Copyright © All Right Reserved by AI QUANT INVESTING LAB
이 자료는 저작권법에 따라 보호받는 저작물이므로 무단배포 및 무단복제를 금합니다

GPS

GPS

pt rovide ervice

파이썬을 이용한 금융데이터 분석 및 예측

2023. 07. 07

AI QUANT INVESTING LAB

INDEX

AI QUANT
INVESTING LAB

▪ 개발 배경 및 목표

2022년 개인
투자수익률
-25.4%

제한된 정보로 인한
객관적 매매 불가

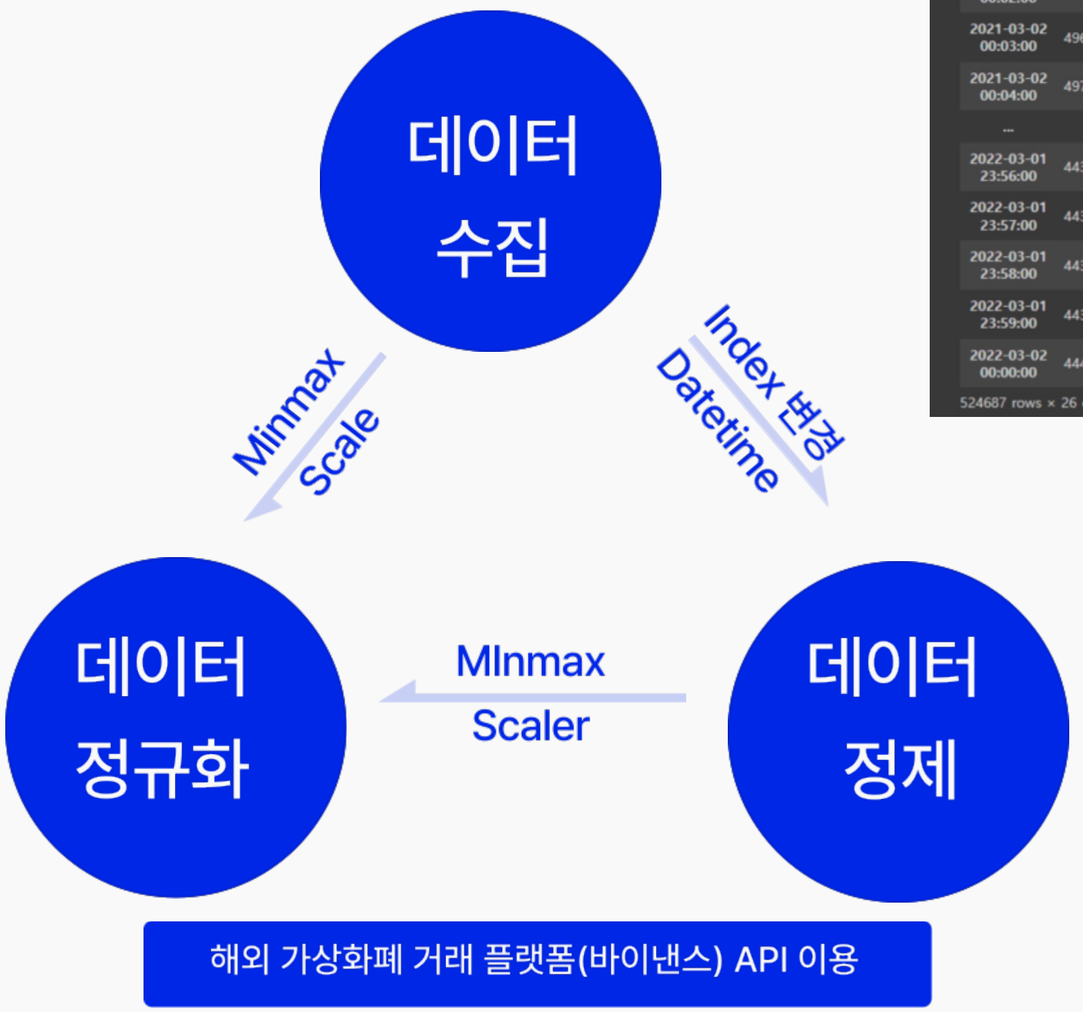
객관적인
판단난항

데이터 기반
객관적 결정

알고리즘 기반
기계적 매매

AI QUANT
INVESTING LAB

▪ 데이터 전처리 과정



selected_data

date	open	high	low	close	volume
2021-03-02 00:00:00	49595.76	49617.70	49515.15	49614.76	76.03
2021-03-02 00:01:00	49614.77	49679.99	49603.07	49636.14	56.30
2021-03-02 00:02:00	49636.15	49680.59	49615.92	49680.59	62.83
2021-03-02 00:03:00	49680.60	49789.99	49680.59	49773.62	145.77
2021-03-02 00:04:00	49773.62	49825.55	49764.84	49807.69	88.04
...	...	...	...	...	...
2022-03-01 23:56:00	44376.27	44399.44	44365.33	44371.87	25.73
2022-03-01 23:57:00	44371.86	44412.22	44371.86	44395.03	15.54
2022-03-01 23:58:00	44395.03	44406.81	44392.40	44392.40	11.45
2022-03-01 23:59:00	44392.40	44423.47	44392.40	44421.20	13.91
2022-03-02 00:00:00	44421.20	44450.98	44378.84	44382.00	49.73

