**딥러닝 기반 주식 방향성 예측 AI 프로젝트 보고서**

**초기 계획 :**

* 주가 방향성 예측&포지션 추천 모델
* 다양한 분봉 멀티타임프레임 사용 (2m ~ 1d)
* 딥러닝 기반 시계열 분석(LSTM + Transformer)
* Streamlit 시각화

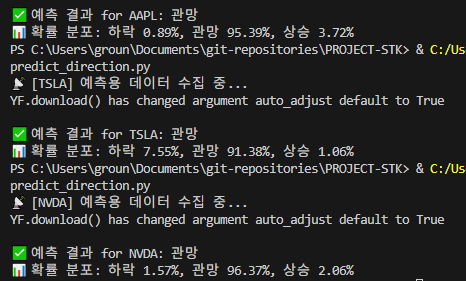
**결과 :**

* **주가 방향성 예측&포지션 추천 (50%)**  
  -> 기능 자체는 구현, 방향성 예측과 추천이 가능하지만, 잘못된 전처리와 라벨링으로 정상적인 방향성 예측과 포지션 추천이 불가능함.
* **다양한 분봉 멀티타임프레임 사용 (변경, 100%)**  
  -> 기존에는 야후 파이낸스 API의 데이터 제공 기준이 2m ~ 60m이 2달 이내 자료를 제공하는 것으로 알고 있어 2분봉~1일봉 사이 자료로 학습시키려 했지만, 실제로 2분봉은 한달 이내 자료만 제공되었기에, 결과적으로 2분봉 자료를 제외함.
* **딥러닝 기반 시계열 분석(LSTM + Transformer) (100%)**  
  -> 기존 계획대로 LSTM 기반 + Transform 구조로 설계함.
* **시각화 (변경)**  
  -> 기존에는 Stramlit 혹은 Pytorch로 시각화 할 계획을 가졌지만, 실제로 진행해보니 발표자료로서는 두 라이브러리보다 Matplotlib이 현실적으로 유용하다 판단되어 변경함.

저는 객관적인 시점으로 보았을 때, 프로젝트를 성공적으로 완성하지 못한 학생입니다.  
때문에 제가 30일이 넘는 기간동안 프로젝트를 위해 얼마나 노력했고 공부했고, 앞으로 어떤 목표를 가지고 있는지 최대한 작성하겠습니다.  
  
그리고 **만약 교수님께서 의향이 있으시다면 해당 프로젝트를 몇 개월, 몇 년이 더 소요되든 정상적으로 작동할 때까지 수정 및 업그레이드 하여 개인적으로 제출하고 싶습니다.**

**프로젝트 진행 기록**

1. **PPO를 통한 주가예측 + git 환경 세팅**  
   텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
     
   사실 계획서 제출 이전, 간단히 PPO를 사용해 프로젝트를 진행해보았다.  
     
     
     
   실제로 PPO 버전의 프로젝트는 TSLA의 적은 데이터만을 사용한 상태였지만, 하락, 관망, 상승의 비율을 비교적 잘 나타냈다.  
   하지만 PPO 알고리즘은 기존의 흐름을 분석해 미래 주가를 예측하기보다는, 존재하는 데이터를 분석하는 알고리즘에 가깝기에 사용할수록 진행하는 프로젝트와는 맞지 않다고 생각했다. 그래서 LSTM으로 변경하게 되었다.
2. **LSTM + Transformer 리팩토링 - LSTM**  
   텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
     
   PPO 알고리즘이 프로젝트에 적합하지 않다는 것을 알게 된 이후, 보다 장기기억 + 미래 예측에 적합한 LSTM과 멀티프레임 분석에 효과적인 Trasformer 알고리즘을 섞어 프로젝트를 진행하기로 마음먹었다.  
   그래서 계획서를 제출함과 동시에 프로젝트를 전부 리팩토링했다.  
   그 과정에서 여러 종목 데이터를 추가했다.  
     
