

딥러닝 시계열 모델을 이용한 가상화폐 가격 예측

3조 / 김선일, 유현민, 한상안



Contents



Phase 01

분석 필요성 및 목적



Phase 02

분석 과정 및 방법

- 분석 개요
- 사용 데이터 및 전처리
- 모델 설계 및 학습
- 매매 프로그램 설계



Phase 03

결과

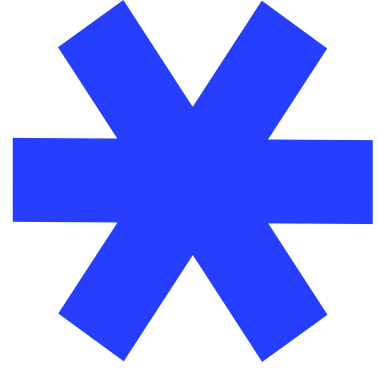
Contents



Phase 01

분석 필요성 및 목적

01. 분석 필요성 및 목적



경제·금융 > 금융가

암호화폐 하루 거래액, 코스피 넘어섰다

입력 2021-03-15 06:20:16 수정 2021.03.15 06:20:16 김지영 기자 jikim@sedaily.com



비트코인 개당 7,000만원 돌파 속
국내 4곳서 총 16.7조 거래 신기록
인플레 우려에 안전 투자처 기대감



비트코인이 7,100만원을 돌파하며 최고치를 경신한 14일 서울 강남구 빗썸 강남고객센터에서 빗썸 관계자가 시황판을 바라보고 있다./성형주기자

- 암호화폐는 매일 400억 달러 이상이 거래되고 있다.
- 하지만, 매우 불안정하여 예측하기 어렵다.
- 본 분석은 딥러닝 모델을 활용하여 암호화폐의 가격을 예측하고, 투자전략을 통해 암호화폐의 수익성을 분석한다.
- 이후 다양한 모형들의 성과 비교를 통해 최적의 예측 모형을 개발하고자 한다

Contents



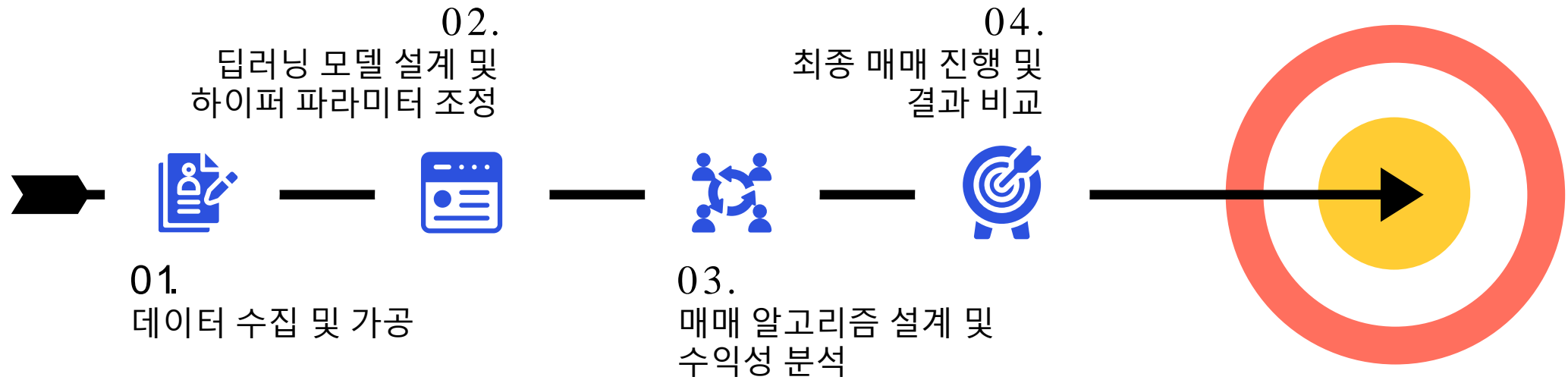
Phase 02

분석 과정 및 방법

- 분석 개요
- 사용 데이터 및 전처리
- 모델 설계 및 학습
- 매매 프로그램 설계

02. 분석 과정 및 방법

- 분석 개요



02. 분석 과정 및 방법



- 사용 데이터 및 전처리

1) Kaggle에서 제공하는 14개 Asset의 데이터 활용

| 1 | df1.head() | | | | | | | | | |
|---|------------|----------|-------|------------|------------|------------|------------|-------------|--------------|-----------|
| | timestamp | Asset_ID | Count | Open | High | Low | Close | Volume | VWAP | Target |
| 0 | 1514764860 | 2 | 40.0 | 2376.5800 | 2399.5000 | 2357.1400 | 2374.5900 | 19.233005 | 2373.116392 | -0.004218 |
| 1 | 1514764860 | 0 | 5.0 | 8.5300 | 8.5300 | 8.5300 | 8.5300 | 78.380000 | 8.530000 | -0.014399 |
| 2 | 1514764860 | 1 | 229.0 | 13835.1940 | 14013.8000 | 13666.1100 | 13850.1760 | 31.550062 | 13827.062093 | -0.014643 |
| 3 | 1514764860 | 5 | 32.0 | 7.6596 | 7.6596 | 7.6567 | 7.6576 | 6626.713370 | 7.657713 | -0.013922 |
| 4 | 1514764860 | 7 | 5.0 | 25.9200 | 25.9200 | 25.8740 | 25.8770 | 121.087310 | 25.891363 | -0.008264 |