524687 rows × 26 columns

해외 가상화폐 거래 플랫폼(바이낸스) API 이용

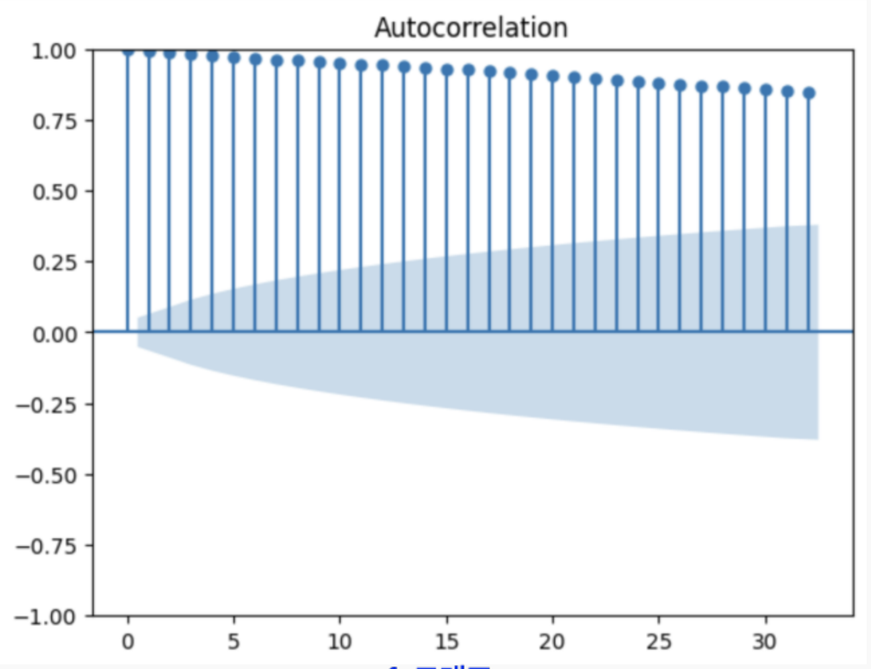
AI QUANT
INVESTING LAB

▪ 모델 선정 과정 - ARIMA

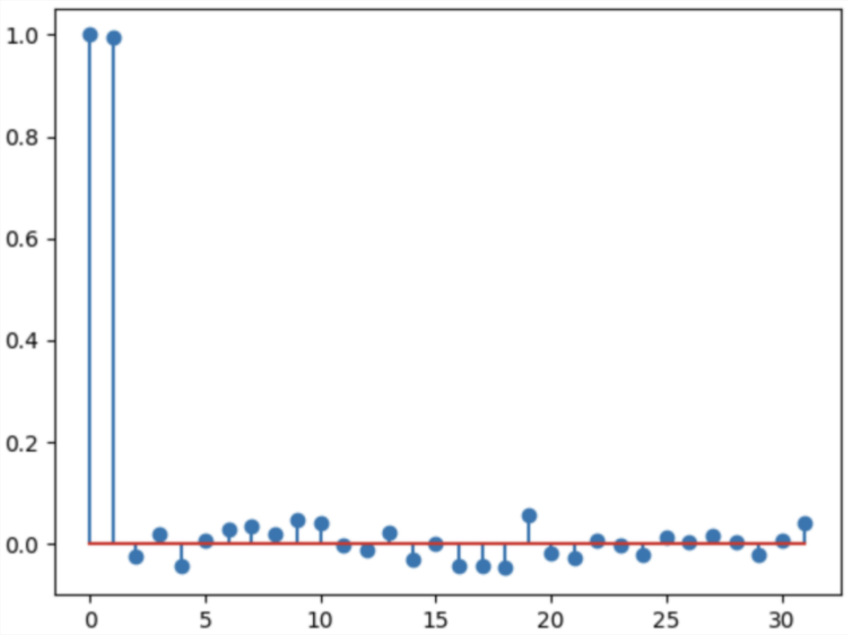
1

ARIMA

- 전통적 시계열 예측 알고리즘
- AFC와 PAFC로 그래프 개형확인 후 주관적 판단 모델
- AFC: 비정상성의 시계열 데이터를 모델링 하는 방법
- PAFC: AFC 모델에서 자기회귀가 추가된 모델