   LSTM을 사용한 구조는 다음과 같았다.   
   텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
     
   주어진 종목의 가격 데이터를 2분봉부터 1일봉까지 여러 타임프레임으로 나누어 불러오고, 각 구간별로 동일한 윈도우 크기만큼 시계열 데이터를 추출한 후, 이를 하나의 시퀀스로 통합하였다.  
   텍스트, 스크린샷, 문서, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   LSTM이 시간 순서에 따라 학습할 수 있도록 구성했고, 입력값으로는 주가뿐 아니라 나스닥 시장 지표도 함께 포함했다.  
   (사용한 종목은 모두 나스닥 종목이었고, 실제 트레이더들도 나스닥 관련 종목을 분석할 때, 나스닥 지표를 참고한다. 때문에 나스닥 지표를 시계열에 포함하여 함께 분석하도록 하였다.)  
     
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   최종적으로는 일정 시간 이후의 가격 변동 방향(상승, 하락, 관망)을 예측하도록 분류 레이블을 생성하여 학습시켰다.  
   (이 과정에서는 아직 라벨링이 단순했기에, 일반적인 상승-2, 하락-0, 관망-1의 전형적인 라벨링을 사용했다.)
3. **전처리**  
     
   이번 프로젝트에서 가장 어려웠던 부분은 전처리와 라벨링이었다.  
   야후 파이낸스 API로 주가 데이터를 다운받는 것은 간단했지만, 저장된 데이터는 실제로 모델의 학습데이터로 사용하기에는 여러 문제가 존재했다.  
     
   **1) Nan값**  
   야후 파이낸스는 기본적으로 정해진 기간 내의 데이터만 제공한다.  
   1분봉은 7일 이내, 2분봉은 1달 이내, 5분봉~60분봉은 80일 이내 자료를 제공한다. 이후 1일, 1주, 1년봉은 몇 년치 데이터를 모두 제공한다.  
   때문에 제공하지 않는 데이터는 다운로드 시 로그로 나타나지도 않고, 빈 값으로 다운받아진다.  
   이 Nan값의 이유를 찾기 위해 2~3일간 해맸다.  
   텍스트, 번호, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   최종적으로 해당 문제는 쓸데없는 자료들을 모두 제거하도록 전처리 코드를 작성하고, 시간 정보를 UTC로 통일하여 맞췄다.  
     
     
     
   **2) 결측치**  
   야후 파이낸스에서 제공하는 종목 타임라인은 종목별로 일관된 결측치(주말, 공휴일), 종목 특화 결측치(종목별 휴장일)이 존재했다.  
   때문에 이 문제를 해결하기 위해 여러 방법을 찾았다.  
   처음에는 간단히 데이터가 존재하는 부분만 학습에 사용하려 했지만, 그러면 이후 종목별로 모두 길이가 달라 학습에 오차가 존재할 수 밖에 없었다.  
   그렇게 약 5일간 정보를 찾아 해맨 결과 최적의 방법은 다음과 같았다.  
   텍스트, 스크린샷이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   \ 나스닥 마켓 기준 거래일을 받아 기본적으로 모두 공통적으로 가지는 결측일을 일전에 방지하여 제거한다.  
     
   텍스트, 폰트, 스크린샷이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   이후 종목별 휴장일은 이전 인덱스를 복제해 늘리는 방식으로 이어붙여 결측을 제거했다.
4. **라벨링**  
     
   앞서 언급했던 대로, 전처리 이후 가장 어려웠던 부분은 라벨링이었다.  
   먼저, 기존 계획에서 필요했던 라벨링은 다음과 같았다.
5. **상승 / 하락 / 횡보 예측**  
   -> 본질적으로 해당 모델은 트레이더가 직접 이상적이고 수익 가능성이 높은 단기~장기 목표설정을 도와주는 서포트 모델이기에, 주가의 상승/하락 가능성만은 도출해낼 수 있어야 했다.  
     
