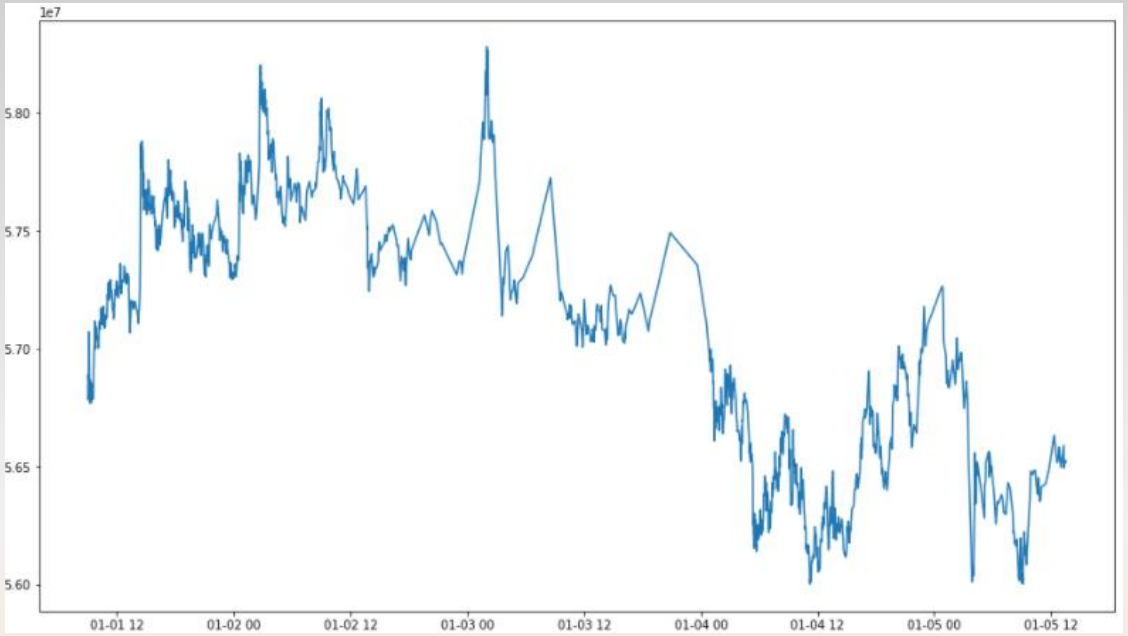
```
((1697, 5), (1697,))
```

코드

데이터 전처리 시각화

Y:price X:time

훈련데이터 21/12~22/01/01

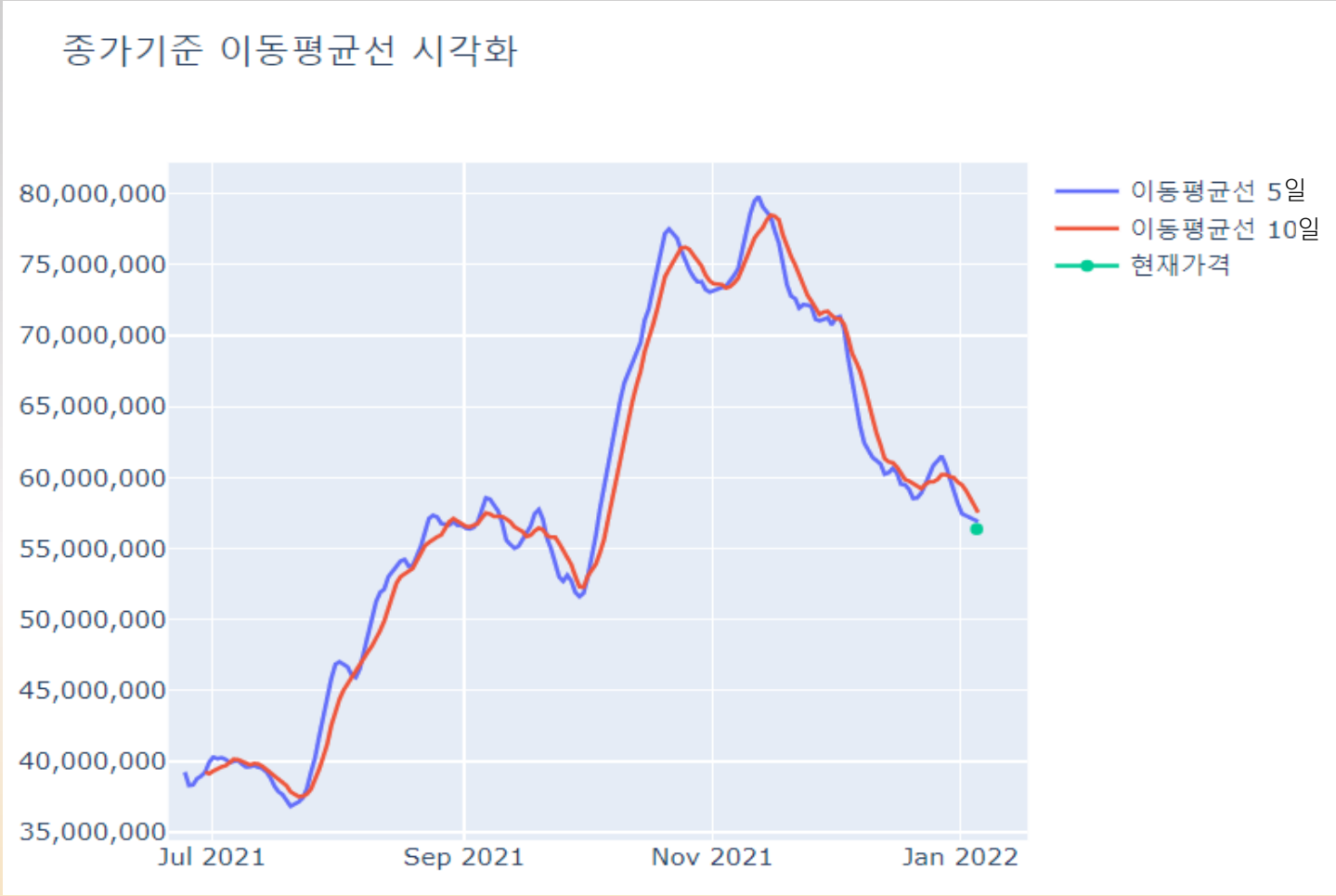


검증데이터 22/01/01~01/05

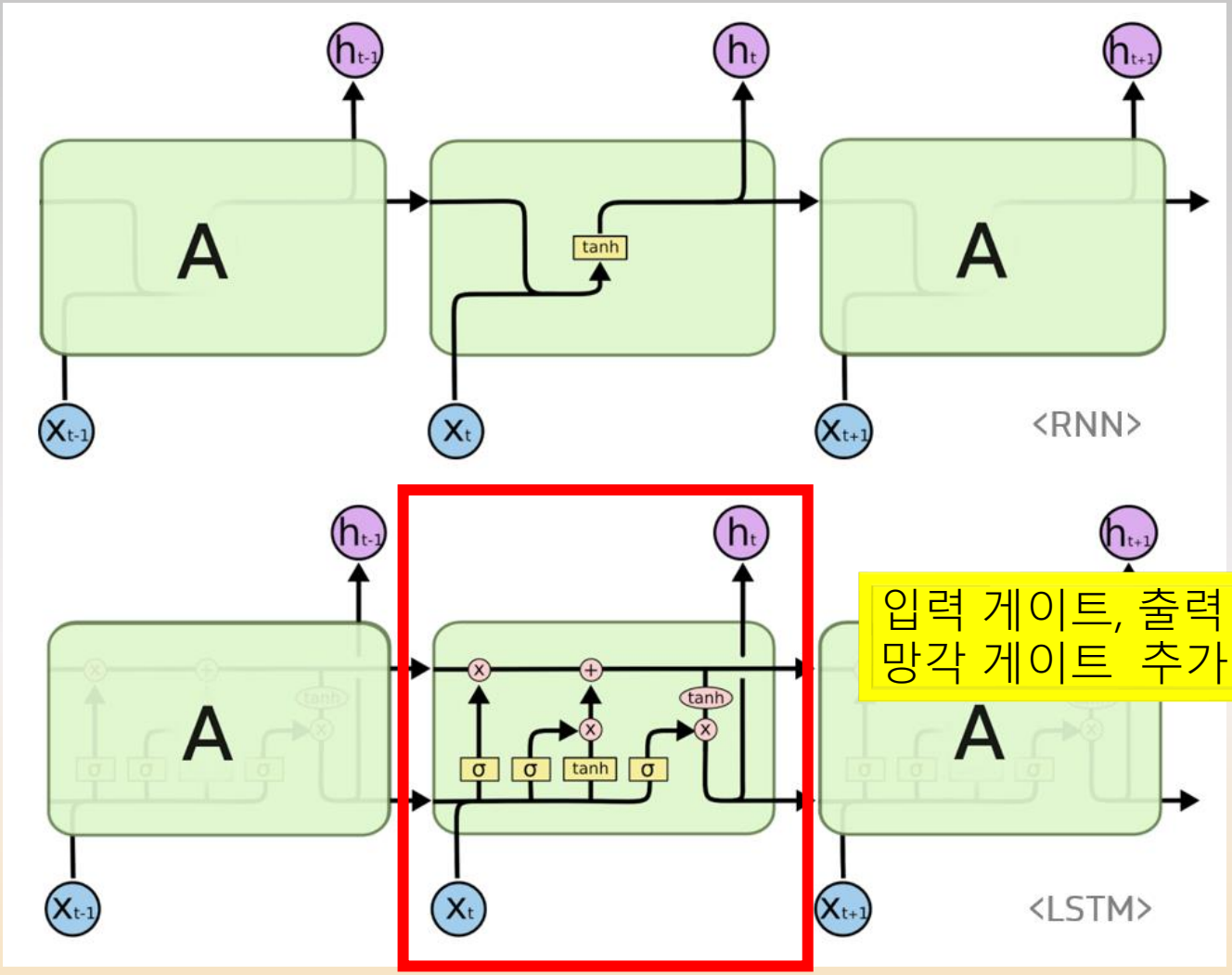
Train : Test = 0.84 : 0.16

코드

데이터 전처리 시각화



LSTM이란?



기울기 소실
Vanishing Gradient 문제 발생

입력 게이트, 출력 게이트,
망각 게이트 추가

코드

LSTM모델 설계 및 학습

```
WINDOW_SIZE=30 #얼마동안의 기간을 학습해서 예측할 것인지
BATCH_SIZE=32
```

```
def windowed_dataset(x, y, window_size, batch_size, shuffle):

# X값 window dataset 구성
    ds_x = tf.data.Dataset.from_tensor_slices(x)
    ds_x = ds_x.window(window_size, shift=1, stride=1, drop_remainder=True)
    ds_x = ds_x.flat_map(lambda x: x.batch(window_size))

# y값 추가
    ds_y = tf.data.Dataset.from_tensor_slices(y[window_size:])
    ds = tf.data.Dataset.zip((ds_x, ds_y))
    if shuffle:
        ds = ds.shuffle(1000)
    return ds.batch(batch_size).prefetch(1)
```