<afc그래프>



<pafc 그래프>

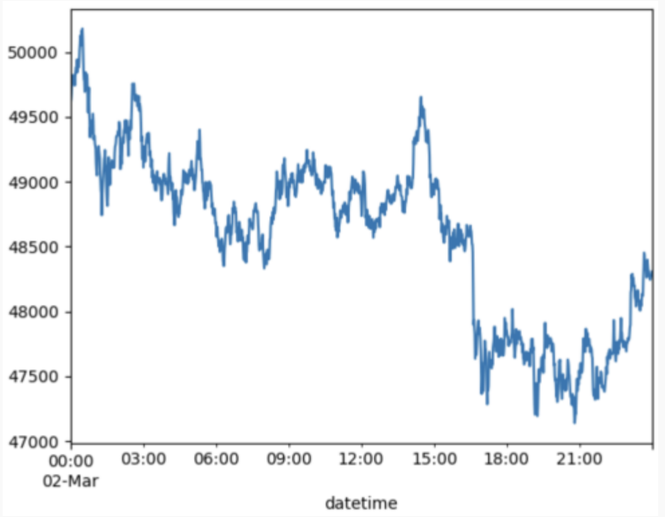
AI QUANT
INVESTING LAB

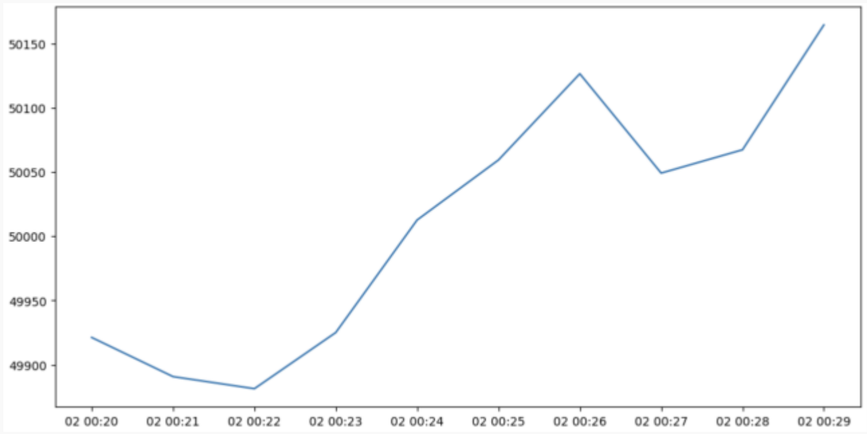
▪ 모델 선정 과정 - ARIMA

1

ARIMA

- 전통적 시계열 예측 알고리즘
- AFC와 PAFC로 그래프 개형확인 후 주관적 판단 모델
- AFC: 비정상성의 시계열 데이터를 모델링 하는 방법
- PAFC: AFC 모델에서 자기회귀가 추가된 모델






```
(0,1,0) AIC: 15971.519898111983
(1,1,0) AIC: 15973.053351602595
(0,1,1) AIC: 15973.028690807987
(1,1,1) AIC: 15972.430320745574
(2,1,0) AIC: 15974.17345807761
(0,1,2) AIC: 15974.029683427518
(2,1,2) AIC: 15975.8307573983
```

AI QUANT
INVESTING LAB

▪ 모델 선정 과정 - RNN

2

RNN

- 순차적 데이터,
  시계열 데이터 처리 적합
- 직전 데이터와 현재 입력 데이터의 관계를 학습하여 순차적 데이터의 특성 이해하고 예측
- RNN은 이전 단계의 정보를 현재로 전달 순차적의존성 학습 가능

```
train_size = 9*24*60 # 7일치, train:test = 7:1
val_size = 2*24*60 # 1일치
test_size = 1*24*60 # 1일치

train_data = data[-train_size:-val_size]
val_data = data[-val_size:-test_size]
test_data = data[-test_size:]


# 데이터 정규화
scaler = MinMaxScaler()
train_data = scaler.fit_transform(train_data)
val_data = scaler.transform(val_data)
test_data = scaler.transform(test_data)


n_steps = 20  # 시퀀스 길이 설정
X_train, y_train = split_sequence(train_data, n_steps)
X_val, y_val = split_sequence(val_data, n_steps)
X_test, y_test = split_sequence(test_data, n_steps)
n_features = len(data.columns)

# LSTM 모델 구축
model = Sequential()
model.add(SimpleRNN(16, activation='tanh', input_shape=(n_steps, n_features), return_sequences=True))
model.add(SimpleRNN(16))
model.add(Dense(n_features))
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse')

# 모델 훈련
history = model.fit(X_train, y_train, epochs=10,
          validation_data=(X_val, y_val),
          batch_size=32)

# 예측 수행
y_pred = model.predict(X_test)


# 예측 결과 역정규화
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)

pred_df = pd.DataFrame(data= y_pred, columns=data.columns)
test_df = pd.DataFrame(data= y_test, columns=data.columns)
```