   이 목표를 위해 필요한 라벨링은 다음과 같았다.
6. **주가 방향성 예측**  
   실제 차트 분석에서는, 주가를 예측할 때 여러 지표와 차트도구도 사용하지만, 해당 지표들을 사용하는 이유는 목표 기간 내 주가의 방향성과 고점/저점을 예측하기 위함이다.  
     
   때문에 이 모델은 “주가의 방향성”을 세밀히 알아야 했다.  
   그래서 추가한 라벨링 함수이다.  
   텍스트, 폰트, 스크린샷이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   해당 라벨링 함수 label\_trend는 주가 변화율을 이전과 같은 단순한 정수 0, 1, 2로 표현하는 것이 아니라, 주가의 변화율을 실수로 표현하여 보다 가변적인 주가 변동의 성질을 표현할 수 있도록 하였다.  
   항상 주가 변동의 확률은 0, 1, 2로 삼분법적인 것이 아니라, 상승 확률이 어느정도 되는지, 하락 확률이 어느정도 되는지를 고려한 뒤 확률적인 투자를 진행하는 것이 투자의 기초이기 때문이다. 대부분 상승 100%, 하락 100%는 존재하지 않기에, 해당 라벨링을 통해 어떤 방향을 가지는지 보다 세분화하여 라벨링할 수 있게 하였다.  
     
   또한 해당 라벨링을 통해 이후 포지션 예측에서 확률 대비 비중을 어느정도 가져가야 좋은지에 대한 라벨링도 가능하도록 기획적인 미래시를 열어 두었다.
7. **주가 변화율 예측 (예상 수익률)**  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   해당 라벨 label\_regression은 이전 라벨과 비슷해 보이지만, 목적이 완전히 다른 라벨이다.  
     
   label\_trend가 주가의 방향성을 예측하기 위해서였다면, label\_regression은 주가가 n캔들 뒤 어떻게 변화할 것인지를 예측한다.  
     
   해당 라벨을 추가한 이유는 예상 수익률 예측을 위해서였다.  
   또한 해당 라벨링을 통해 마지막 포지션 예측에도 종합적인 영향을 줄 수 있게 할 기획 미래시를 열어두었다.  
     
   해당 함수를 통해 n시점 후 손익률을 라벨링하였다.
8. 기간 내 최대 수익률 예측  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   해당 라벨 label\_highest는 모델이 목표로 하는 특정 기간 이내 최대 가격을 예측한다.  
     
   사용자가 지정한 구간 종료 시의 수익률 예측만으로는 효과적인 주가예측을 진행할 수 없다.  
     