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

Tendroflow dataset - window

Tensorflow dataset을 이용하면 다양한 Dataset loader 생성

window를 이용하면 시계열 데이터를 더 용이하게 처리 가능

- Window size : 해당 사이즈 만큼의 과거 데이터를 가지고 예측할 것인지 설정
- drop_remainder : 남은 데이터를 버리고 남길지 여부(True/False)
- Shift : 1iteration당 몇 개씩 이동할 것인지 설정
- stride :요소 간격

코드

LSTM모델 설계 및 학습

```
model = keras.Sequential([
# 1차원 feature map 생성
    keras.layers.Conv1D(filters=32, kernel_size=5,
           padding="causal",
           activation="relu",
           input_shape=[WINDOW_SIZE, 5]),
# LSTM
    keras.layers.LSTM(16, activation='tanh'),
    keras.layers.Dense(16, activation="relu"),
    keras.layers.Dense(1),
])

# Sequence 학습에 비교적 좋은 퍼포먼스를 내는 Huber()사용
loss = keras.losses.Huber()
optimizer = keras.optimizers.Adam(0.0005)
model.compile(loss=loss, optimizer=optimizer, metrics=['mse'])
```

```
# earlystopping은 35번 epoch통안 val_loss 개선이 없다면 학습 멈춤
earlystopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=35)

  history = model.fit(train_data,
                      validation_data=(test_data),
                      epochs=100,
                      callbacks=[earlystopping])
```

```
Epoch 14/100
276/276 [==============================] - 2s 5ms/step - loss: 5.9958e-05 -
Epoch 15/100
276/276 [==============================] - 2s 5ms/step - loss: 5.7375e-05 -
Epoch 16/100
276/276 [==============================] - 2s 5ms/step - loss: 5.6502e-05 -
Epoch 17/100
276/276 [==============================] - 2s 5ms/step - loss: 5.6355e-05 -
Epoch 18/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3554e-05 -
Epoch 19/100
276/276 [==============================] - 2s 5ms/step - loss: 5.5489e-05 -
Epoch 20/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3709e-05 -
Epoch 21/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3891e-05 -
```

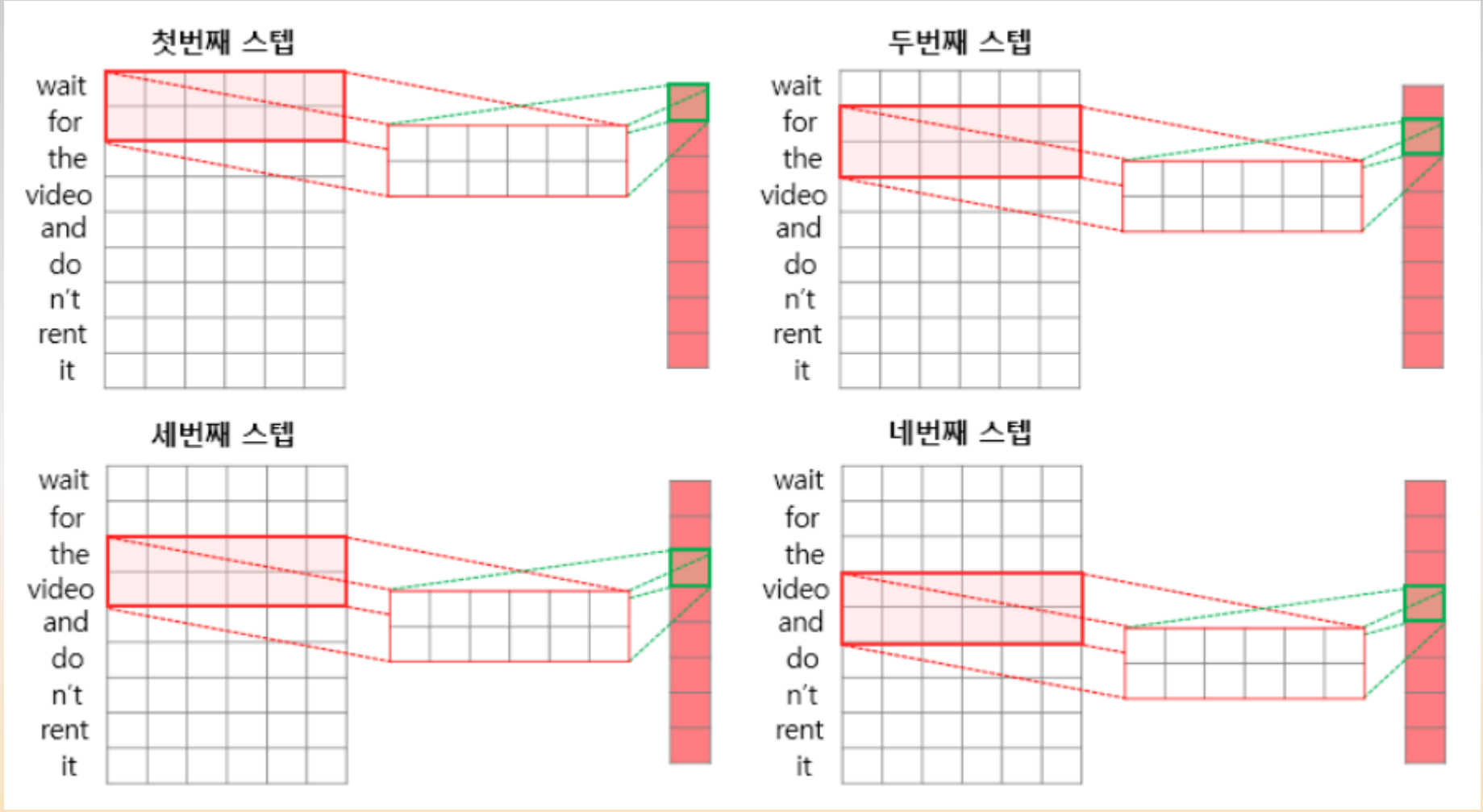
```
model.save('./bitcoin_model1.h5')
```