AI QUANT
INVESTING LAB

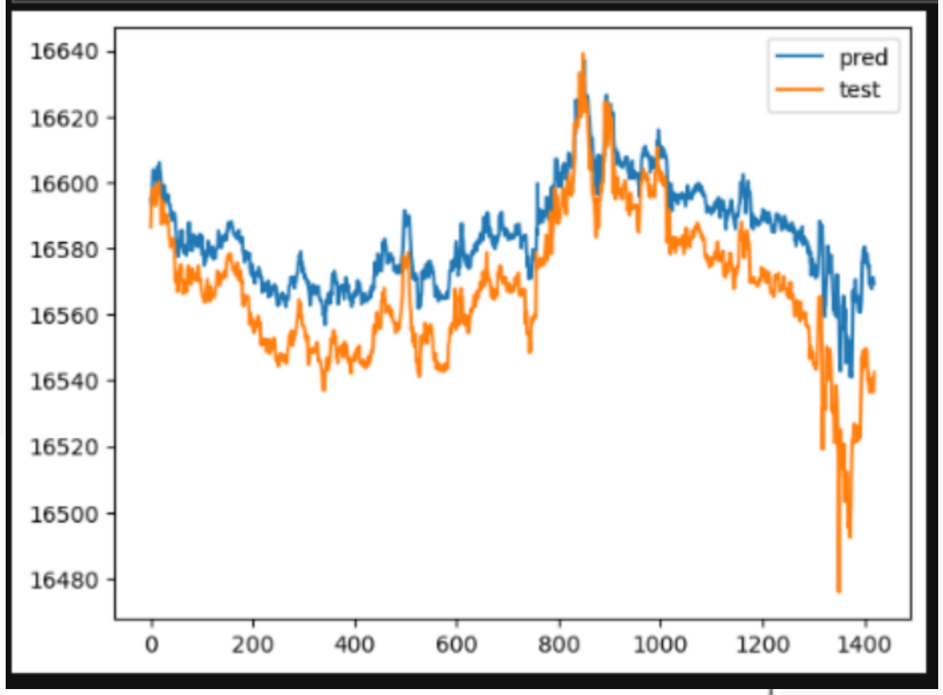
▪ 모델 선정 과정 - RNN

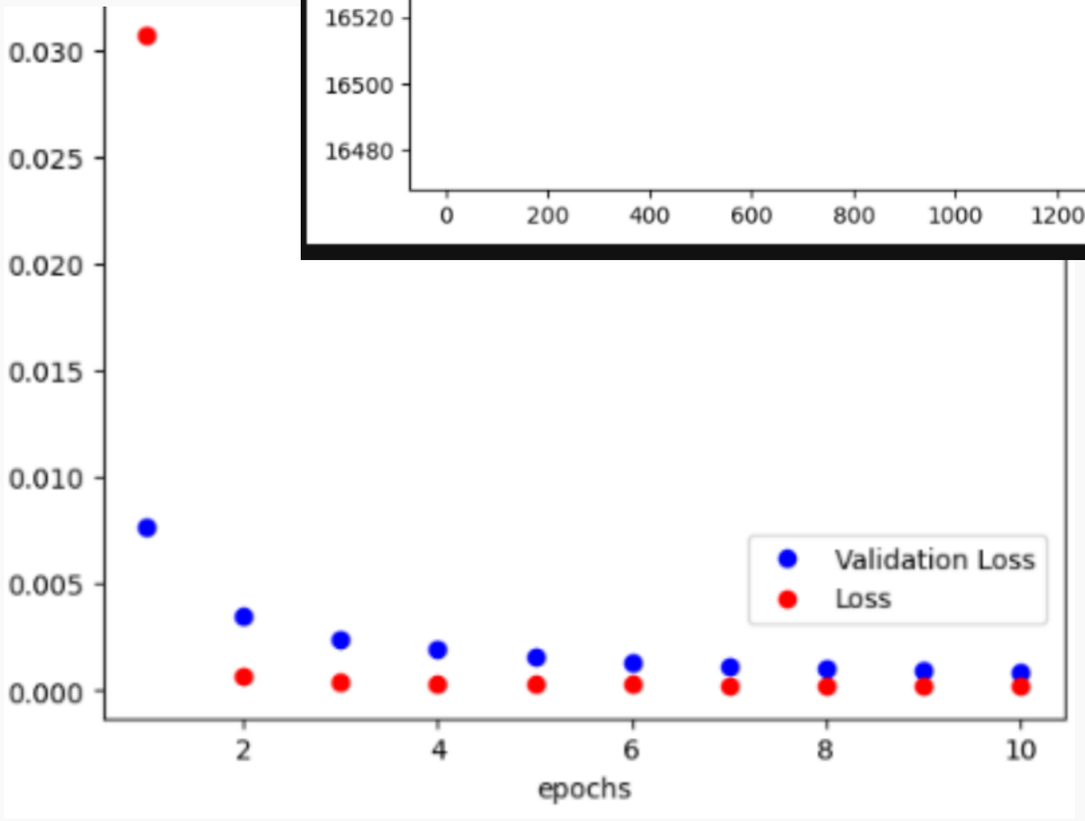
2

RNN

- 순차적 데이터,
  시계열 데이터 처리 적합
- 직전 데이터와 현재 입력 데이터의 관계를 학습하여 순차적 데이터의 특성 이해하고 예측
- RNN은 이전 단계의 정보를 현재로 전달 순차적의존성 학습 가능

```
plt.plot(pred_df['close'], label = 'pred')
plt.plot(test_df['close'], label = 'test')
plt.legend()
plt.show()
```





AI QUANT
INVESTING LAB

▪ 모델 선정 과정 - LSTM

3

LSTM

- 메모리를 가진 순환신경망
- 시간 의존성이 있는 데이터를 효과적으로 학습 할 수 있는 구조
- LSTM은 RNN의 약점인 장기의존성 부분을 해결하기 위해 고안된 알고리즘