0~13 총 14개의 가상화폐 종류

02. 분석 과정 및 방법



- 사용 데이터 및 전처리

2) 각 가상화폐 별 시작하는 날짜와 제공되는 데이터 시간이 다름

```
1 # check time start by asset
2 df_change.reset_index().groupby('Asset_ID')['timestamp'].min()
```

```
Asset_ID
0    2018-01-01 00:01:00
1    2018-01-01 00:01:00
2    2018-01-01 00:01:00
3    2018-04-17 09:11:00
4    2019-04-12 14:34:00
5    2018-01-01 00:01:00
6    2018-01-01 00:01:00
7    2018-01-01 00:01:00
8    2018-05-09 08:07:00
9    2018-01-01 00:01:00
10   2018-05-10 15:21:00
11   2018-01-01 00:01:00
12   2018-02-16 23:53:00
13   2018-02-06 21:37:00
Name: timestamp, dtype: datetime64[ns]
```

```
1 # check
2 for i in range(14):
3     print(i, df_change[df_change.Asset_ID == i].shape)

0 (1942619, 10)
1 (1956282, 10)
2 (1953537, 10)
3 (1791867, 10)
4 (1156866, 10)
5 (1955140, 10)
6 (1956200, 10)
7 (1951127, 10)
8 (1592071, 10)
9 (1956030, 10)
10 (670497, 10)
11 (1701261, 10)
12 (1778749, 10)
13 (1874560, 10)
```


02. 분석 과정 및 방법

- 사용 데이터 및 전처리

- 3) 각 가상화폐 시작일 과 제공되는 데이터 단위시간 통일

```
1 fittime_df = df_change[df_change.index >= '2019-04-12 14:34:00']
2 fittime_df.head(2)
```

←시작일 통일

| | Asset_ID | Count | Open | High | Low | Close | Volume | VWAP | Target3 |
|---------------------|----------|-------|------------|------------|------------|------------|------------|------------|-----------|
| timestamp | | | | | | | | | |
| 2019-04-12 14:34:00 | 2 | 76.0 | 282.343323 | 283.540009 | 280.640015 | 282.480011 | 143.997940 | 282.396240 | -0.003690 |
| 2019-04-12 14:35:00 | 2 | 57.0 | 282.776672 | 283.790009 | 281.029999 | 282.873322 | 85.751717 | 282.843658 | -0.001639 |

```
1 # check time start by asset
2 fittime_df.reset_index().groupby('Asset_ID')['timestamp'].min()
3
4 # asset10!!
```

←단위 시간이 다름으로 인한 시간 차이

```
Asset_ID
0    2019-04-12 14:34:00
1    2019-04-12 14:34:00
2    2019-04-12 14:34:00
3    2019-04-12 14:34:00
4    2019-04-12 14:34:00
5    2019-04-12 14:34:00
6    2019-04-12 14:34:00
7    2019-04-12 14:34:00
8    2019-04-12 14:34:00
9    2019-04-12 14:34:00
10   2019-04-12 14:41:00
11   2019-04-12 14:34:00
12   2019-04-12 14:34:00
13   2019-04-12 14:34:00
Name: timestamp, dtype: datetime64[ns]
```

```
1 fulltable_df = pd.DataFrame()
2 for i in df_change.Asset_ID.unique():
3     tmp_df = df_change[df_change.Asset_ID == i].resample(rule='T')
4     tmp_df = tmp_df.interpolate(method='linear')
5     fulltable_df = pd.concat([fulltable_df, tmp_df])
6
7 fulltable_df.head()
```

← 보간법을 이용한
제공 데이터 통일

| | Asset_ID | Count | Open | High | Low | Close | Volume | VWAP | Target3 |
|---------------------|----------|-------|-------------|-------------|-------------|-------------|-----------|-------------|-----------|
| timestamp | | | | | | | | | |
| 2018-01-01 00:01:00 | 2.0 | 40.0 | 2376.580078 | 2399.500000 | 2357.139893 | 2374.590088 | 19.233006 | 2373.116455 | -0.004218 |
| 2018-01-01 00:02:00 | 2.0 | 53.0 | 2374.553223 | 2400.899902 | 2354.199951 | 2372.286621 | 24.050259 | 2371.434570 | -0.004079 |
| 2018-01-01 00:03:00 | 2.0 | 61.0 | 2371.633301 | 2401.899902 | 2353.699951 | 2372.063232 | 42.676437 | 2375.442871 | -0.002892 |
| 2018-01-01 00:04:00 | 2.0 | 95.0 | 2376.060059 | 2406.399902 | 2344.000000 | 2370.566650 | 37.820919 | 2371.096191 | -0.003718 |
| 2018-01-01 00:05:00 | 2.0 | 33.0 | 2372.656738 | 2404.600098 | 2343.399902 | 2370.173340 | 8.519679 | 2370.345703 | -0.002171 |

