



RAINBOW COMPANY
infinite of possibilities

Part 1 팀 원 소개

데이터 수집 및 전처리

오정화

전원석

김서영

강화학습

형수진

김현수

웹

전찬

기획 및 총괄

라수정

Contents

1

개요

Introduction

2

수집 및 전처리

Data collection & Preprocessing

3

강화학습 모델링

Reinforcement Learning Model

4

웹 구축

Web Development

5

개선사항 및 결론

Room for improvement & Conclusion



Part 1

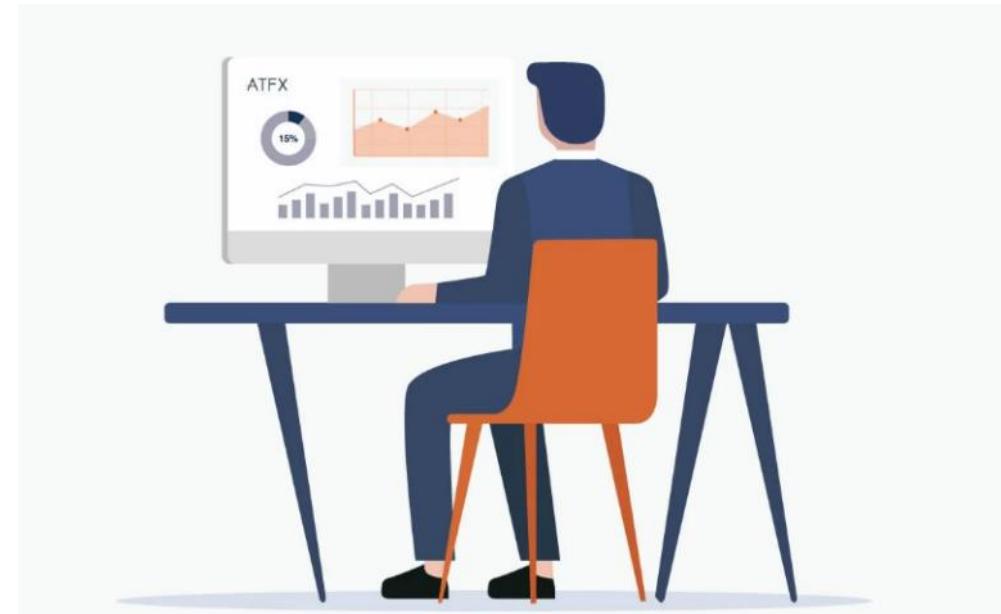
개요

Introduction



Targeting Stock Investment Trading

- 목적 : 고객이 원하는 종목 (이하 메인 종목) 의 매매 시점 추천 서비스 제공
- 과정 :
 - 1) 메인 종목의 단기적 탐색 -> 재무재표, 최신 뉴스 등 조사
 - 2) 메인 종목과 가격 흐름이 비슷한 서브종목(및 지표) 구성
 - 3) 뉴스 검색 키워드 선정
 - 4) 모델 학습용 과거데이터 수집 및 전처리
 - 5) 실시간 데이터 수집 및 전처리 자동화
 - 6) 강화학습 모델 구성 및 테스트
 - 7) 실시간 매수 / 매도 / 관망 추천
 - 8) 웹 서비스 제공



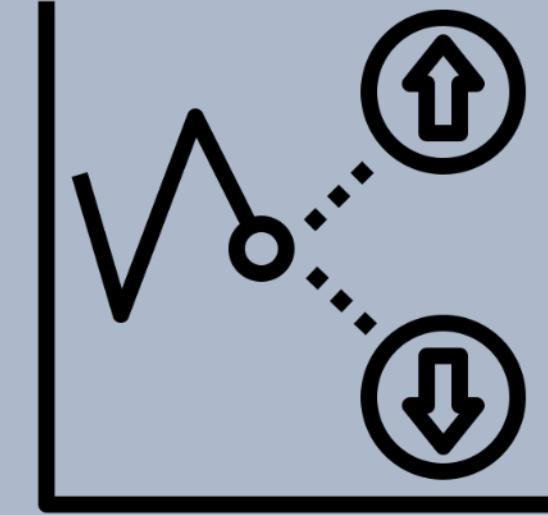
Part 1 개요 - 목적 및 방향성



고객 원하는 종목/투자금액



단기적 탐색



매수/매도 예측

Part1 개요 - 환경

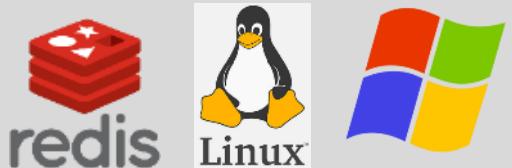
수집



데이터분석



연동 시스템

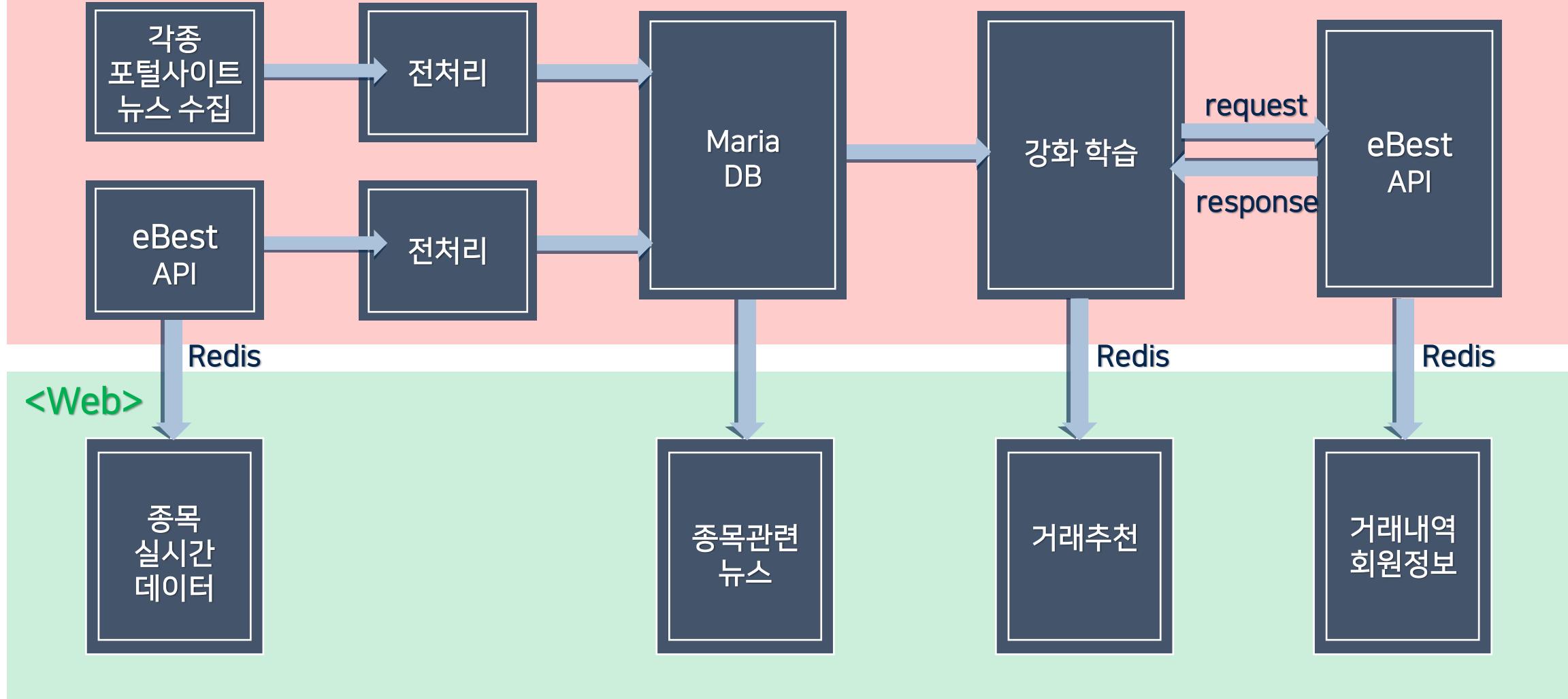


Web 구축



Part 1 개요 - 프로세스

<Linux Google VM Computer>



Part 1 개요 - 개발 일정

Part 2

수집 및 전처리

Data collection & Preprocessing



Part 2 수집 및 전처리 - 과거 데이터 수집

- 수집 기간 : 2021년 6월 28일 ~ 2023년 7월 20일

- 수집 단위 : 틱 (Tick), 분 (minute)