   때문에 모델이 가변적으로 기간 내 최적 투자기간을 상정하고 기간 내 최대 주가를 파악할 수 있게 하기 위해 해당 라벨링을 추가했다.
9. **미래 캔들 예측**  
   이번 프로젝트 라벨링에서 가장 골머리를 앓게 했던 라벨은 캔들 예측이었다.  
   캔들은 단순히 고점, 저점만 가지고 있는 것이 아니라, 고점, 저점, 캔들 최고점, 캔들 최저점으로 4가지가 ‘대표적’으로 존재했고, 추가적으로 volume 등 여러 캔들 관련 지표가 존재했다. 그 중에서 프로젝트에서 필요로 했던 부분은 대표적으로 존재하는 지표 4가지였기에, 해당 지표 4가지를 사용했다.
10. **캔들 회귀 예측**  
    텍스트, 스크린샷, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
    앞서 말한 캔들의 지표를 활용한 캔들 회귀예측 라벨, label\_candle\_regression이다.  
    해당 라벨은 앞서 말한 최고점~최저점을 라벨링해주고, 이후 예측 과정에서 회귀예측을 통해 예상되는 미래 캔들을 그려낼 수 있게 라벨링 하였다.
11. **포지션 추천**앞서 구성한 라벨링을 종합적으로 합쳐 마지막으로 포지션을 추천하기 위한 기반 라벨, label\_position이다.  
    **텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
    주가 최고점 예측을 통해서 정해진 목표 기간 이내의 포지션 기준으로, 종가의 변화율을 계산해 포지션을 기본적으로 추천해주는 라벨이다.  
    해당 라벨에 이후 다른 라벨들을 합쳐 최종 포지션 추천을 진행하게 할 것이다.
12. **투자 지표 계산**해당 라벨링은 현재 시점에서 완성하지는 못했다.  
    학습 도중 충돌이 너무 많이 일어나 모두 주석처리해두었고, 현재 사용중이지는 않다.  
    텍스트, 스크린샷, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
    실제 차트분석에서 흔히 사용하는 RSI, MACD, 볼린저밴드, 거래량 등을 기본적으로 라벨링하여 사용하려 하였고, 계산하는 과정까지는 구현해두었다.  
    추가적으로 전용 config를 구성하여 이후 작업 및 유지보수 측면에서 유동적으로 지표 라벨을 변경할 수 있게 하였다.  
    텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.
13. **최종 라벨 return  
    텍스트, 스크린샷, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**앞서 구성한 라벨링 함수들을 통해 라벨을 최종적으로 딕셔너리 형태로 리턴했다.  
    임시적으로 목표기간은 100캔들로 설정해두었다.
14. **학습 설정**별로 중요한 파트는 아니지만, 학습 설정 파트다.  
    기존부터 학습속도가 그렇게 느리지 않았기에, gpu를 사용중인줄 알았다.  
    하지만 테스트해 본 결과, cpu를 사용중이었고, 노트북의 cpu가 성능이 좋아 빠른 속도로 학습이 진행됐던 것이었다.  
      
    노트북의 gpu가 gtx3060으로 비교적 좋은 성능을 가졌기에, 해당 gpu를 통해 학습한다면 보다 좋은 결과를 낼 수 있을 것이라고 생각했다.  
      
    하지만 cuda 12.1을 설치하고, 그래픽카드 드라이버를 업데이트한 뒤 잘 설정되었음을 확인했지만, 설정 이후 학습이 정상적으로 작동하지 않고 모든 값을 nan으로 리턴하기 시작했다.  
      
    코드에 문제가 존재하는 것으로 판단하여 코드를 수차례 수정하였지만, 결과적으로 cuda 세팅 문제였고, 프로젝트 발표와 제출을 위해 cpu로 다시 되돌려 학습하였다.  
    **텍스트, 스크린샷, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**
15. **베이스 모델 학습**라벨링과 데이터 전처리를 모두 끝낸 이후, 해당 정보를 기반으로 최종 모델 이전의 베이스 모델이 될 분봉별 종목+나스닥 모델 학습환경을 구성했다.  
      
    이전에는 단순히 여러 분봉 모델이 앙상블 구조를 통해 하나의 결과를 도출하도록 하려 했지만, 개발하다 보니 여러 모델의 결과값을 앙상블하여 하나의 모델로 만들고, 해당 모델을 추가적으로 학습하여 최종 결과값을 가지는 방식으로 모델을 구성하면더 효과적으로 모델을 학습할 수 있겠다고 생각했다.  
      
    때문에 결과적으로 하나의 메타모델을 구성하기 위한, 독립적인 각각의 분봉모델을 구성하기로 마음먹었다.  
      