02. 분석 과정 및 방법

- 사용 데이터 및 전처리

- 4) Scaling (MinMaxScaler 사용)

```
1 from sklearn.preprocessing import MinMaxScaler
2 mms = MinMaxScaler()
3 scaled_df = mms.fit_transform(resample_df)
4 scaled_df = pd.DataFrame(scaled_df)
5 scaled_df.head()
```

| | Count | Open | High | Low | Close | Volume | VWAP | Asset_ID | Target |
|---------------------|----------|----------|----------|----------|----------|--------------|----------|----------|-----------|
| timestamp | | | | | | | | | |
| 2019-04-12 14:34:00 | 0.000455 | 0.004357 | 0.004369 | 0.004340 | 0.004359 | 1.900141e-07 | 0.016496 | 2 | -0.003690 |
| 2019-04-12 14:35:00 | 0.000339 | 0.004363 | 0.004373 | 0.004346 | 0.004365 | 1.133496e-07 | 0.016503 | 2 | -0.001639 |
| 2019-04-12 14:36:00 | 0.000448 | 0.004365 | 0.004375 | 0.004347 | 0.004365 | 1.288732e-07 | 0.016504 | 2 | -0.002862 |
| 2019-04-12 14:37:00 | 0.000188 | 0.004365 | 0.004378 | 0.004348 | 0.004365 | 4.711323e-08 | 0.016505 | 2 | -0.000222 |
| 2019-04-12 14:38:00 | 0.000158 | 0.004361 | 0.004376 | 0.004349 | 0.004361 | 3.066040e-08 | 0.016500 | 2 | -0.005420 |

02. 분석 과정 및 방법

- 사용 데이터 및 전처리

- 5) Create X, Y

```
1 X_candi = final_df3[['Count', 'Open', 'High', 'Low', 'Close', 'Volume', 'VWAP']]
2 X_candi.head()
```

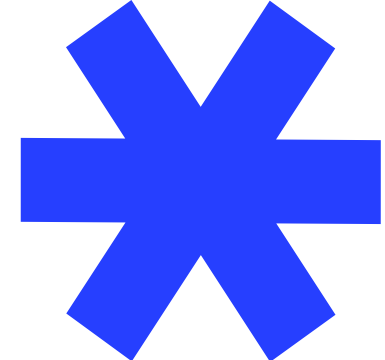
| | Count | Open | High | Low | Close | Volume | VWAP |
|---|----------|--------------|--------------|--------------|--------------|--------------|----------|
| 0 | 0.000073 | 2.752653e-04 | 2.749215e-04 | 2.758567e-04 | 2.752558e-04 | 1.213030e-06 | 0.012463 |
| 1 | 0.002024 | 7.809439e-02 | 7.862402e-02 | 7.776030e-02 | 7.811114e-02 | 1.024982e-07 | 0.089355 |
| 2 | 0.000455 | 4.356732e-03 | 4.368857e-03 | 4.339532e-03 | 4.358667e-03 | 1.900141e-07 | 0.016496 |
| 3 | 0.000067 | 1.270240e-06 | 1.268490e-06 | 1.287542e-06 | 1.270374e-06 | 1.051638e-04 | 0.012193 |
| 4 | 0.000055 | 2.476008e-08 | 2.434514e-08 | 3.978628e-08 | 2.474365e-08 | 2.528569e-04 | 0.012191 |

(17990658, 7)

```
1 X = X[0:17990449:14]
2 X.shape
```

(1285033, 210, 7)

sliding_window_view / reshape를 활용한 X 생성



02. 분석 과정 및 방법

- 사용 데이터 및 전처리(4분으로 15분 예측)

sliding_window_view

| | Count | Open | High | Low | Close | Volume | VWAP | |
|---|----------|--------------|--------------|--------------|--------------|--------------|----------|-------|
| 0 | 0.000073 | 2.752653e-04 | 2.749215e-04 | 2.758567e-04 | 2.752558e-04 | 1.213030e-06 | 0.012463 | (4,7) |
| 1 | 0.002024 | 7.809439e-02 | 7.862402e-02 | 7.776030e-02 | 7.811114e-02 | 1.024982e-07 | 0.089355 | (4,7) |
| 2 | 0.000455 | 4.356732e-03 | 4.368857e-03 | 4.339532e-03 | 4.358667e-03 | 1.900141e-07 | 0.016496 | (4,7) |
| 3 | 0.000067 | 1.270240e-06 | 1.268490e-06 | 1.287542e-06 | 1.270374e-06 | 1.051638e-04 | 0.012193 | |
| 4 | 0.000055 | 2.476008e-08 | 2.434514e-08 | 3.978628e-08 | 2.474365e-08 | 2.528569e-04 | 0.012191 | |
| 5 | 0.003303 | 8.293215e-05 | 8.354245e-05 | 8.268373e-05 | 8.290876e-05 | 1.439672e-05 | 0.012273 | |
| 6 | 0.000661 | 2.544759e-03 | 2.558996e-03 | 2.532681e-03 | 2.544477e-03 | 1.761146e-07 | 0.014705 | |
| 7 | 0.000236 | 9.829469e-05 | 9.905708e-05 | 9.773386e-05 | 9.828379e-05 | 1.592073e-06 | 0.012288 | |
| 8 | 0.000036 | 4.827946e-06 | 4.947689e-06 | 4.723945e-06 | 4.830052e-06 | 1.982870e-06 | 0.012196 | |
| 9 | 0.000527 | 1.223813e-03 | 1.231507e-03 | 1.217862e-03 | 1.223810e-03 | 6.702172e-07 | 0.013400 | |