데이터	수집 출처	수집 방법	특성
종목	대신증권 eBest증권	대신증권 CYBOS eBest XING	<ul style="list-style-type: none">- 신세계, 신세계 푸드, 이마트, 신세계건설, 신세계 I&C, 현대건설, CJ제일제당
코스피	대신증권	대신증권 CYBOS	
환율	신한은행	Selenium	<ul style="list-style-type: none">- 달러 (USD/KRW), 위안 (CNY/KRW)
종목 뉴스	'네이버 증권' 뉴스 검색	BeautifulSoup	<ul style="list-style-type: none">- 종목 키워드 검색 (종목 명, 종목 관련 키워드, 트렌드 키워드)

Part 2 수집 및 전처리 - 과거 데이터 전처리

- Datetime 형식 : '%Y-%m-%d %H:%M:%S'
- TimeLine : 매일 09:01 ~ 15:20
- 종가 데이터 1분씩 위로 당기기 (과거 분봉 종가는 1분씩 밀려서 제공됨)

	구분	전처리 내용
종목	2021.06.28 ~ 2022.07.09	<ul style="list-style-type: none">- 데이터 일별 제공 -> 모두 병합- Tick data (주문 발생 건 데이터) -> 분봉 (minute) 데이터로 변환- 신세계 I&C 매매거래 중지일 (22년 4월 6,7,8일) 전일 종가 데이터로 채움
	2021.07.10 ~ 2023.07.14	<ul style="list-style-type: none">- 데이터 일별 제공 -> 모두 병합
코스피		<ul style="list-style-type: none">- 데이터 일별 제공 -> 모두 병합- 개장시간 1시간 지연(10시 개장) 일자 (수능일, 매년 첫 시장) TimeLine 맞춤
환율	달러 / 환율	<ul style="list-style-type: none">- 데이터 일별 제공 -> 모두 병합- Tick data (주문 발생 건 데이터) -> 분봉 (minute) 데이터로 변환- 폐장일 (21년 12월31일) 데이터 Drop

Concat

Part 2 수집 및 전처리 - 과거 데이터 전처리

- 지표 컬럼 추가 : 모든 컬럼의 단위를 맞추고자 모두 비율로 변환

구분	추가 지표	지표 설명
종목	Ratio_close (close - last close) / last close	직전 종가 대비 현재 종가
	Ratio_high_close (high - last close) / last close	직전 종가 대비 현재 고가
	Ratio_low_close (low - last close) / last close	직전 종가 대비 현재 저가
	Ratio_open_close (open - last close) / last close	직전 종가 대비 현재 시가
	Ratio_volume (volume - last volume) / last volume	직전 거래량 대비 현재 거래량
	Ratio_MA20_close	직전 20분 동안의 가격을 산술평균한 값
	Ratio_MA20_volume	직전 20분 동안의 거래량을 산술평균한 값
	Sign(label)	직전가 대비 현재가 상승(1) / 하락(-1) / 보합(0)
코스피 / 환율	Change (close - last close) / last close	직전가 대비 현재가
	Ratio_MA20_close	직전 20분 동안의 가격을 산술평균한 값

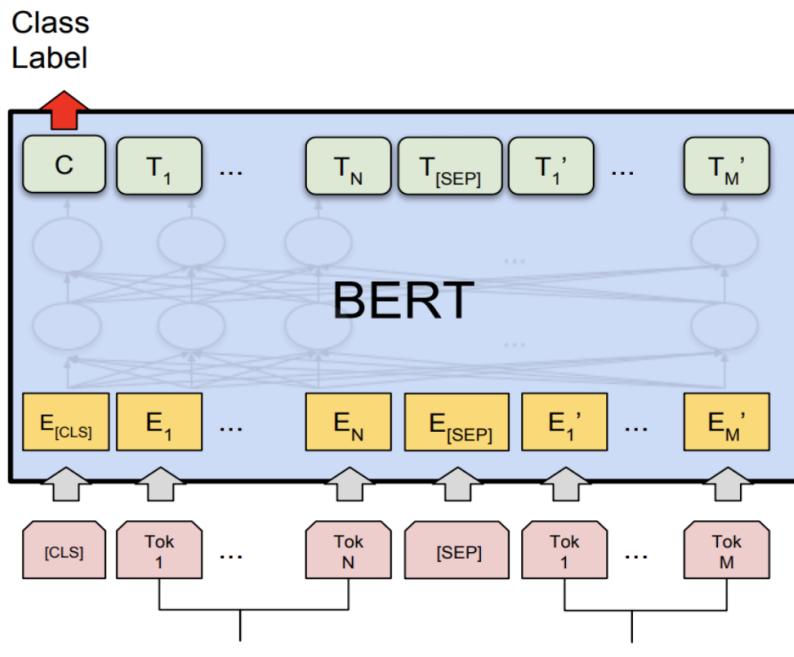
Part2 수집 및 전처리 - 뉴스 데이터셋 전처리

자연어 분석 Model

KR-FinBERT을 활용하여 감성분석을 진행하였습니다.

KR-FinBERT는 BERT 모델 중 한국 금융 분야에 특화된 모델로 텍스트에 대하여 토큰화와 불용어 제거 과정을 수행 가능합니다.

전처리된 텍스트가 입력되면 긍정, 부정의 부문에서 각자 어느 정도의 비율을 내포하는지 환산해줍니다.



[BERT 모델 구성]

```
# 토크나이저 지정
tokenizer = AutoTokenizer.from_pretrained("snunlp/KR-FinBert", return_tensors='pt')

Downloading (...)okenizer_config.json: 100%    336/336 [00:00<00:00, 26.8kB/s]
Downloading (...)solve/main/vocab.txt: 100%    143k/143k [00:00<00:00, 414kB/s]
Downloading (...)main/tokenizer.json: 100%    294k/294k [00:00<00:00, 574kB/s]
Downloading (...)cial_tokens_map.json: 100%    112/112 [00:00<00:00, 8.33kB/s]
```

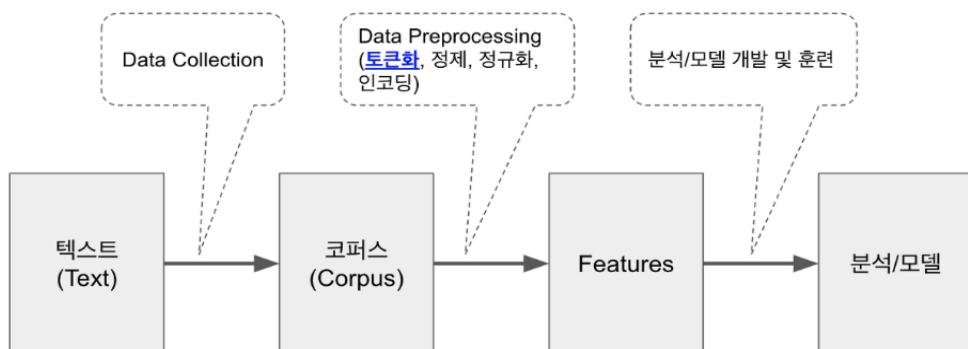
[KR-finBERT 활용]