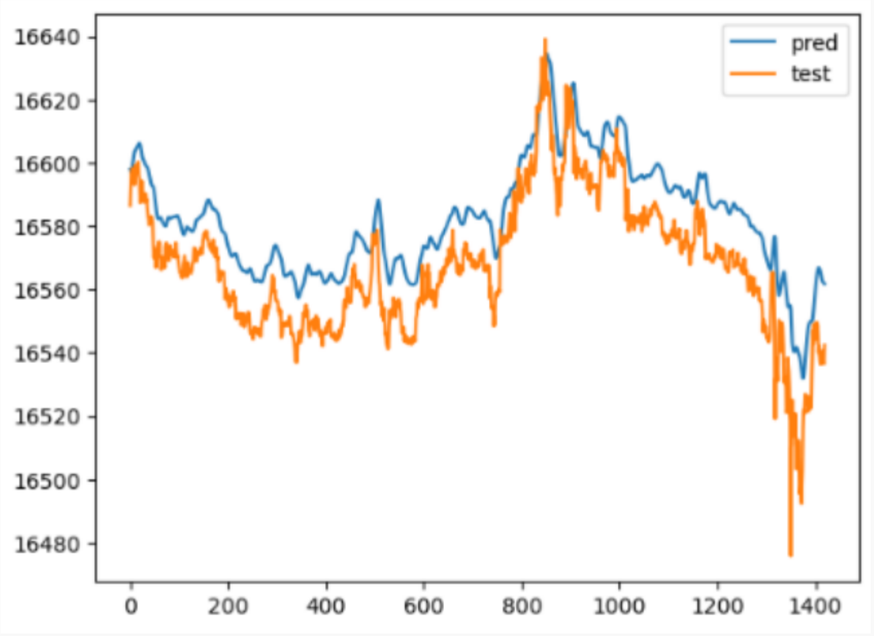
* 장기의존성?
: 직전 데이터보다 더 이전 데이터를 기억하는것

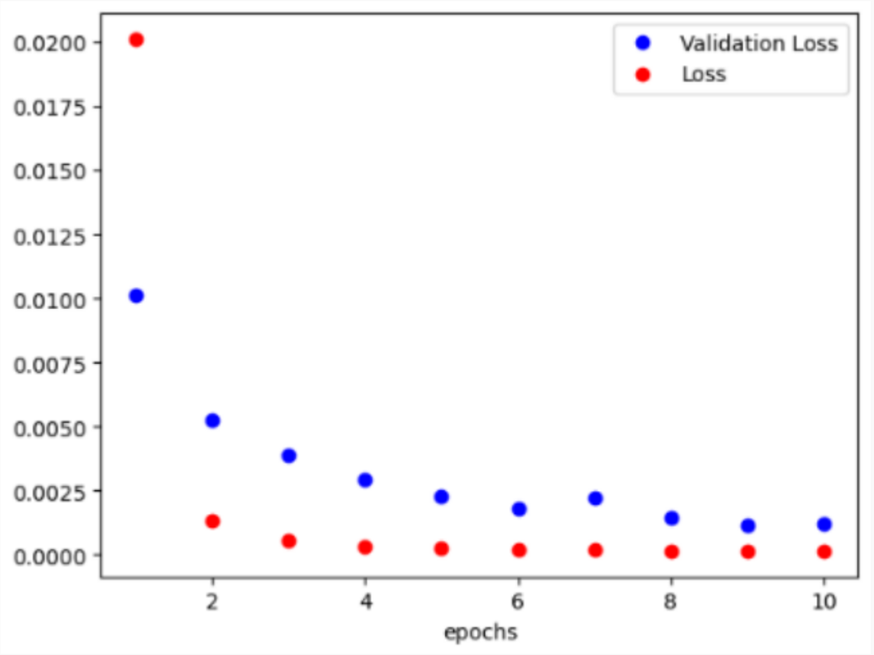
```
train_size = 9*24*60 # 7일치, train:test = 7:1
val_size = 2*24*60 # 1일치
test_size = 1*24*60 # 1일치

train_data = data[-train_size:-val_size]
val_data = data[-val_size:-test_size]
test_data = data[-test_size:]


# 데이터 정규화
scaler = MinMaxScaler()
train_data = scaler.fit_transform(train_data)
val_data = scaler.transform(val_data)
test_data = scaler.transform(test_data)


n_steps = 20  # 시퀀스 길이 설정
X_train, y_train = split_sequence(train_data, n_steps)
X_val, y_val = split_sequence(val_data, n_steps)
X_test, y_test = split_sequence(test_data, n_steps)
n_features = len(data.columns)

# LSTM 모델 구축
model = Sequential()
model.add(LSTM(16, activation='tanh', input_shape=(n_steps, n_features), return_sequences=True))
model.add(LSTM(16))
model.add(Dense(n_features))
optimizer = Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse')

# 모델 훈련
history = model.fit(X_train, y_train, epochs=10,
          validation_data=(X_val, y_val),
          batch_size=32)

# 예측 수행
y_pred = model.predict(X_test)


# 예측 결과 역정규화
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)

pred_df = pd.DataFrame(data= y_pred, columns=data.columns)
test_df = pd.DataFrame(data= y_test, columns=data.columns)
```

AI QUANT
INVESTING LAB

▪ 모델 선정 과정 - LSTM

MAE		
epoch \ model	RNN	LSTM
1	24.650581	6.179807
2	9.346464	4.197919
3	31.455802	6.49105
4	4.068997	5.047941
5	19.550031	5.058647
6	6.699778	6.415832