02. 분석 과정 및 방법



- 사용 데이터 및 전처리

- 6) train/valid/test 설정

```
test_idx = 43200  
valid_idx = 86400
```

```
X_train, X_valid, X_test = resize_X[:-valid_idx], resize_X[valid_idx:-test_idx], resize_X[-test_idx:]  
y_train, y_valid, y_test = y[:-valid_idx], y[valid_idx:-test_idx], y[-test_idx:]
```

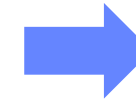
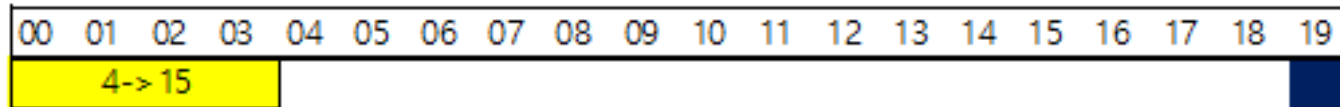


02. 분석 과정 및 방법

- 모델 설계 및 학습

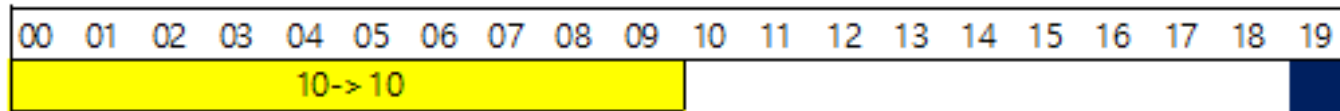
- 3가지 예측가설 설정

- A : 4분 간의 데이터를 통해 15분 후의 가격 예측



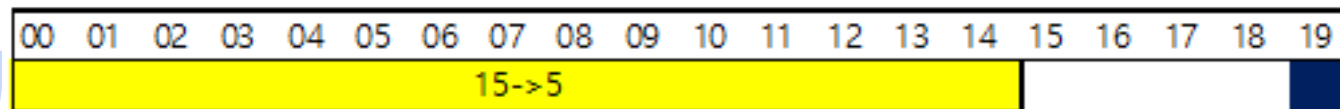
7개 모델 확인
(1,2,3,4,5,6,7 모델)

- B : 10분 간의 데이터를 통해 10분 후의 가격 예측



7개 모델 확인
(1,2,3,4,5,6,7 모델)

- C : 15분 간의 데이터를 통해 5분 후의 가격 예측



7개 모델 확인
(1,2,3,4,5,6,7 모델)