Part2 수집 및 전처리 - 뉴스데이터셋 전처리

Pipeline를 통해 “토큰화 -> KR-FinBERT -> 결과 전송” 과정을 연결 (긍정, 부정 비율 환산처리를 단계별로 동시에 수행하여 예측 값 추출)

abc NewsDate	abc NewsTitle
2022-12-08 06:00:00	『2030 눈길 솔리네』 신세계百, 신진 디자이너 모시기 열전』
2022-08-10 14:17:00	『2분기·상반기 사상 최대 실적』 신세계, 온·오프 고른 성장 빛나
2022-08-10 14:17:00	『2분기·상반기 사상 최대 실적』 신세계, 온·오프 고른 성장 빛나
2022-11-10 14:46:00	『2천만원대 카라반 쏜다』...신세계, 랜더스 우승 기념 전사 이
2023-05-01 09:42:00	『5월엔 대전신세계 아트앤사이언스서 신나게 놀아볼까?』
2022-08-11 06:00:00	『60만명 와인, 최대 85% 할인』 신세계百, 와인 할인행사』
2023-01-12 13:43:00	『6병 한정』 신세계백화점, 발베니 42년산 선보여』
2023-04-27 09:10:00	『75종의 도어 음선』...신세계까사, 불박이장 신제품 출시』
2023-02-06 08:53:00	『7일 자정부터』 신세계라이브쇼핑, 갤럭시 S23 예약판매』
2022-08-03 09:13:00	『80% 할인부터 환율 보상까지』 신세계免, 대규모 행사』

[DB 뉴스데이터셋]



[자연어 처리 과정 토큰화의 위치]

```
models = AutoModelForSequenceClassification.from_pretrained("snunlp/KR-FinBert")  
  
classifier = pipeline("sentiment-analysis", tokenizer=tokenizer, model=models, device=0)  
  
results = []  
label_ = []  
  
for text in data_kr:  
    result = classifier(text)  
    results.append(result)  
  
# label_에 긍, 부정과 그에 대한 scores 확률이 담김  
for i, result in enumerate(results):  
    for r in result:  
        label_.append(r['label']) # label로 긍정과 부정만 추출  
data['predict']=label_ # predict 컬럼에 담기  
  
data['predict']=data['predict'].map({'LABEL_0':1,'LABEL_1':2})
```

[KR-finBERT 활용 감성분석]

Part 2 수집 및 전처리 - 뉴스데이터셋 전처리

자연어 Model 감성분석결과

```
data['predict']=data['predict'].map({'LABEL_0':1,'LABEL_1':2}) # LABEL_0은 긍정, LABEL_1은 부정이므로 1,2로 긍정,부정을 나눔
```

kor_sentence	predict
국제 전자산업 회사인 엘코텍은 탈린 공장에서 수십 명의 직원을 해고했으며, 이전의 ...	2
새로운 생산공장으로 인해 회사는 예상되는 수요 증가를 충족시킬 수 있는 능력을 증가...	1
2009-2012년 회사의 업데이트된 전략에 따르면, Basware는 20% - 4...	2
ASPOCOMP의 성장기에 대한 자금 조달은 기술적으로 더 까다로운 HDI 인쇄 회...	2
2010년 4분기 Componenta의 순매출은 전년 동기의 7600만 유로에서 2...	1
...	...
헬싱키 톰슨 파이낸셜 - 카고텍의 주가는 화물 취급 그룹이 4월부터 6월까지 3개월...	1
런던 마켓워치 -- 은행주의 반등이 FTSE 100지수의 약세를 상쇄하지 못하면서 ...	1
영업이익은 2007년 68.8 mn에서 35.4 mn으로 떨어졌으며, 선박 판매 이...	2
페이퍼 부문 순매출은 2008년 2분기 241.1 mn에서 2009년 2분기 221...	1
핀란드에서의 판매는 1월에 10.5% 감소한 반면, 국외에서의 판매는 17% 감소했다.	2

Part 2 수집 및 전처리 - 뉴스데이터셋 전처리

자연어 Model 검증

[금융뉴스 기사 제목 데이터셋 활용 검증]

* 종목과 관계없는 데이터를 통한 검증진행

```
25 dl = data['labels']
26 dp = data['predict']
27
28 for idx, da in enumerate(dl):
29     if (dp[idx] == 1) and (dl[idx] == 1):
30         TP += 1
31     elif (dp[idx] == 2) and (dl[idx] == 2):
32         TN += 1
33     elif (dp[idx] == 1) and (dl[idx] == 2):
34         FP += 1
35     elif (dp[idx] == 2) and (dl[idx] == 1):
36         FN += 1
37
38 precision = TP / (TP + FP)
39 recall = TP / (TP + FN)
40 accuracy = (TP + TN) / (TP + TN + FP + FN)
41 f1_score = 2 * (precision * recall) / (precision + recall)
42
43 print('정밀도(Precision): ', precision)
44 print('재현율(Recall): ', recall)
45 print('F1 score: ', f1_score)
46 print('정확도(Accuracy): ', accuracy)
47
```

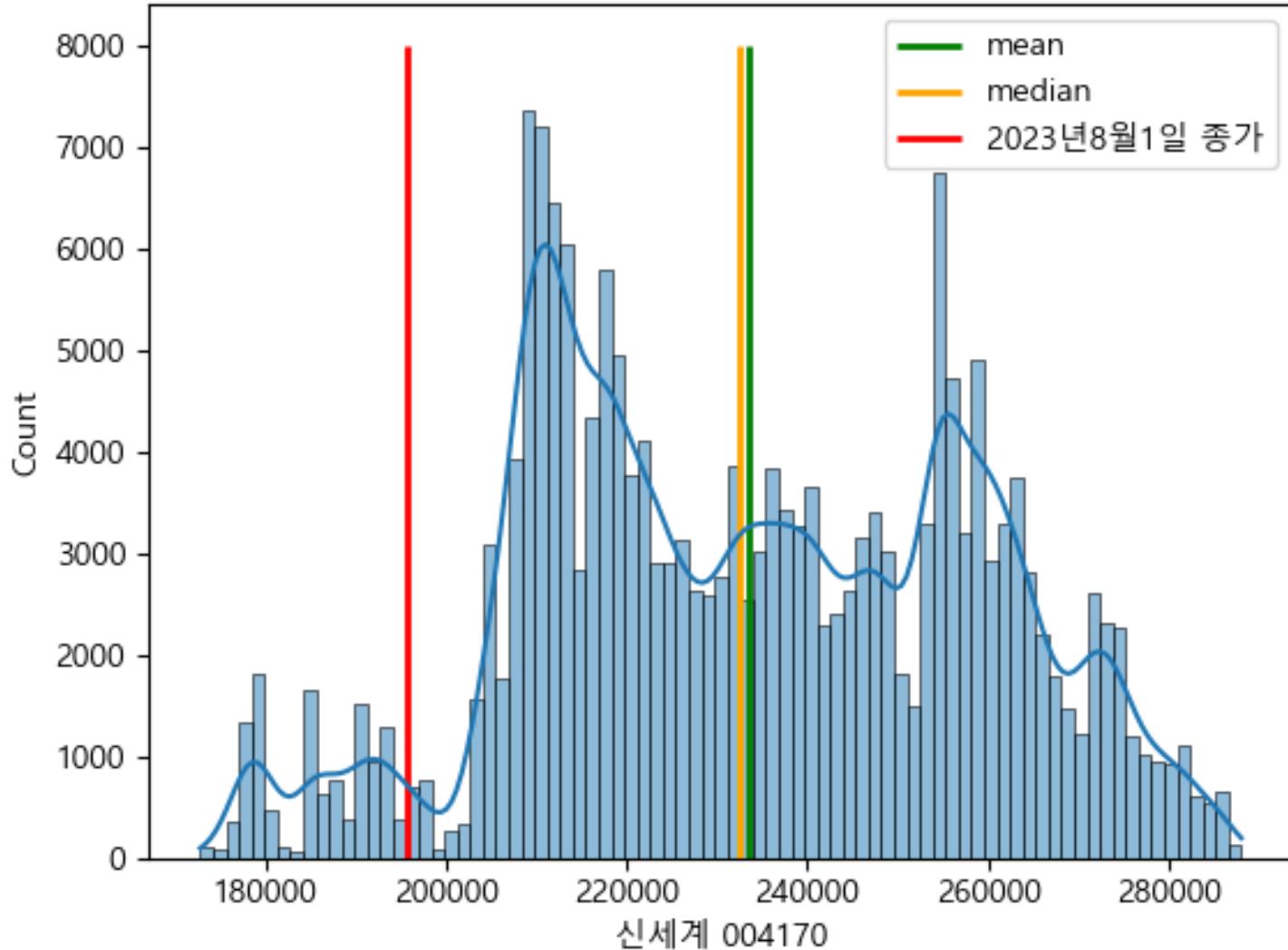
```
정밀도(Precision): 0.7010036978341257
재현율(Recall): 0.9743024963289281
F1 score: 0.8153609831029186
정확도(Accuracy): 0.7243031536113937
```