Conv1D

- 2차원CNN이 이미지에서 특징을 추출하 듯 1차원 CNN은 이미지가 아닌 시계열분석이나 텍스트 분석에서 특징 추출에 이용

- Sequence 데이터에서 중요한 정보를 추출하는 용도로 많이 사용 됨.
- RNN 모델을 생성할 때 conv1D 층이 쓰이는 경우가 종종 있음.

Conv1D의 연산 모습



Huber

제곱 오차 손실보다 데이터의 특이치에 덜 민감하게 반응하는 강력한 Regression또는 Classification에 사용되는 손실 함수

- Qingsong Wen **알리바바 - 알고리즘 엔지니어**

시계열에는 이상값뿐만 아니라 실제 시나리오의 급격한 추세 변화도 포함될 수 있습니다.

이러한 문제를 해결하기 위해 강력한 통계 및 희소 학습을 기반으로 하는 강력한 추세 필터링 알고리즘을 제안합니다.

특히 Huber loss를 채택하여 이상값을 억제하고 추세 구성 요소의 1차 및 2차 차이 조합을 정규화로 활용하여 느리고 급격한 추세 변화를 모두 포착합니다.

인용출처 :https://www.researchgate.net/publication/334844444_RobustTrend_A_Huber_Loss_with_a_Combined_First_and_Second_Order_Difference_Regularization_for_Time_Series_Trend_Filtering

코드

LSTM모델 설계 및 학습

```
model = keras.Sequential([
# 1차원 feature map 생성
    keras.layers.Conv1D(filters=32, kernel_size=5,
           padding="causal",
           activation="relu",
           input_shape=[WINDOW_SIZE, 5]),
# LSTM
    keras.layers.LSTM(16, activation='tanh'),
    keras.layers.Dense(16, activation="relu"),
    keras.layers.Dense(1),
])

# Sequence 학습에 비교적 좋은 퍼포먼스를 내는 Huber()사용
loss = keras.losses.Huber()
optimizer = keras.optimizers.Adam(0.0005)
model.compile(loss=loss, optimizer=optimizer, metrics=['mse'])
```

```
# earlystopping은 35번 epoch통안 val_loss 개선이 없다면 학습 멈춤
earlystopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=35)

  history = model.fit(train_data,
                      validation_data=(test_data),
                      epochs=100,
                      callbacks=[earlystopping])
```

```
Epoch 14/100
276/276 [==============================] - 2s 5ms/step - loss: 5.9958e-05 -
Epoch 15/100
276/276 [==============================] - 2s 5ms/step - loss: 5.7375e-05 -
Epoch 16/100
276/276 [==============================] - 2s 5ms/step - loss: 5.6502e-05 -
Epoch 17/100
276/276 [==============================] - 2s 5ms/step - loss: 5.6355e-05 -
Epoch 18/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3554e-05 -
Epoch 19/100
276/276 [==============================] - 2s 5ms/step - loss: 5.5489e-05 -
Epoch 20/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3709e-05 -
Epoch 21/100
276/276 [==============================] - 2s 5ms/step - loss: 5.3891e-05 -
```

```
model.save('./bitcoin_model1.h5')
```