총 21개 모델 확인



02. 분석 과정 및 방법

- 모델 설계 및 학습 - 기본 모델

```
1  ## GRU
2  model = keras.models.Sequential()
3  model.add(keras.layers.InputLayer(input_shape=[210, 7]))
4  model.add(keras.layers.GRU(32, activation='tanh', return_sequences=True))
5  model.add(keras.layers.BatchNormalization())
6  keras.layers.Dropout(0.2),
7  model.add(keras.layers.GRU(32, activation='tanh', return_sequences=True))
8  model.add(keras.layers.BatchNormalization())
9  keras.layers.Dropout(0.2),
10 model.add(keras.layers.Dense(64, activation='relu'))
11 model.add(keras.layers.Dense(32, activation='relu'))
12 model.add(keras.layers.Dense(1))
13 model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```



02. 분석 과정 및 방법

- 모델 설계 및 학습 - 모델 조합

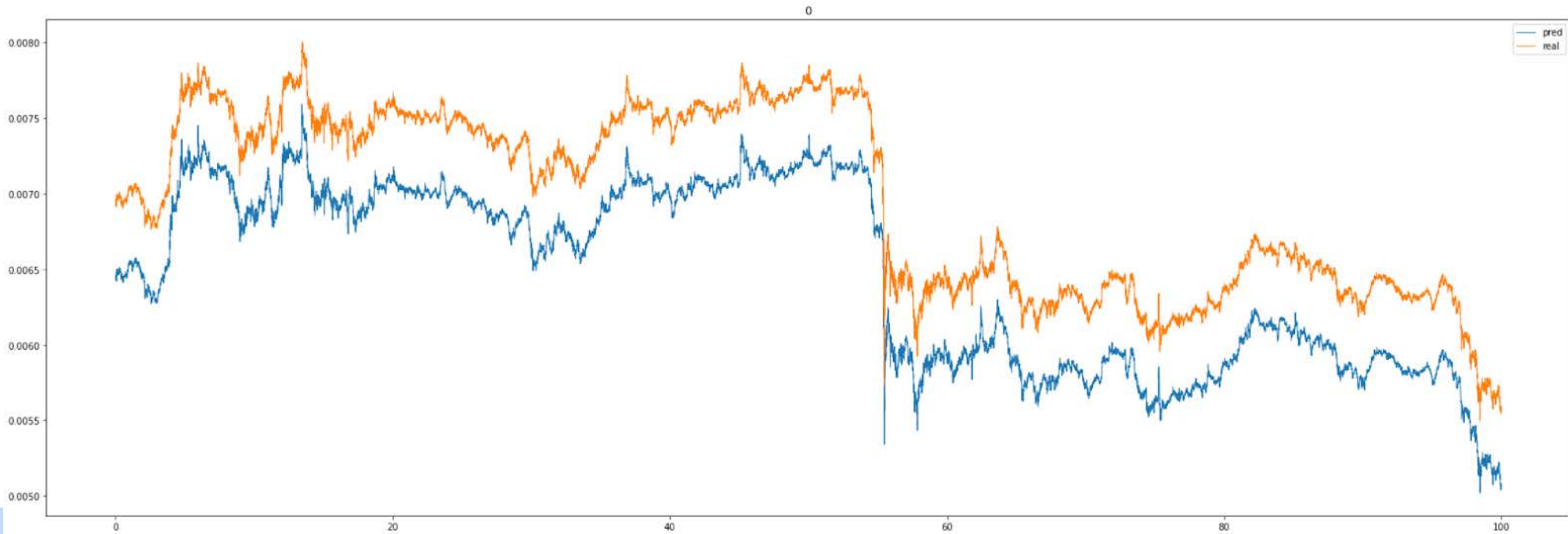
| | 모델01 (RNN +RNN) | 모델02 (LSTM +LSTM) | 모델03 (GRU +GRU) | 모델04 (LSTM +GRU) | 모델05 (RNN +LSTM) | 모델06 (GRU +LSTM) | 모델07 (LSTM +GRU_2) |
|----------|-----------------------|-------------------------|-----------------------|------------------------|------------------------|------------------------|--------------------------|
| Layer1 | 32 | 32 | 32 | 32 | 32 | 32 | 128 |
| Dropout1 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.3 |
| Layer2 | 32 | 32 | 32 | 32 | 32 | 32 | 64 |
| Dropout2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.3 |
| Dense | d(64 32) | d(64 32) | d(64 32) | d(64 32) | d(64 32) | d(64 32) | d(64 32) |



02. 분석 과정 및 방법

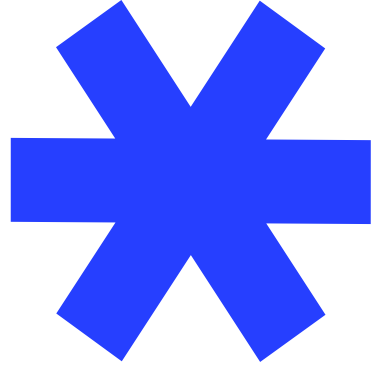
- 최종가설

- 처음 예측한값(predict)과 실제값(y_test)을 비교하려고 하였다



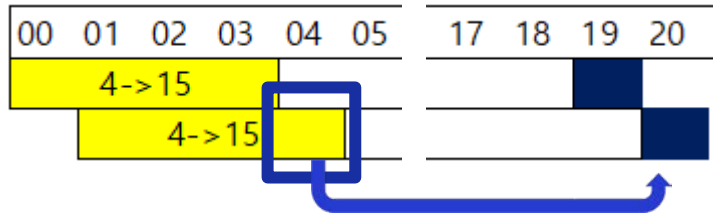
- 하지만 위 그래프와 같이 방향성과 볼륨을 모두 예측을 하더라도 예측값 그래프가 아래에 있으면 매매를 진행하지 않는 문제 발생할 수 있다고 판단

02. 분석 과정 및 방법

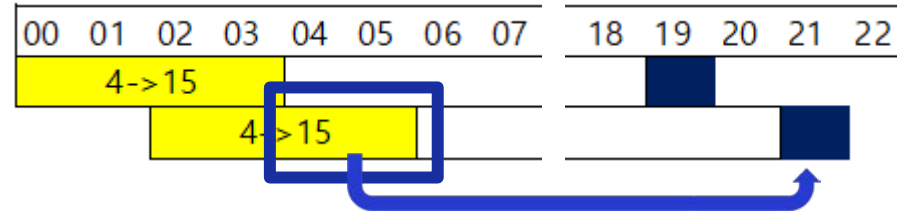


- 최종가설(A가설)

- 4분으로 - 15분 후 예측 의 경우 매분 1/4 의 새로운 정보로 가격의 변화가 생김



4분의 데이터(1/4)로 인하여 가격이 변동



4분&5분의 데이터(1/2)로 인하여 가격이 변동