[종목 뉴스기사 제목 데이터셋 활용 검증]

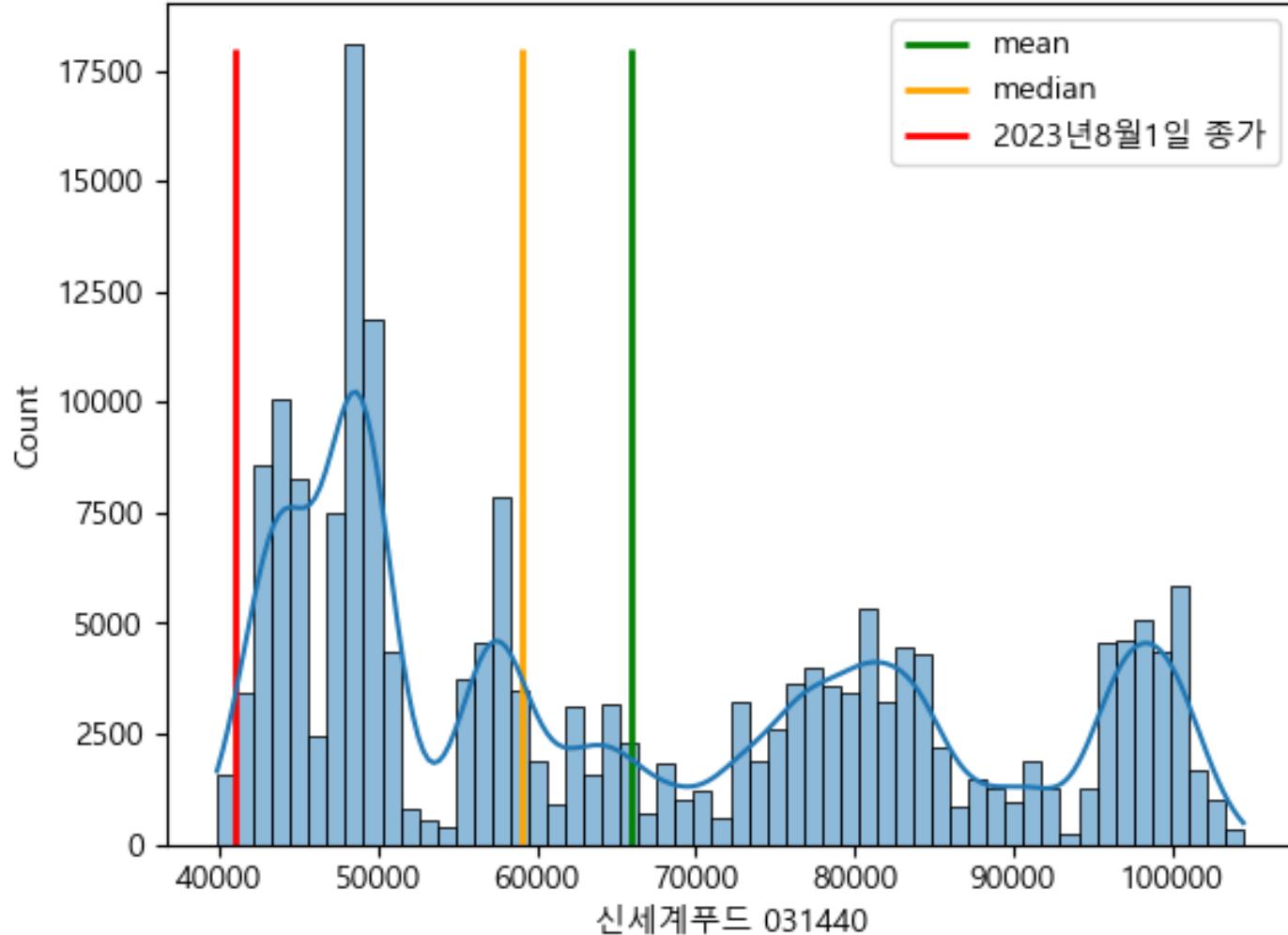
```
6 sent=shin['sentiment']
7 pred=shin['predict']
8 for idx, da in enumerate(sent):
9     if (pred[idx] == 1) and (sent[idx] == 1):
10        TP += 1
11    elif (pred[idx] == 2) and (sent[idx] == 2):
12        TN += 1
13    elif (pred[idx] == 1) and (sent[idx] == 2):
14        FP += 1
15    elif (pred[idx] == 2) and (sent[idx] == 1):
16        FN += 1
17
18 precision = TP / (TP + FP)
19 recall = TP / (TP + FN)
20 accuracy = (TP + TN) / (TP + TN + FP + FN)
21 f1_score = 2 * (precision * recall) / (precision + recall)
22
23 print('정밀도(Precision): ', precision)
24 print('재현율(Recall): ', recall)
25 print('F1 score: ', f1_score)
26 print('정확도(Accuracy): ', accuracy)
27
```

```
정밀도(Precision): 0.8189245887159103
재현율(Recall): 0.8779592903030303
F1 score: 0.8474150242787775
정확도(Accuracy): 0.7958610846812559
```