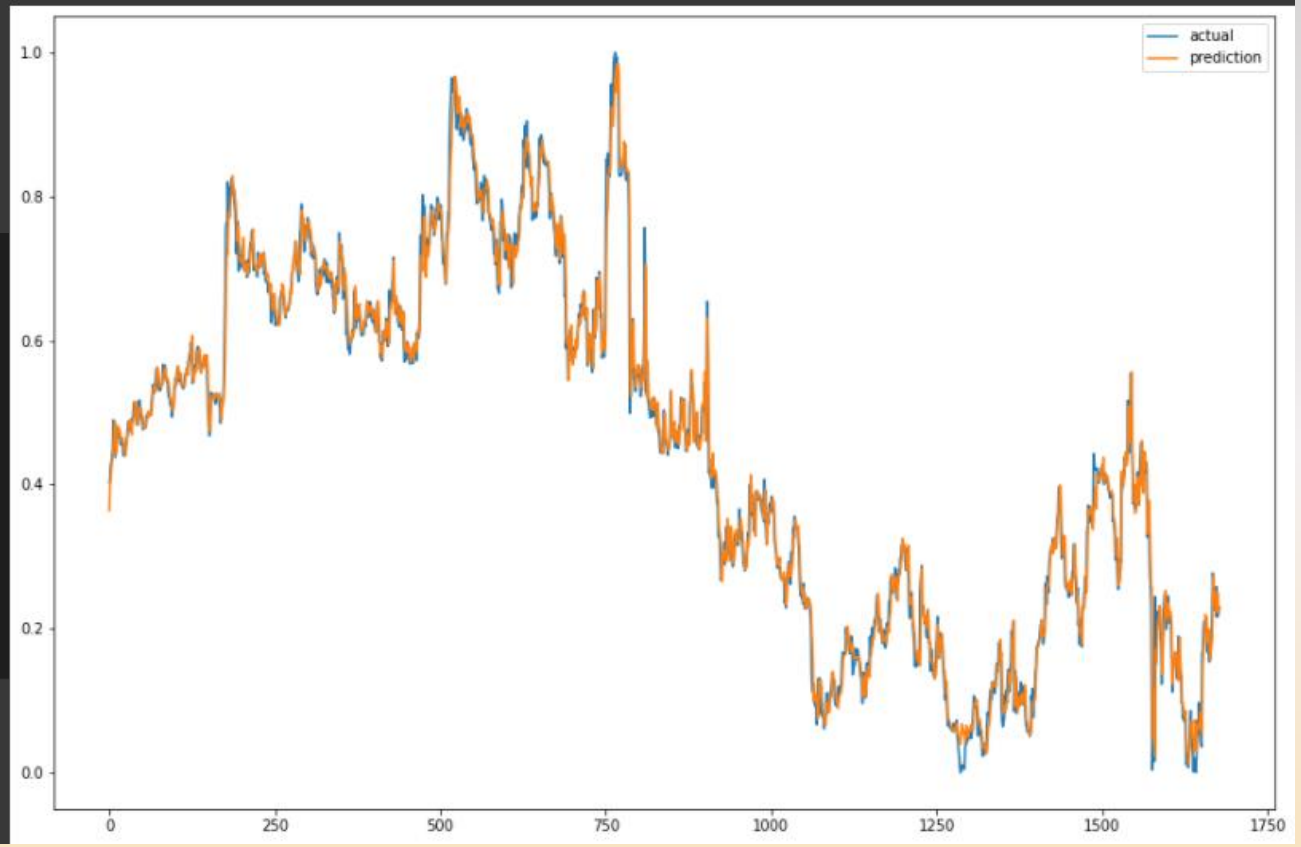
코드

예측 및 시각화

```
print("%.7f" % (float(min(history.history['val_loss']))))

model.load_weights('./bitcoin_model1.h5')
pred = model.predict(test_data)
actual = np.asarray(y_test)[WINDOW_SIZE:]
actual = np.reshape(actual, (len(actual), 1))

print(pred.shape)
print(actual.shape)
```

```
0.0003847
(1678, 1)
(1678, 1)
```

```
plt.figure(figsize=(15,10))
plt.plot(actual, label='actual')
plt.plot(pred, label='prediction')
plt.legend()
plt.show()
```



코드

예측 및 시각화

현 30분간의 누적 데이터로 1분 후 데이터 예측

```
x_realtime = x_test.iloc[-30:,:]
y_realtime = y_test.iloc[-30:,]

live_data = windowed_dataset(x_realtime, y_realtime , 29 , 32, False)
live_pred = model.predict(live_data)

trainPredict_dataset_like = np.zeros(shape=(len(live_pred), 6) )
trainPredict_dataset_like[:,0] = live_pred[:,0]
real_pred = scaler.inverse_transform(trainPredict_dataset_like)[:,0]

print("예측 close:",real_pred[0])

# 13:30 기준
```

```
예측 close: 56511171.97185755
```

오차율 : 0.00677

코드

모델 평가

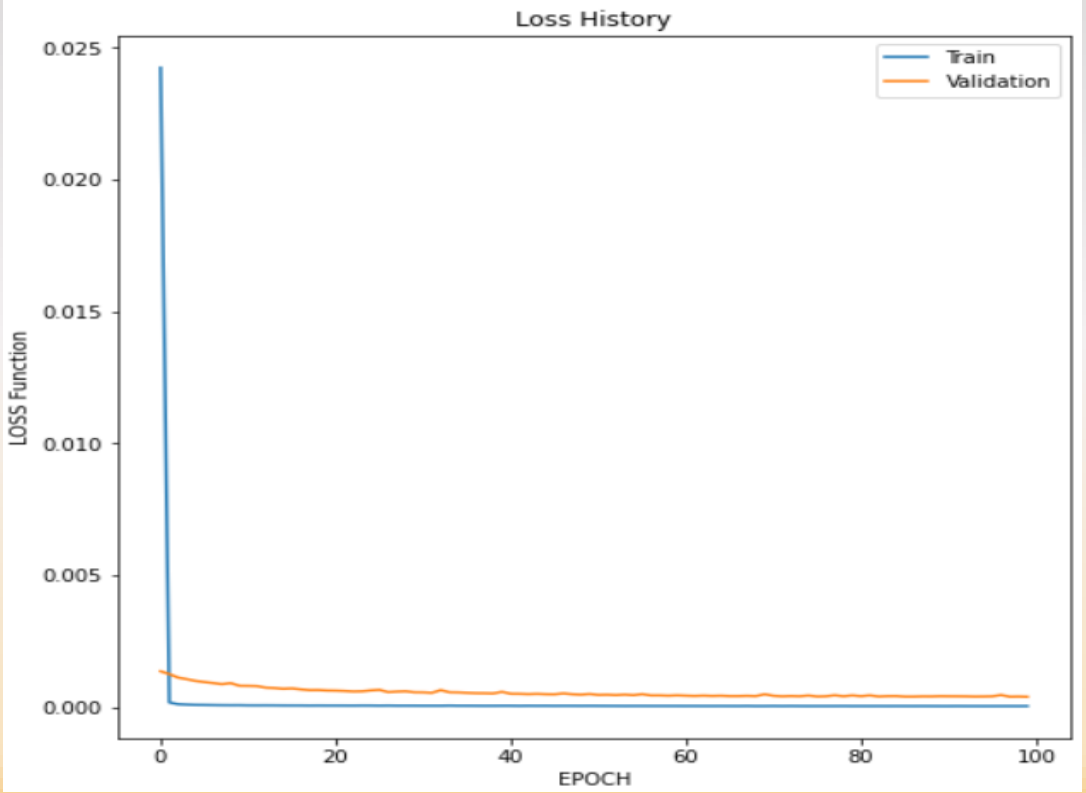
```
ev = model.evaluate(train_data, batch_size=32)
print("손실 값", ev[0], "MSE:",ev[1])
```

```
276/276 [==============================] - 1s 4ms/step - l
손실 값 4.1978586523327976e-05 MSE: 8.395717304665595e-05
```

```
ev= model.evaluate(test_data, batch_size=32)
print("손실 값", ev[0], "MSE:",ev[1])
```