    분봉모델은 다음과 같은 순서로 학습된다.
16. **모델 및 옵티마이저 초기화**우선 각 분봉(interval)에 대해,   
    **MultiHeadTransformer(다음 섹션, VII에서 따로 설명하겠습니다.)**   
    인스턴스를 생성하고, config의 연산 설정을 반영하도록 .to(DEVICE)로 올린다. 이어서 손실 함수를 바탕으로 파라미터를 자동으로 조정해 줄 Adam 옵티마이저(lr=1e-3)를 정의한다.   
    텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.
17. **에포크 단위 학습  
    텍스트, 시계, 폰트, 디지털 시계이(가) 표시된 사진

    AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**config에서 설정된 EPOCHS만큼 반복하면서, 배치 단위로 입력을 모델에 전달해 예측값을 얻고, get\_loss 메서드를 통해 예측값과 실제값 간 차이를 수치화한 손실(loss)을 계산한 뒤,

optimizer.zero\_grad() → loss.backward() → optimizer.step() 순으로 기울기를 초기화하고, 역전파로 가중치 갱신을 수행한다.  
텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
매 에포크가 끝날 때마다 전체 배치에 대한 평균 손실을 출력하여 학습 진행 상황을 모니터링한다.

1. **가중치 저장  
   **모든 에포크 학습이 완료되면, MODEL\_SAVE\_PATH 폴더에 분봉별로 interval\_model.pt 형태의 파일로 모델 가중치를 저장한다.   
   **텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   이 저장된 파일은 이후 앙상블 또는 메타모델 학습 시 불러와 재활용된다.
2. **LSTM + Transformer 리팩토링 – Transformer 기반 멀티태스크 모델**기존 LSTM 구조는 순차적 처리에 강점을 갖지만, 멀티타임프레임 데이터를 한 번에 보기에는 장기적인 관점으로 여러 종목을 비교하고 분석해야 하는 제 모델의 학습에 제약이 있었다. 그래서”Self-Attention 메커니즘”을 활용할 수 있는 Transformer를 도입했다.  
     
   MultiHeadTransformer 클래스는 다음과 같은 순서로 작동한다.
3. **입력 투영 & Transformer Encoder**원본 피처(input\_dim)를 Transformer 내부 차원(d\_model)으로 바꿔주고,

각 타임스텝의 위치 정보를 더해 시퀀스 내 순서를 인식하게 한다.  
**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
이후 Self-Attention 헤드들이 전체 시퀀스를 한꺼번에 보고

서로 간의 중요도를 동적으로 학습한다.  
텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
**메모리 사용량과 계산 효율, 표현력 균형 문제를 고려하여 d\_model=128, nhead=4, num\_layers=2의 하이퍼파라미터를 사용해 Transformer 구조를 설계했다.**

1. **마지막 시점 멀티헤드 분기  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**시퀀스 마지막 시점 표현만 골라 네 가지 헤드로 나눠

분류(direction/position)와 회귀(return/candle)를 동시에 예측한다.

1. **손실 계산 & 파라미터 갱신  
   텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**get\_loss로 태스크별 오차를 구해 합산한 뒤,

역전파(backward)로 기울기를 계산하고

옵티마이저(step)로 모델 가중치를 조금씩 조정한다.

1. **최종 메타 모델 학습**마지막으로 최종 메타모델을 학습한다.  
   앞서 학습한 분봉별 베이스 모델의 예측된 산출값을 모아, 메타 학습용 데이터로 사용하는 구조로 이루어진다.  
     
   순서는 다음과 같다.
2. **메타모델 데이터셋 생성** **기준분봉 : 30m, 종목 : TSLA**  
   베이스 모델들의 예측 결과를 기반으로 메타모델 학습용 입력(X\_meta)와 정답(y\_meta)를 만든다.  
   텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   각 분봉별 윈도우 수에 맞춰 루프를 돌며, 구성할 메타모델의 분봉(30m), 종목(TSLA)을 기반으로 라벨을 계산한다.  
   텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   계산된 라벨을 토대로 .npz파일을 구성해 저장한다.  
   **텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.** <- npz
3. **메타모델 학습  
   스크린샷, 폰트, 텍스트이(가) 표시된 사진