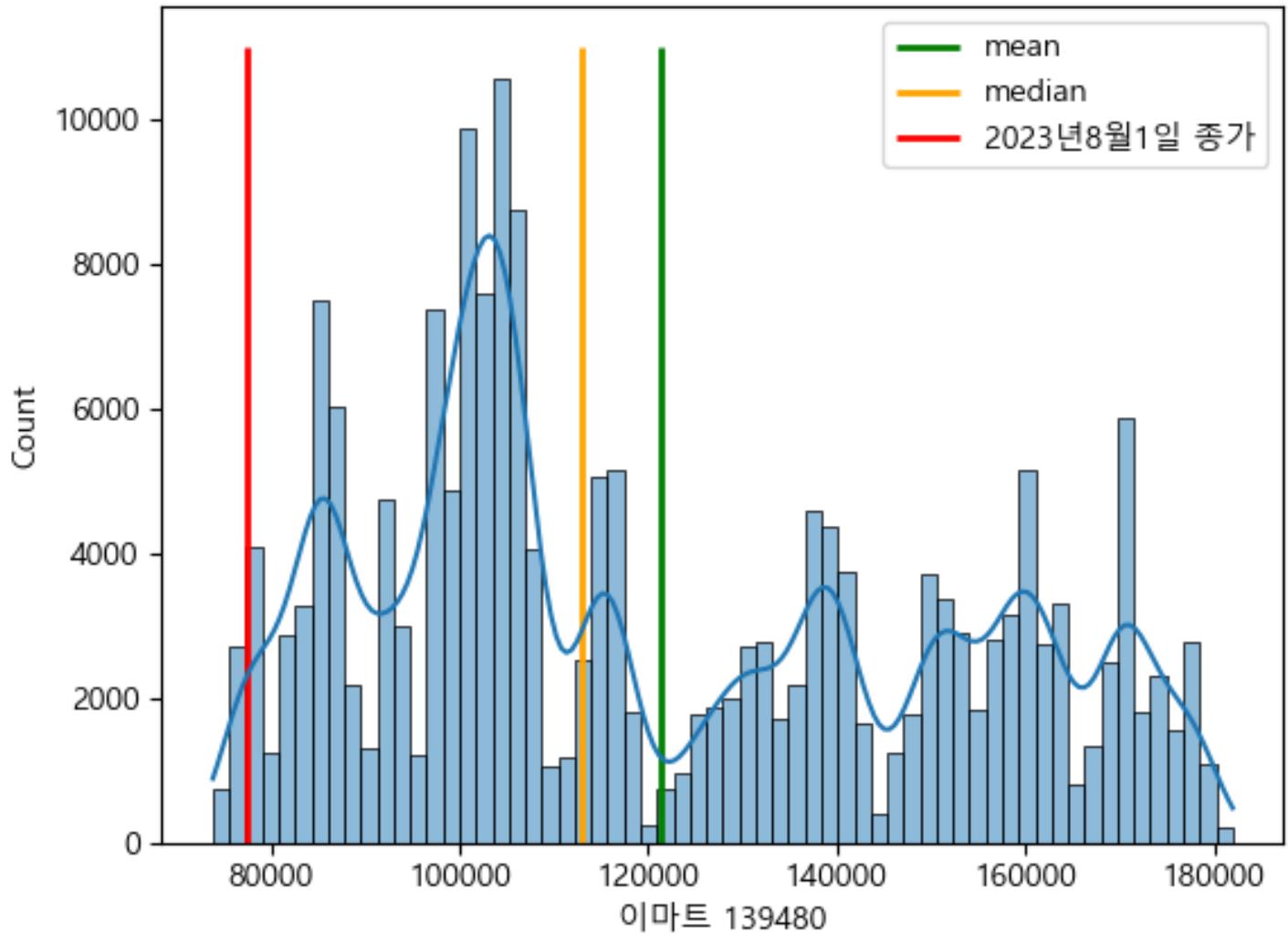
Part 2 수집 및 전처리 - 분석 시각화



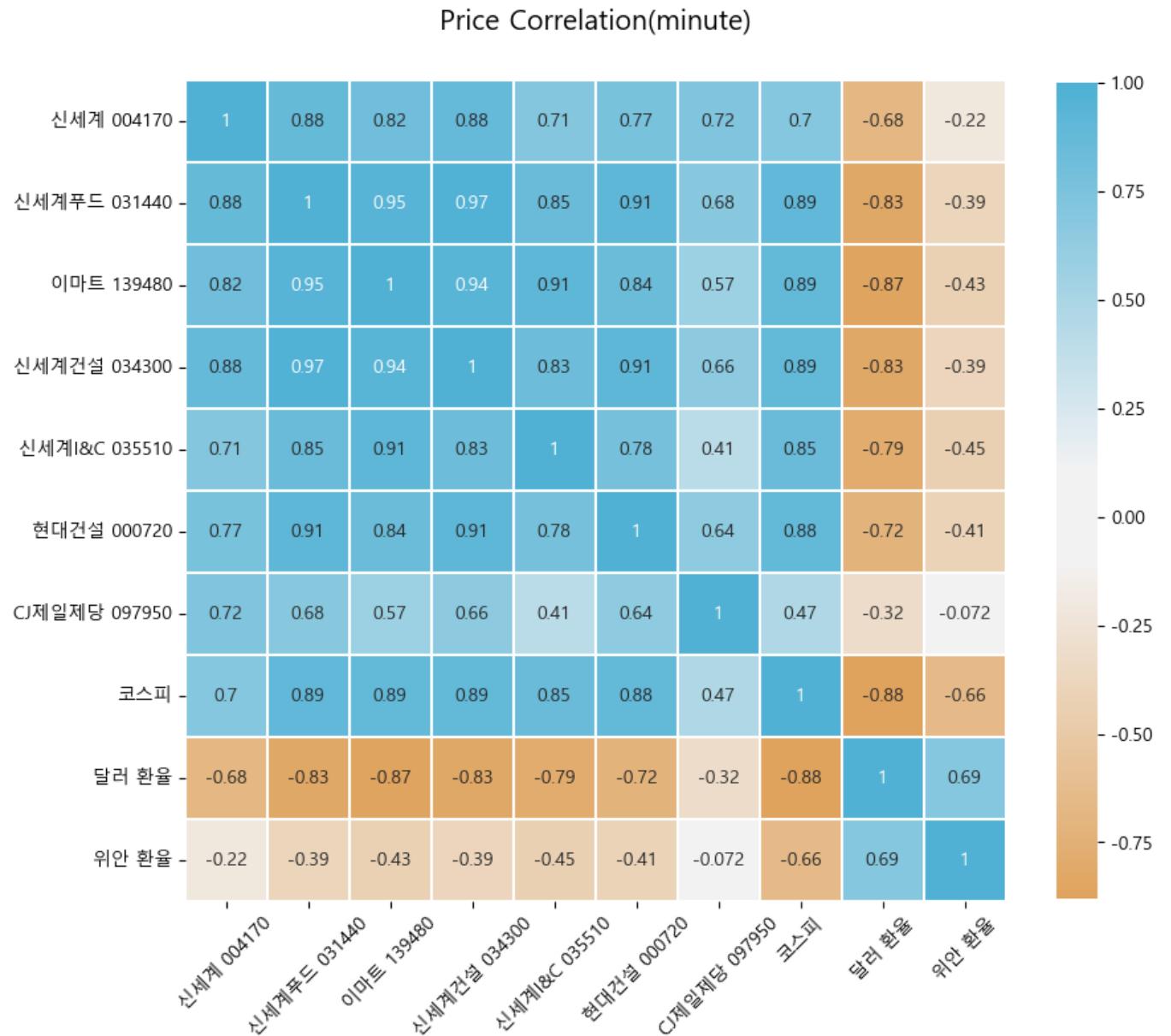
Part 2 수집 및 전처리 - 분석 시각화



Part 2 수집 및 전처리 - 분석 시각화



Part 2 수집 및 전처리 - 분석 시각화

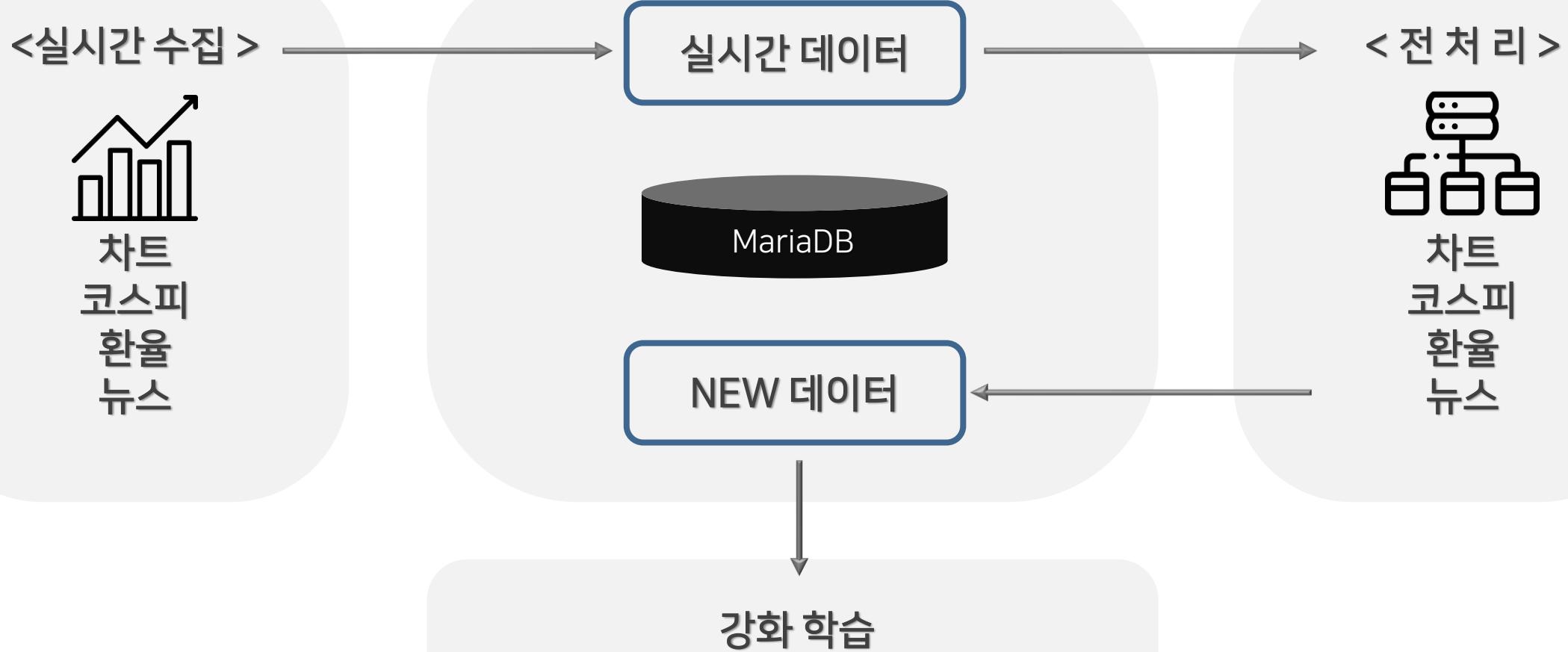


Part 2 수집 및 전처리 - 실시간 데이터 수집

- 수집 기간 : 2023년 7월 21일 ~
- 수집 단위 : 분 (minute)