7	3.992377	5.084698
8	6.461742	11.491305
9	14.127556	6.631837
10	4.585149	15.114103
avg	12.4938477	7.1713139





AI QUANT
INVESTING LAB

▪ 모델 선정 및 결과 - LSTM + Ensemble학습 모델 구현

Enssemble 학습:

여러개의 분류기를 생성하여 그 예측 값들을 결합하여 더 정확한 결과를 도출하는 기법

```python
# 모델 생성 함수
def create_ensemble_models(input_shape, num_models):
    models = []
    for _ in range(num_models):
        model = Sequential()
        model.add(BatchNormalization(input_shape=input_shape))
        model.add(LSTM(50, return_sequences=True))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))
        model.add(LSTM(50))
        model.add(Dropout(0.5))
        model.add(Dense(1))
        model.compile(optimizer=Adam(learning_rate=0.0001), loss='mse')
        models.append(model)  # 모델을 리스트에 추가
    return models  # 모델 리스트 반환
```

```python
# 모델 생성 및 학습
num_models = 3  # 앙상블에 사용할 모델의 개수
models = create_ensemble_models((X_train.shape[1], 1), num_models)
```

```python
# X_next_day와 y_next_day 정의
X_next_day = scaler.transform(next_day_data.drop('close', axis=1))
X_next_day = X_next_day.reshape((X_next_day.shape[0], X_next_day.shape[1], 1))
y_next_day = scaler_close.transform(next_day_data[['close']])

# 예측 수행
y_preds = [model.predict(X_next_day, verbose=0) for model in models]

# 예측값을 원래의 스케일로 변환
y_preds_original_scale = [scaler_close.inverse_transform(y_pred) for y_pred in y_preds]
y_pred_original_scale = np.mean(y_preds_original_scale, axis=0)
```

AI QUANT
INVESTING LAB

▪ 모델 선정 및 결과 - LSTM + Ensemble학습 모델 구현

Enssemble 학습:

여러개의 분류기를 생성하여 그 예측 값들을 결합하여 더 정확한 결과를 도출하는 기법

```
# 이 X를 모델에 넣어 예측을 수행합니다.
y_future_preds = [model.predict(X_future, verbose=0) for model in models]
# 예측된 값을 원래의 스케일로 변환하고 결과 데이터프레임에 추가합니다.
y_future_preds_original_scale = [scaler_close.inverse_transform(y_future_pred) for y_future_pred in y_future_preds]
y_future_pred_original_scale = np.mean(y_future_preds_original_scale, axis=0)
future_results = future_results.append(pd.DataFrame({'Pred': y_future_pred_original_scale.flatten()}, index=future_data.index))
```

```
# R^2가 평균 임계값보다 낮으면 모델 업데이트
if np.mean(r2_values) < threshold_r2:
    X_update = X_next_day
    y_update = y_next_day

    # 각 모델에 대해 학습을 수행
    for i, model in enumerate(models):
        # 모델별 체크포인트 파일 경로 설정
        model_path = model_dir + "model_{}_{}.h5".format(i, end_date.strftime('%Y%m%d'))
        model_checkpoint = ModelCheckpoint(model_path, save_best_only=True, monitor='val_loss', mode='min')

        try:
            model.load_weights(model_checkpoint.filepath)
        except FileNotFoundError:
            print(f"Model {i}: Checkpoint file not found at {model_checkpoint.filepath}. Starting training from scratch.")

        # 모델 업데이트 (재학습)
        model.fit(X_update, y_update, epochs=10, verbose=1, validation_split=0.2, callbacks=[early_stopping, model_checkpoint])
        model.save(model_checkpoint.filepath)

models = [load_model(model_dir + "model_{}_{}.h5".format(i, start_date.strftime('%Y%m%d'))) for i in range(num_models)]
```

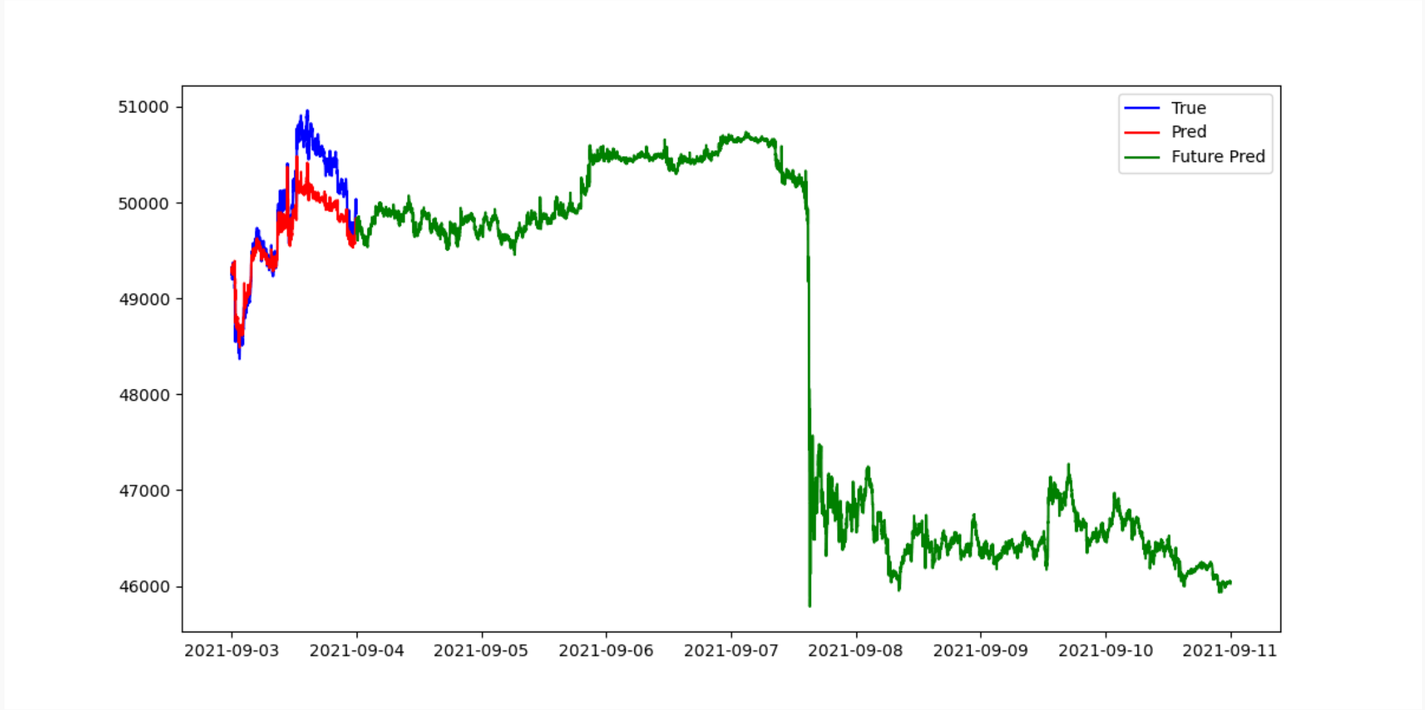
```
# 기존 데이터 불러오기
df_old = pd.read_csv(data_path, index_c
# 가장 최근의 시간 찾기
last_time = df_old.index[-1]

# 바이낸스에서 데이터 불러오기
klines = client.get_historical_klines('

# 데이터프레임으로 변환
df_new = pd.DataFrame(klines, columns=[

# 필요한 열만 선택하고 이름 변경
df_new = df_new[['open_time', 'open', '
df_new.columns = ['date', 'open', 'high

# 'date' 열을 datetime으로 변환
df_new['date'] = pd.to_datetime(df_new[

# 'date' 열을 인덱스로 설정
df_new.set_index('date', inplace=True)

# 데이터 병합
df = pd.concat([df_old, df_new])
```