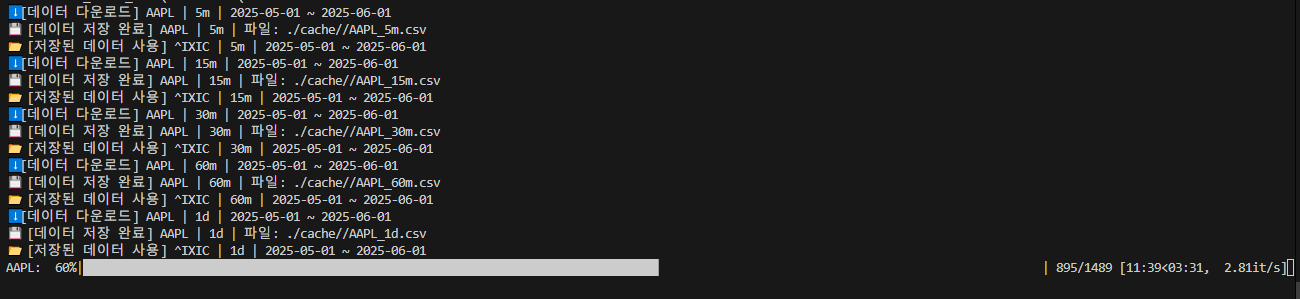
   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**생성된 .npz 데이터셋을 불러온 뒤,  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   라벨을 분리하고,   
   **텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   네트워크 구조를 구성한 뒤, 옵티마이저, 손실 함수를 세팅한다.  
   **텍스트, 스크린샷이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   config에서 정해진 에포크만큼 학습/검증 루프를 돌며 포지션 분류의 정확도를 출력한다.  
   **텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   훈련이 끝난 뒤, 메타모델을 저장한다.
4. **메타 모델 평가 (evaluate)**메타모델을 과거의 데이터를 기반으로 예측하도록 하고, 실제 과거 데이터와 대조하며 평가하게 하였다.  
   **텍스트, 책, 스크린샷, 메뉴이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**에포크 진행동안 학습 성공률, 검증 성공률을 확인할 수 있다. **(해당 이미지는 정상적인 데이터를 사용하지 못해 32에포크부터 급격히 정답률이 올라갔음.)  
   **메타모델이 저장된 이후 실제 데이터와의 대조를 위해 데이터를 다운로드 or 캐시하고, 평가를 시작합니다.
5. **메타 모델 예측 (predict)**

학습된 모델이 적절하다고 생각된다면, 모델 예측 스크립트를 통해 실제로 모델을 사용해볼 수 있다.

1. **모델 예측 (predict\_meta)**모델 예측은 먼저 베이스 모델을 로드하며 시작한다.  
   **텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   베이스 모델 로드 이후,   
   **텍스트, 스크린샷, 폰트이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   메타 모델도 로드하여 주고,  
   텍스트, 폰트, 시계, 스크린샷이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.  
   config에서 주어진 기간 내의 범위에서, 추론할 심볼의 여러 분봉 데이터를 불러온 뒤, 각 분봉별 윈도우 길이만큼 데이터를 자른다.  
   **텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