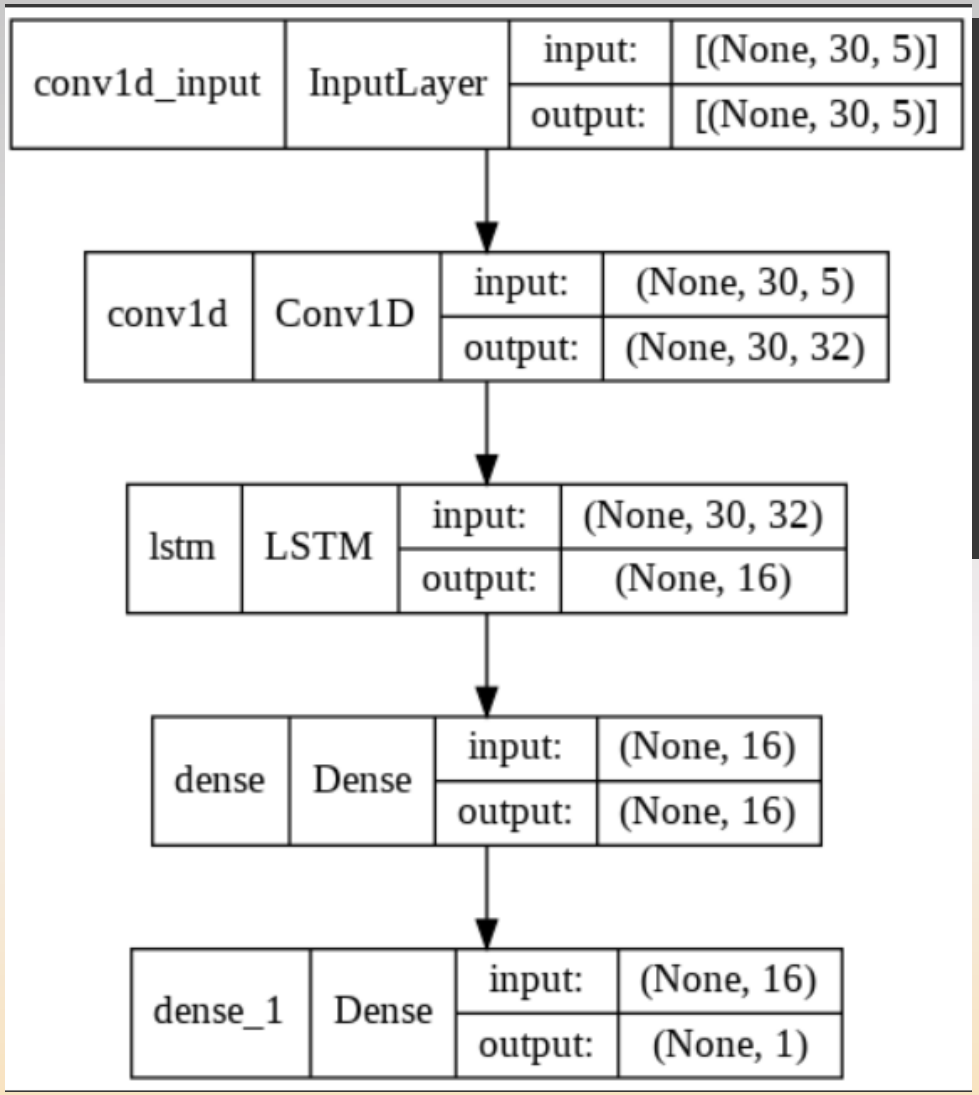
```
53/53 [==============================] - 0s 4ms/step - lo
손실 값 0.0003847014158964157 MSE: 0.0007694028317928314
```

```
train_history = history.history["loss"]
validation_history = history.history["val_loss"]
fig = plt.figure(figsize=(8, 8))
plt.title("Loss History")
plt.xlabel("EPOCH")
plt.ylabel("LOSS Function")
plt.plot(train_history, label = 'Train')
plt.plot(validation_history,label = 'Validation' )
plt.legend()
```



코드

모델 평가




```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 30, 32)            832

 lstm (LSTM)                 (None, 16)                3136

 dense (Dense)               (None, 16)                272

 dense_1 (Dense)             (None, 1)                 17

=================================================================
Total params: 4,257
Trainable params: 4,257
Non-trainable params: 0
_________________________________________________________________
```