- 가장 최근 정보가 반영됨으로써 변하는 예측값 활용
- 4분에서 19분까지 15분 동안 30번의 거래(구매 15번 ,판매 15번)가 발생
- 수익률 지표인 기준 자산은 편의상 각 자산 가격의 최고가의 *15로 설정함

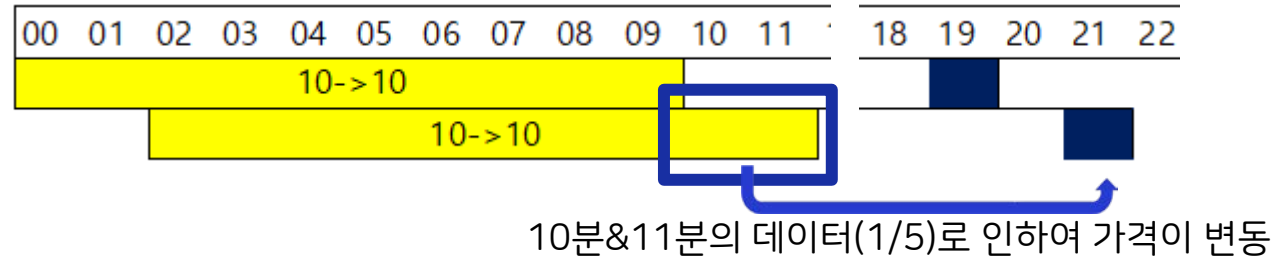
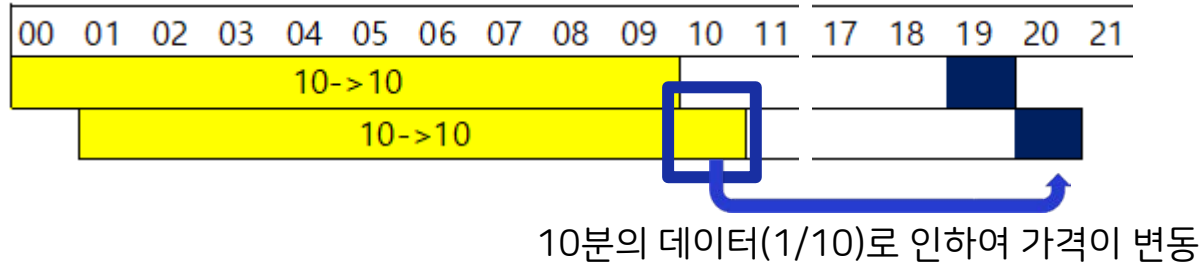
- 어느 정도의 변화가 가장 최적의 수익을 달성할 수 있을지 실험

02. 분석 과정 및 방법



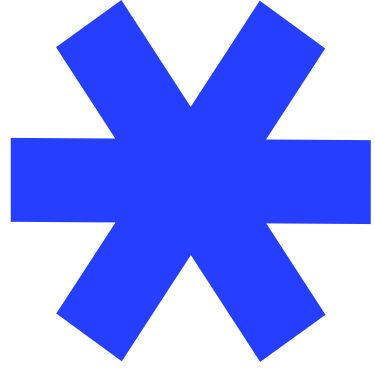
- 최종가설(B가설)

- 10분으로 - 10분 후 예측 의 경우 매분 1/10 의 새로운 정보로 가격의 변화가 생김



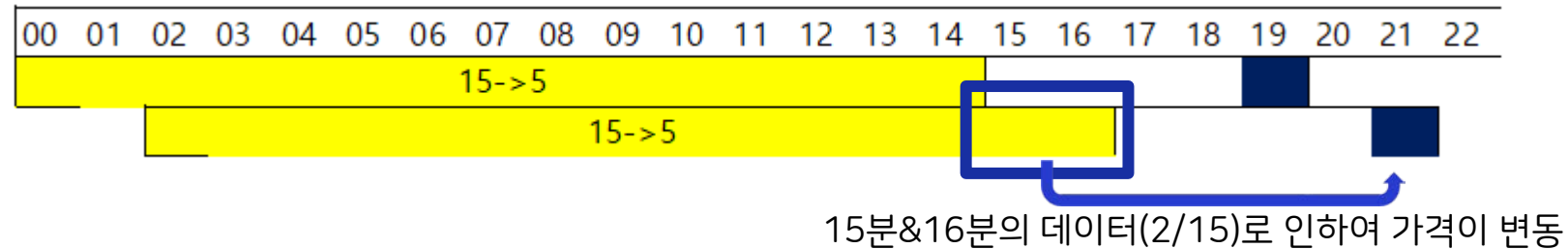
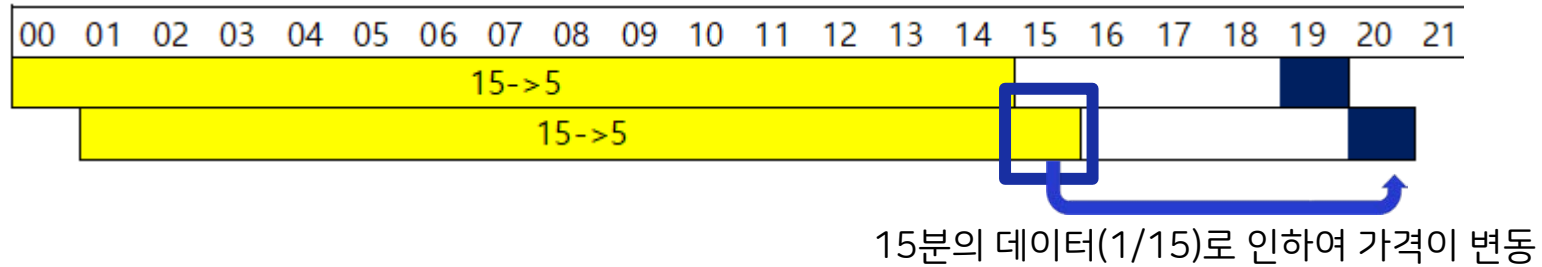
- 가장 최근 정보가 반영됨으로써 변하는 예측값 활용
- 9분에서 19분까지 10분 동안 20번의 거래(구매 10번 ,판매 10번)가 발생
- 수익률 지표인 기준 자산은 편의상 각 자산 가격의 최고가의 *10로 설정함
- 어느 정도의 변화가 가장 최적의 수익을 달성할 수 있을지 실험

02. 분석 과정 및 방법



- 최종가설(C가설)

- 15분으로 - 5분 후 예측 의 경우 매분 1/15 의 새로운 정보로 가격의 변화가 생김



- 가장 최근 정보가 반영됨으로써 변하는 예측값 활용
- 14분에서 19분까지 5분 동안 10번의 거래(구매 5번 ,판매 5번)가 발생
- 수익률 지표인 기준 자산은 편의상 각 자산 가격의 최고가의 *5로 설정함
- 어느 정도의 변화가 가장 최적의 수익을 달성할 수 있을지 실험

02. 분석 과정 및 방법

- 최종가설

| | | RNN ONLY | LSTM ONLY | GRU ONLY | LSTM+GRU | RNN+LSTM |
|----|----------|-------------|--------------|--------------|--------------|-----------|
| AS | ASSET 0 | -98.0395703 | -114.9389531 | -83.80241901 | -115.5176991 | -95.49308 |
| AS | ASSET 1 | -0.02646256 | 0.011341098 | 0.003703216 | 0.045981596 | 0.001157 |
| AS | ASSET 2 | -0.000952 | 0 | -0.002215816 | -9.79E-05 | |
| AS | ASSET 3 | 0.040981771 | 0 | -0.108759315 | -0.437795649 | |
| AS | ASSET 4 | -8600.67959 | -5824.08 | | | 6907.632 |
| AS | ASSET 5 | 8908.840981 | 11692.8 | | | 7628.19 |
| AS | ASSET 6 | -0.49211504 | -1.20396 | ■ ■ ■ | | 0.357069 |
| AS | ASSET 7 | -0.10868614 | | | | |
| AS | ASSET 8 | -2.40213878 | 0 | -2.141861894 | 0 | |
| AS | ASSET 9 | -0.07488465 | -0.134451977 | 0.010614771 | -0.041791683 | |
| AS | ASSET 10 | 0.996419075 | 2.345999341 | 2.711773431 | 2.118967146 | 3.468547 |