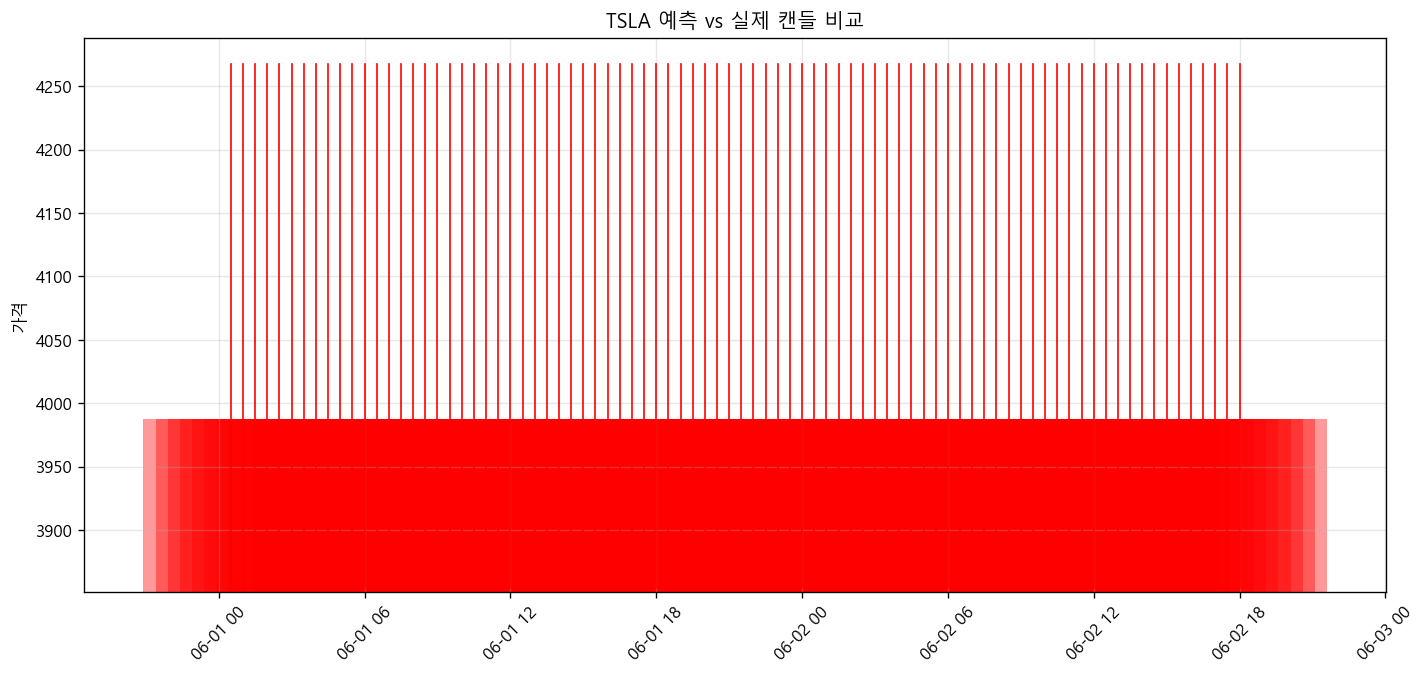
   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   이후 분류된 데이터를 통해 추론을 진행하여 실제 해당 기간 내의 라벨 정보를 예측한다. **텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**  
   산출된 결과를 가공하여 리턴한다.
2. **모델 예측 반복 (predict\_meta\_seq)**predict\_meta\_seq은, predict\_meta가 캔들 등 연속되어야 하는 예측을 한번만 진행하는 문제가 발생해 급조하게 만들었던 반복 예측 스크립트였다.   
   반복되어 여러 추론 결과를 가져와야 하는 캔들, 수익률 등 n의 기간 뒤의 정보를 리턴하는 예측을 n번 반복시켜 충분한 예측 결과를 제공한다.
3. **심볼 탐색**종목 심볼을 보다 쉽게 다운로드하기 위해 구현한 스크립트이다.
4. **나스닥 종목 다운로드 (nasdaqlisted.txt)  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