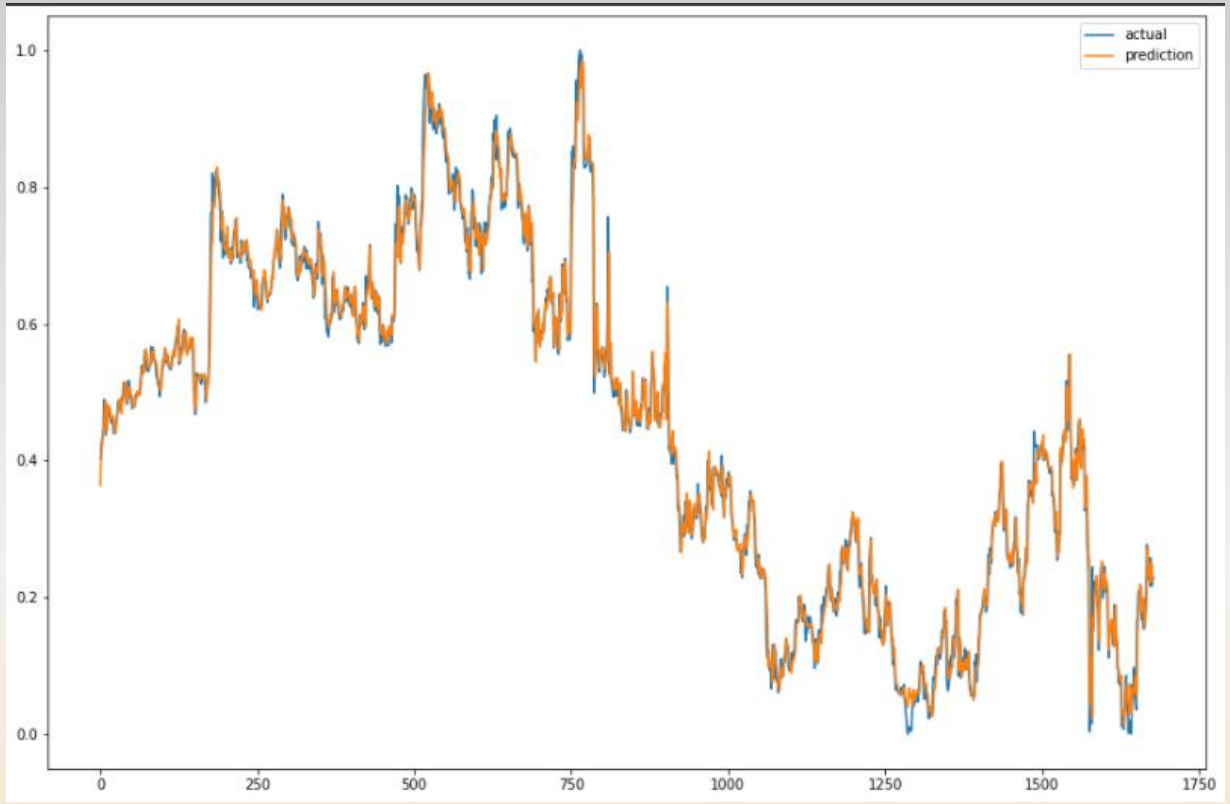
Layer(type) : 레이어 이름과 타입

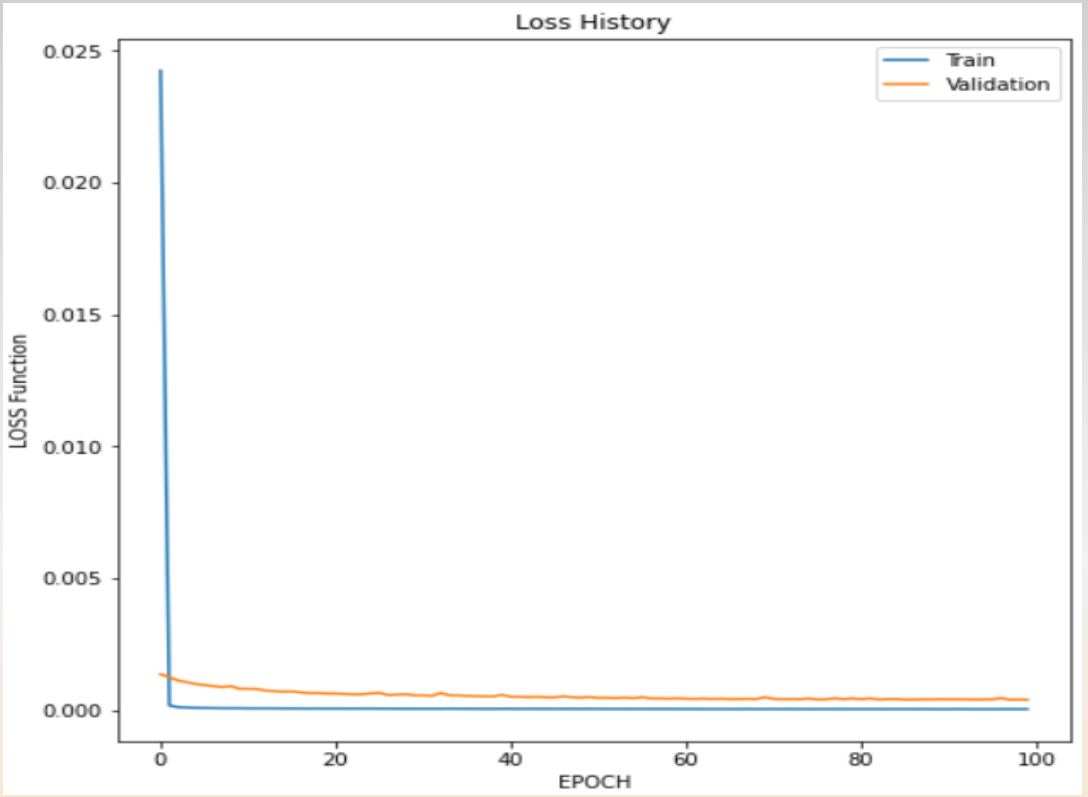
Output Shape: 아웃풋 개수

행이 None으로 지정된 이유:
데이터의 갯수는 계속 추가가 될 수 있기 때문에
딥러닝 모델에서는 주로 행을 무시

Param:: 입력과 출력노드에 연결된 간선의 수

결과


actual
prediction
1.0
0.8
0.6
0.4
0.2
0.0
0
250
500
750
1000
1250
1500
1750


Loss History
Train
Validation
LOSS Function
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
EPOCH