데이터	수집 출처	수집 방법	저장 방법
종목	이베스트증권	eBest OPEN API	<ul style="list-style-type: none">- Chart (Open High Low Close Volme) => DB : CHART_종목번호, Redis : 시간, 현재가(Close)
코스피	네이버 증권	BeautifulSoup	<ul style="list-style-type: none">- DB : kospicrawling, Redis : 시간, 현재가
환율	네이버 증권	BeautifulSoup	<ul style="list-style-type: none">- DB : exchange_crawling, Redis : 시간, 현재가
종목 뉴스	'네이버 증권' 뉴스 검색	BeautifulSoup	<ul style="list-style-type: none">- DB : shin/shinFood/eMart

Part 2 수집 및 전처리 - 실시간 데이터 전처리



Part 3

강화 학습 모델링

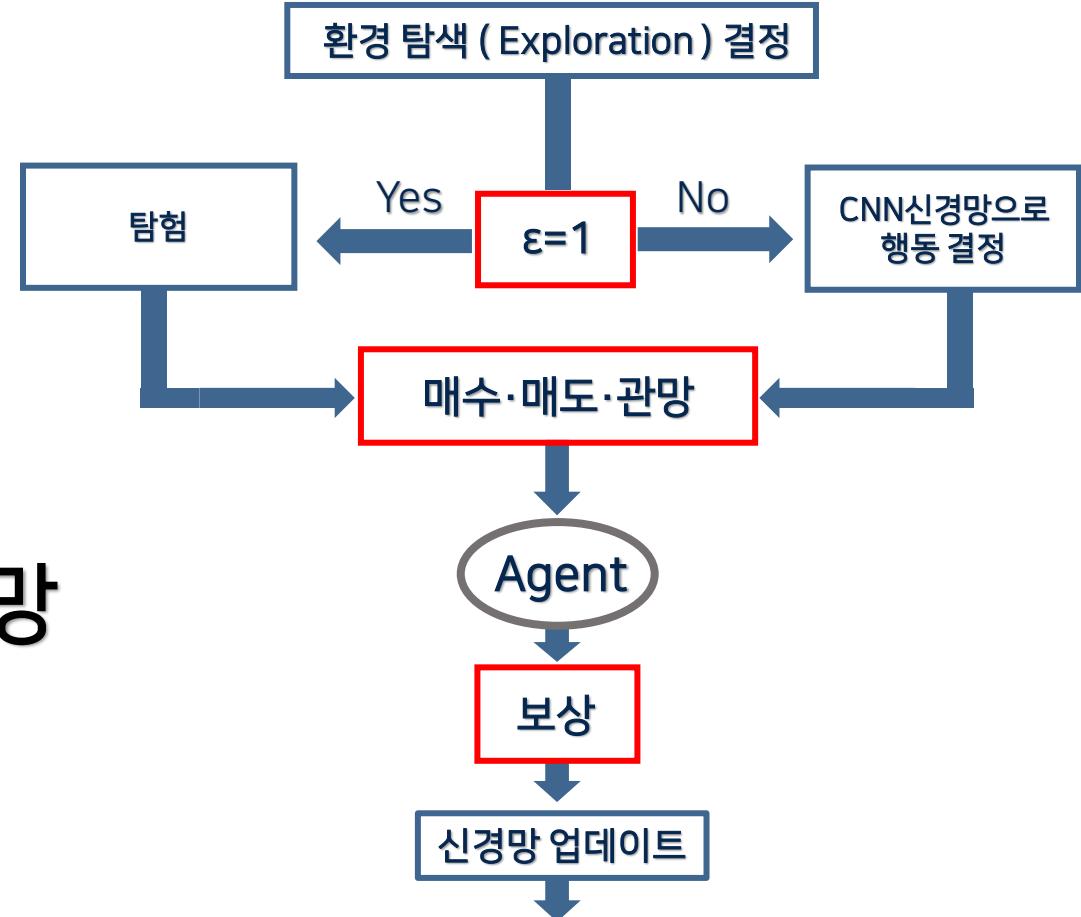
Reinforcement Learning Modeling



Part 3 강화학습 모델링 - 강화학습순서도

어떤 환경에서 행동을 했을 때
잘 된 행동인지 잘못된 행동인지를 판단하고
보상을 줌으로써 반복을 통해 스스로 학습

강화학습 A3C 기법 · CNN 신경망



Part 3 강화학습 모델링 - RLTrader

```
def act(self, action, confidence):
    if not self.validate_action(action):
        action = Agent.ACTION_HOLD

    # 환경에서 현재 가격 얻기
    curr_price = self.environment.get_price()

    # 매수
    if action == Agent.ACTION_BUY:
        # 매수할 단위를 판단
        trading_unit = self.decide_trading_unit(confidence)
        balance = (
            self.balance - curr_price * 
            (1 + self.TRADING_CHARGE) * trading_unit
        )
        # 보유 현금이 모자랄 경우 보유 현금으로 가능한 만큼 최대한 매수
        if balance < 0:
            trading_unit = min(
                int(self.balance / (curr_price * (1 + self.TRADING_CHARGE))),
                int(self.max_trading_price / curr_price)
            )
        # 수수료를 적용하여 총 매수 금액 산정
        invest_amount = curr_price * (1 + self.TRADING_CHARGE) * trading_unit
        if invest_amount > 0:
            self.avg_buy_price = \
                (self.avg_buy_price * self.num_stocks + curr_price * trading_unit) \
                / (self.num_stocks + trading_unit) # 주당 매수 단가 갱신
            self.balance -= invest_amount # 보유 현금을 갱신
            self.num_stocks += trading_unit # 보유 주식 수를 갱신
            self.num_buy += 1 # 매수 횟수 증가
```

```
# 매도
elif action == Agent.ACTION_SELL:
    # 매도할 단위를 판단
    trading_unit = self.decide_trading_unit(confidence)
    # 보유 주식이 모자랄 경우 가능한 만큼 최대한 매도
    trading_unit = min(trading_unit, self.num_stocks)
    # 매도
    invest_amount = curr_price * (
        1 - (self.TRADING_TAX + self.TRADING_CHARGE) * trading_unit
    )
    if invest_amount > 0:
        # 주당 매수 단가 갱신
        self.avg_buy_price = \
            (self.avg_buy_price * self.num_stocks - curr_price * trading_unit) \
            / (self.num_stocks - trading_unit) \
            if self.num_stocks > trading_unit else 0
        self.num_stocks -= trading_unit # 보유 주식 수를 갱신
        self.balance += invest_amount # 보유 현금을 갱신
        self.num_sell += 1 # 매도 횟수 증가

    # 관망
elif action == Agent.ACTION_HOLD:
    self.num_hold += 1 # 관망 횟수 증가

# 포트폴리오 가치 갱신
self.portfolio_value = self.balance + curr_price * self.num_stocks
self.profitloss = self.portfolio_value / self.initial_balance - 1
return self.profitloss
```