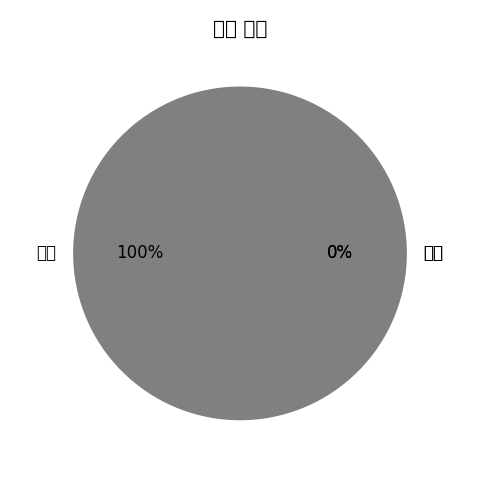
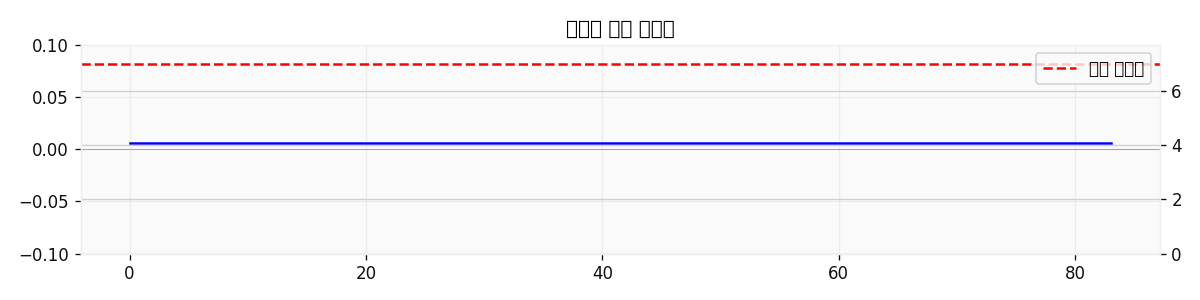
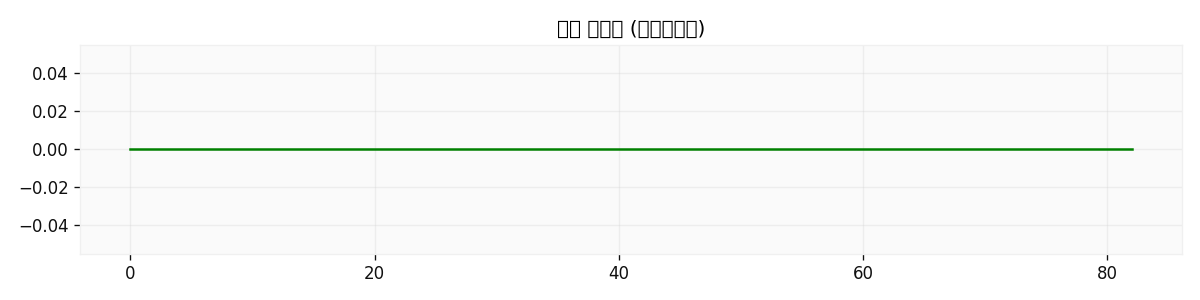
   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**나스닥의 모든 종목을 받아와,  
   ****  
   nasdaqlisted.txt로 저장한다.
5. **유효한 심볼 추출**

**텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

1. **시각화 (viz\_panel)**matplotlib 라이브러리로 시각화했다.  
   **데이터/라벨링이 잘못되어 시각화는 현재 실패했지만, 데이터만 정상적이라면 기능적으로는 사용가능하다.**
2. **예측 캔들 시각화  
   **
3. **30분봉 수익률 변화량  
   텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**
4. **주가 방향 예측 시각화  
   텍스트, 스크린샷, 라인이(가) 표시된 사진

   AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**
5. **상승/관망/하락 비율을 보여주는 원형 차트 시각화  
   **
6. **포지션 추천 변화를 시간축에 따라 나타내어주는 시각화  
   **
7. **수익률 회귀예측을 시계열로 그린 선 그래프 시각화  
   **
8. **이후 추가 가능한 요소**
9. **차트 분석 지표 재추가**  
   차트도구(RSI, MACD, 볼린저밴드 등) 라벨링을 다시 활성화
10. **앨리엇파동 학습**차트분석에서 흔히 쓰이는 “앨리엇 파동 이론”을 AI가 직접 학습하고 차트 분석에 사용할 수 있게 함.

Git repository URL : <https://github.com/groun519/PROJECT-STK>  
프로젝트 재현을 위해서는 git의 readme.md를 열람해주세요.