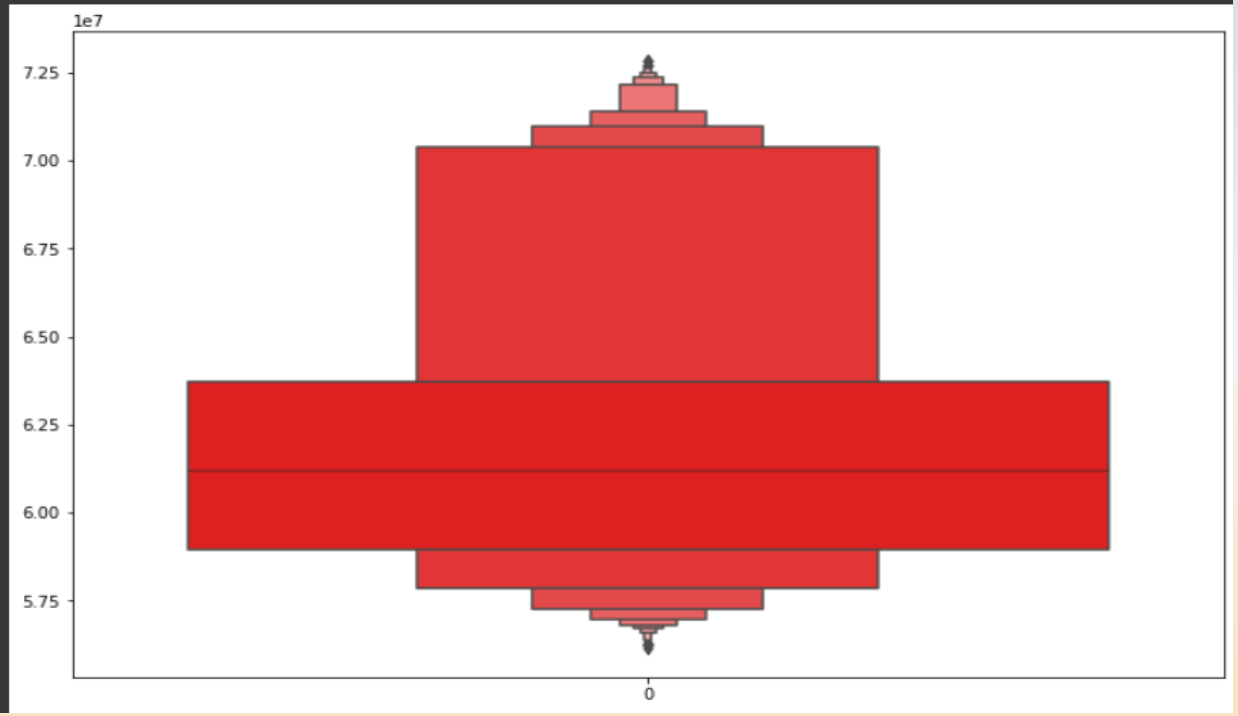
결과


1e7
7.25
7.00
6.75
6.50
6.25
6.00
5.75
5.50
y
2021-12-05
2021-12-12
2021-12-19
2021-12-26
2022-01-02
ds

보완할 점

<이상치 제거>

```
[ ]    # 이상치 boxplot
       plt.figure(figsize=(12,8))
       sns.boxenplot(data=df['close'], color='red')
       plt.show()
```



Boxplot 확인 결과

연속형 데이터로 확인

소감

김호준

처음으로 팀원과 협업을 통해 데이터를 전처리 하고,
딥러닝 모델을 만들어 시각화 까지 해보니 매우 뿌듯했습니다.
또한 모델의 예측결과까지 좋게 나와 만족스럽고 1차원 lstm에 대한 이해가 조금 생긴 것 같습니다.

김찬희

'이게 될까' 싶었던 것들이 저와 팀원들의 손을 거쳐 학습되고 예측되는 것을 보면서
딥러닝에 대한 흥미와 관심이 많이 생겼습니다. 앞으로 다양한 주제로 실력을 향상시키고 싶습니다.

박유정

혼자하면 오래 걸리는 부분을 팀원들과의 협업으로 빠른 시일내에 결과를 도출 할 수 있었습니다.
변수들간의 관계파악, 생소한 주식용어들을 접하며 도메인 지식이 우선시 돼야 한다는 걸 다시 한번 깨달았습니다.

정새하

얕게 알고 있던 부분을 좀 더 정확하게 공부할 수 있는 계기가 되었습니다.
팀원들이 해나가는 모습과 작업을 보면서 제가 알지 못했던 부분을 알고 배울 수 있어서 좋았습니다.
앞으로 좀 더 실력을 키우고 싶습니다.

정한슬

이론 수업때 미처 소화시키지 못한 부분들을 프로젝트를 통해서 공부할 수 있어서 좋았습니다.
팀원들과 모르는 부분은 서로 보완해주고 회의를 통해 의견을 맞춰가는게 뜻깊었습니다.

참고 자료

코드 분석 참조

https://superhky.tistory.com/154

https://mskim8717.tistory.com/104

https://mooyoungblog.tistory.com/114

로고

https://www.tensorflow.org

https://scikit-learn.org/stable/

https://pandas.pydata.org/

https://numpy.org/

https://matplotlib.org/

https://docs.upbit.com/reference ,https://www.python.org/

감사합니다.

Q & A