Part 3 강화학습 모델링 - RLTrader

```
def run(self, learning=True):
    # 에포크 반복
    for epoch in tqdm(range(self.num_epochs)):
        time_start_epoch = time.time()

        # step 샘플을 만들기 위한 큐
        q_sample = collections.deque(maxlen=self.num_steps)

        # 환경, 메이저트, 신경망, 가시화, 메모리 초기화
        self.reset()

        # 학습을 진행할 수록 탐험 비율 감소
        if learning:
            epsilon = self.start_epsilon * (1 - (epoch / (self.num_epochs - 1)))
        else:
            epsilon = self.start_epsilon

        for i in tqdm(range(len(self.training_data)), leave=False):
            # 샘플 생성
            next_sample = self.build_sample()
            if next_sample is None:
                break

            # num_steps만큼 샘플 저장
            q_sample.append(next_sample)
            if len(q_sample) < self.num_steps:
                continue
```

```
# 가치, 정책 신경망 예측
pred_value = None
pred_policy = None
if self.value_network is not None:
    pred_value = self.value_network.predict(list(q_sample))
if self.policy_network is not None:
    pred_policy = self.policy_network.predict(list(q_sample))

# 신경망 또는 탐험에 의한 행동 결정
action, confidence, exploration = \
    self.agent.decide_action(pred_value, pred_policy, epsilon)

# 결정한 행동을 수행하고 보상 획득
reward = self.agent.act(action, confidence)

# 행동 및 행동에 대한 결과를 기억
self.memory_sample.append(list(q_sample))
self.memory_action.append(action)
self.memory_reward.append(reward)
if self.value_network is not None:
    self.memory_value.append(pred_value)
if self.policy_network is not None:
    self.memory_policy.append(pred_policy)
self.memory_pv.append(self.agent.portfolio_value)
self.memory_num_stocks.append(self.agent.num_stocks)
if exploration:
    self.memory_exp_idx.append(self.training_data_idx)

# 반복에 대한 정보 갱신
self.batch_size += 1
self.itr_cnt += 1
self.exploration_cnt += 1 if exploration else 0

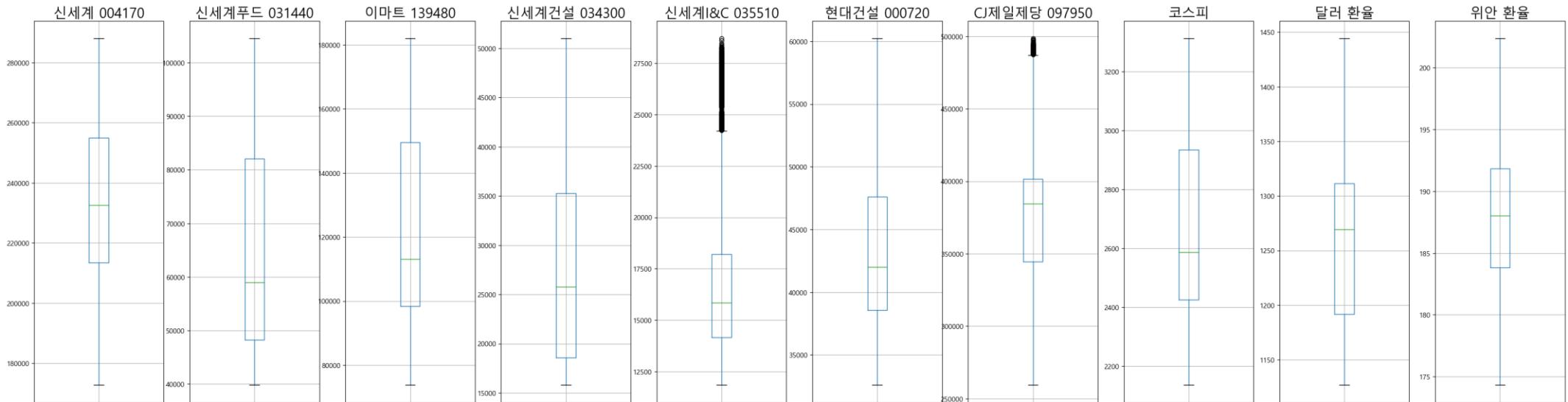
# 에포크 종료 후 학습
if learning:
    self.fit()
```

Part 3 강화학습 모델링 - Robust Scaler

중앙값(median)을 0으로 하고 IQR차이 만큼 정규화 한다.

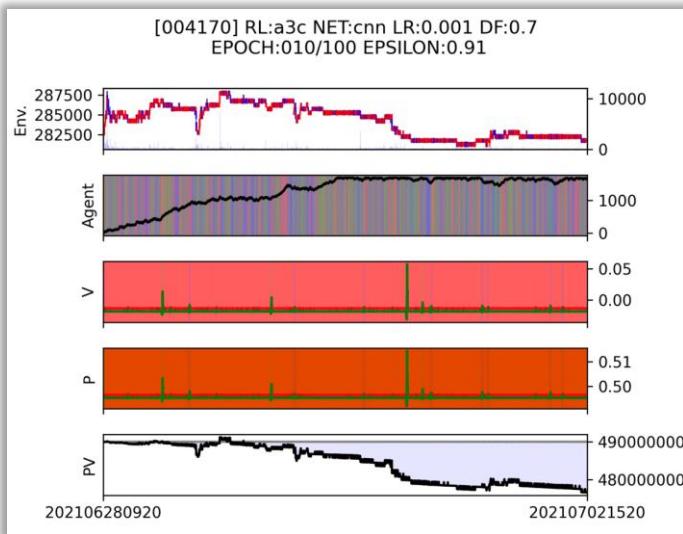
* IQR = Q3(데이터의 75% 이 값보다 작거나 같음) - Q1(데이터의 25% 이 값보다 작거나 같음)

따라서, RobustScaler는 이상치의 영향을 받지 않아 이상치가 많은 데이터를 다룰 때 유용하다.