| AS | ASSET 11 | -0.4491018 | -0.823353293 | -0.67606722 | -2.124782693 | -1.376279 |
| AS | ASSET 12 | 5.177575892 | 0 | -8.981008419 | 23.48312857 | 4.760425 |
| AS | ASSET 13 | 6755.171459 | 20932.69181 | 8617.708446 | 15331.18132 | 21641.56 |

| | RNN ONLY | LSTM ONLY | GRU ONLY | LSTM+GRU | RNN+LSTM |
|---------|-------------|-------------|-------------|-------------|-------------|
| ASSET 0 | 251.2336332 | 232.4533259 | 164.6821708 | 246.3721669 | 206.6701922 |
| ASSET 1 | 211.4881773 | 209.2563727 | 181.1119337 | 206.4320536 | 201.5141832 |
| ASSET 2 | 74.51681764 | 44.20466154 | 33.04310338 | 4.93E+01 | 39.05526344 |
| ASSET 3 | 0 | -10.9200657 | -61.8383399 | -72.4809905 | 35.20649365 |
| ASSET 4 | -0.09134846 | -24.641764 | -13.861819 | -20.1710629 | -18.0547547 |
| ASSET 5 | -6.34151355 | 9.549653303 | 9.896295523 | 32.20374189 | -9.90309243 |
| ASSET 6 | 327.6982726 | 324.8783389 | 280.7848297 | 327.0390673 | 311.2547629 |
| ASSET 7 | 60.98016927 | 52.22731228 | 62.19745402 | 5.37634097 | 11.17293501 |

4분으로 15분 예측 : 7(모델) X 4분 = 28개 비교
 10분으로 10분 예측 : 7(모델) X 10분 = 70개 비교
 15분으로 5분 예측 : 7(모델) X 15분 = 105개 비교


총 203개 비교



02. 분석 과정 및 방법

- 최종가설(최적 모델 및 시점, 투자 자산 설정)



| 열1 | 4분으로 15분 예측 최종모델 | 10분으로 10분 예측 최종 모델 | 15분으로 5분 예측 최종 모델 |
|----------|--|--------------------|-------------------|
| | 1시점 전 | 9시점 | 8시점 |
| | LSTM ONLY | GRU ONLY | LSTM+GRU |
| ASSET 0 | -961.3667 | -54.6336 | 646.7776 |
| ASSET 1 | -186.3712 | -0.0672 | 513.3151 |
| ASSET 2 | 42.5682 | -0.0016 | 186.0577 |
| ASSET 3 | 24.5323 | -0.2152 | -452.4135 |
| ASSET 4 |  train.csv 6.0621 | 772.0295 | -6.0315 |
| ASSET 5 | 0.0000 | -11470.1328 | 174.3678 |
| ASSET 6 | 852.1550 | -0.6226 | 779.4004 |
| ASSET 7 | -863.7030 | 0.0000 | 147.4509 |
| ASSET 8 | 2194.9538 | -1.3502 | 454.7905 |
| ASSET 9 | 1263.5603 | 0.0073 | 466.7719 |