AI QUANT
INVESTING LAB

▪ 모델 선정 및 결과 - LSTM + Ensemble학습 모델 구현

Enssemble 학습:

여러개의 분류기를 생성하여 그 예측 값들을 결합하여 더 정확한 결과를 도출하는 기법



웹 사이트 : https://aiquant.netlify.app/

AI QUANT
INVESTING LAB

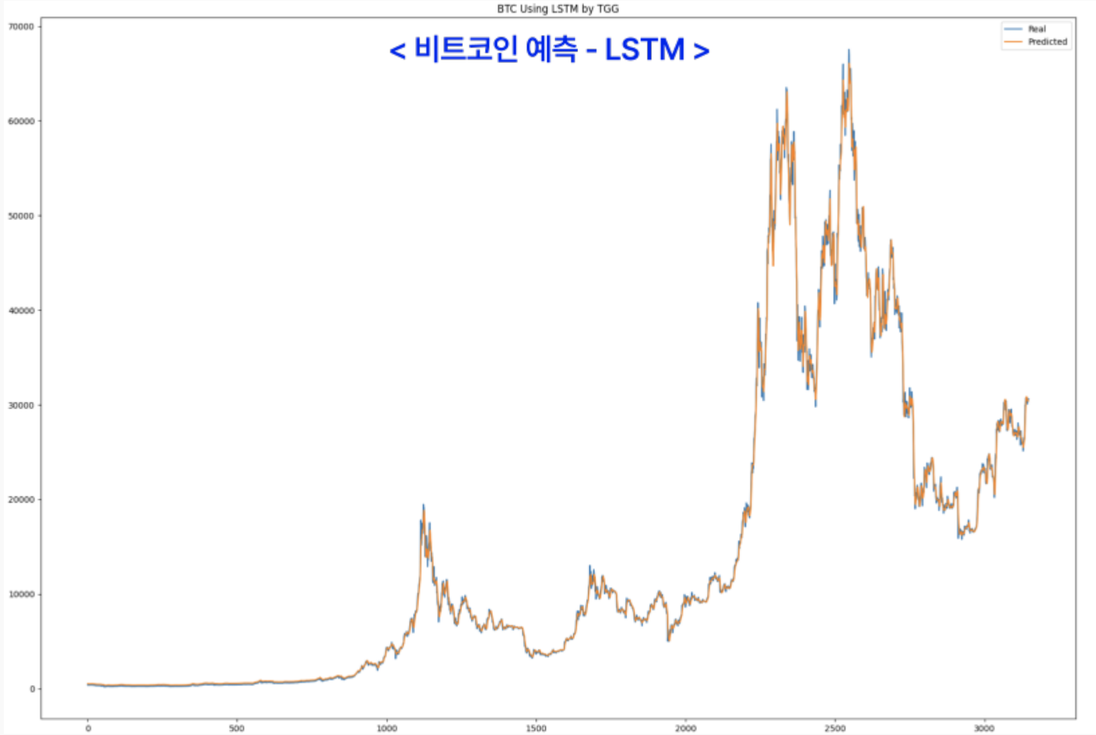
1. 개발 배경 및 목표

2. 데이터 전처리 과정

3. 모델 선정 과정

4. 모델 선정 및 결과

5. 결론 및 고찰



Copyright © All Right Reserved by AI QUANT INVESTING LAB

Welcome, Money

Q & A
NOTICE

AI QUANT LAB department of design