| ASSET 10 | -2271.2694 | 1.3748 | 573.2058 |
| ASSET 11 | -1194.8501 | 0.7340 | 608.4605 |
| ASSET 12 | -1057.7163 | -27.5074 | 100.9505 |
| ASSET 13 | 205.8827 | 29675.4960 | 102.2105 |
| 평균 | -138.9687 | 1349.6508 | 306.8081 |

02. 분석 과정 및 방법



- 최종가설
 - 4분으로 15분 예측 : LSTM ONLY 모델의 ASSET8이 가장 좋은 예측을 가짐
 - 10분으로 10분 예측 : GRU ONLY 모델의 ASSET13이 가장 좋은 예측을 가짐
 - 15분으로 5분 예측 : LSTM+GRU 모델의 전반적으로 좋은 예측을 가짐

-> Valid Set을 통해 실험한 결과, 높은 수익률이 예상되는 모델들로 해당 ASSET에 투자하기로 결정

Contents



Phase 03

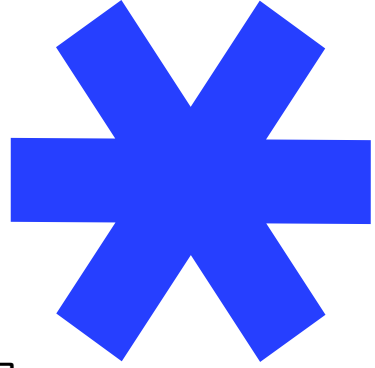
결과

03. 결과

| 열1 ▼ | 1시점 전 ▼ | 9시점 전 ▼ | 8시점 전 ▼ |
|----------|------------------|--------------------|-------------------|
| | 4분으로 15분 예측 최종모델 | 10분으로 10분 예측 최종 모델 | 15분으로 5분 예측 최종 모델 |
| ASSET 0 | -641971.0713 | -51.7399 | 632.6851 |
| ASSET 1 | 47392.8925 | -0.0963 | 508.7725 |
| ASSET 2 | 272262.0370 | 0.0000 | 135.4400 |
| ASSET 3 | 3297796.5190 | 0.0000 | 264.2505 |
| ASSET 4 | -74754800.1100 | -3562.1712 | 61.1481 |
| ASSET 5 | 2804852.9230 | -12193.9761 | -65.8892 |
| ASSET 6 | -2689.8911 | -1.0918 | 774.0169 |
| ASSET 7 | -4326221.4330 | -0.0860 | 233.7477 |
| ASSET 8 | -11068357.5700 | 0.0000 | 113.8856 |
| ASSET 9 | -1631033.2100 | -0.4300 | 470.1122 |
| ASSET 10 | 129932.9426 | 1.5135 | 571.2382 |
| ASSET 11 | -438038.8444 | -1.3449 | 466.8534 |
| ASSET 12 | -46204430.3900 | 0.0000 | 253.8118 |
| ASSET 13 | -379715103.0000 | 18945.0591 | 370.2383 |
| | | | |
| 평균 | -36587886.3004 | 223.9740 | 342.1651 |



03. 결과



- 최종 결과

- 4분으로 15분 예측의 경우 ASSET 8에만 투자 하였고, 이에 대한 수익률은 -11068357.57로 예측에 **실패**하였다.
- 10분으로 10분 예측의 경우 ASSET 13 에만 투자 하였고, 이에 대한 수익률은 18945.0591로 수익률을 얻어 예측에 **성공**하였다.
- 15분으로 5분 예측의 경우 모든 코인에 투자하였고, 이에 대한 수익률은 341.1651로 수익률은 낮지만 ASSET5를 제외한 예측에 **성공**하였다.

- 한계점

- 사용한 데이터 기간이 암호화폐 상승구간에 속해 있었다.

이러한 외부요인이 모델의 예측 정확도를 향상시켰을 가능성이 존재한다.



QnA





The End