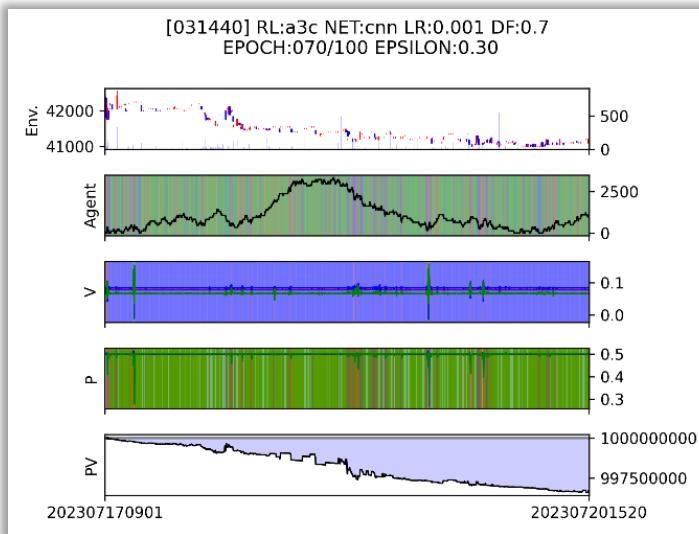


Part 3 강화학습 모델링 - 학습데이터 시각화

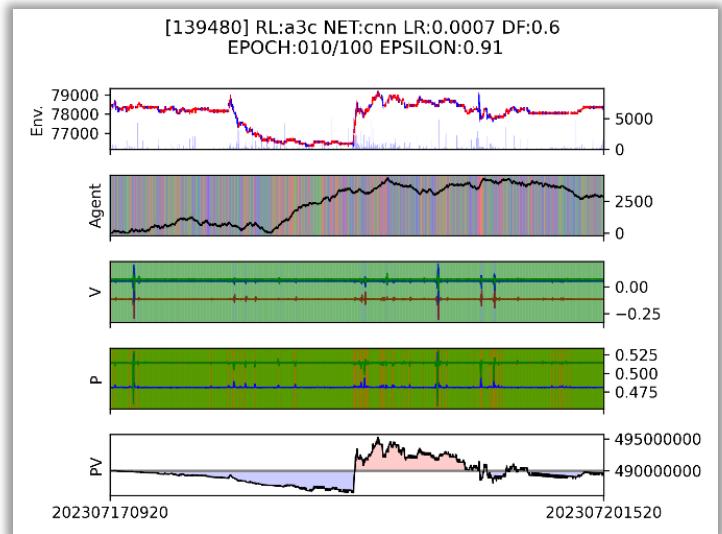
신세계
LR 0.001 / DF 0.7



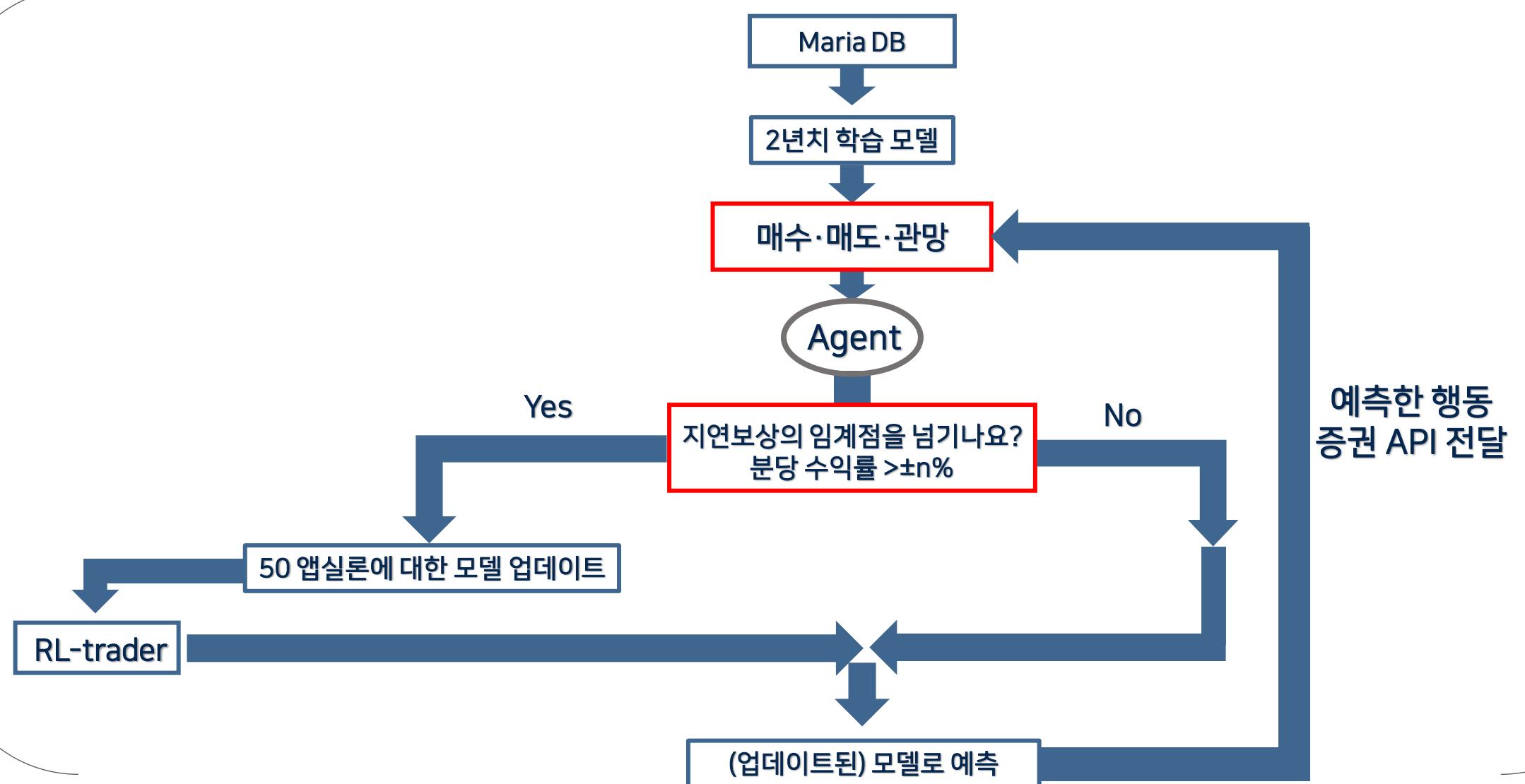
신세계 푸드
LR 0.001 / DF 0.7



이마트
LR 0.007 / DF 0.6



Part 3 강화 학습 모델링 - 실시간데이터 강화 학습



Part 3 강화학습 모델링 - 실시간 update/predict

```
def runrunrun(stock, acc):
    pred_file = 'predict_{}.a3c_cnn'.format(stock) + '/pred_{}.json'.format(stock)
    pred_path = base_path + pred_file

    print(stock)

    print("[Redis] balance를 기다리는 중입니다")
    initial_balance = get_redis(stock + 'balance_channel')
    print('[Redis] eBest api에서 가져온 현재 잔고는 {}'.format(initial_balance))
    now_time1 = (datetime.now() - timedelta(minutes=0)).strftime('%Y-%m-%d %H:%M:%S')
    BBB = datetime.now().strftime('%H:%M:%S')
    print('[시스템] 현재시각은 {}입니다 20초에 predict 시작합니다'.format(BBB))
    ##### 하루의 첫 거래 # 데이터 만들어둘까 대비 09:10분을 첫 학습
    while True:
        now_time = datetime.now().strftime('%S')
        if now_time == '20': #####
            if int(now_time1) > int(today_date_str[:9] + '0910'):
                print('[시스템] 첫거래 predict합니다')
                first_date, _, last_date, sixth = read_close_date_stock(stock=stock)
                next_time, act = predict_command(stock, initial_balance, first_date, last_date, pred_path)

                time.sleep(0.1)
                acc = send_redis(pred_path, stock + 'action_channel', acc, stock)
                print('[Redis] balance를 기다리는 중입니다')
                after_balance = get_redis(stock + 'balance_channel')
                print('[Redis] 첫거래후 잔고는 {}입니다'.format(after_balance))

                break

        BBB = datetime.now().strftime('%H:%M:%S')
        print('[시스템] 현재시각은 {}입니다'.format(BBB))
        # 첫거래후 수익률 계산하는 부분
        print("[Redis] real_close를 기다리는 중입니다")
        real_close = get_redis(stock + 'real_close') # 체결률 증가
        print('첫거래후 eBest api에서 체결증가는 {}입니다'.format(real_close))

        _, now_close, __, __ = read_close_date_stock(stock, last_date) # 가격률 증가
        performance_print = (now_close - int(real_close)) / int(real_close) * 100
        performance_rate = abs(performance_print)

        ##### 수익이 일정률을 넘을때, 업데이트하고, predict도 해서 같이 보내야함
        now_ = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        while True:
            print("====")
            now_time = datetime.now().strftime('%H:%M')
            if int(now_time) > 1521: # 15시20분이면 종료
                print('현재 시각 {} 정상으로 종료합니다'.format(datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
                break
            else:
                BBB = datetime.now().strftime('%H:%M:%S')
                print('[시스템] 현재시각은 {}입니다 20초에 predict 시작합니다'.format(BBB))
```

```
if now_sec != '20':
    if performance_rate >= 0.2 : # 일정 지연 보상 신세계푸드
        if real_close == 0: # 전 action이 광활할 경우
            continue
        print('[시스템] 수익률 {}가 나왔습니다, 모델을 update합니다'.format(performance_print))

    if int(last_date) - int(next_time) > 5:
        update_command(stock, after_balance, next_time, last_date) # 업데이트
    else:
        update_command(stock, after_balance, sixth, last_date) # 업데이트

    next_time, act = predict_command(stock, after_balance, sixth, last_date, pred_path)
    # 50epoch update가 1분내로 된다는 가정하에 predict

    time.sleep(0.1)
    acc = send_redis(pred_path, stock + 'action_channel', acc, stock)
    print('[Redis] eBest api에 action을 보냈습니다')
    print('[Redis] balance를 기다리는 중입니다')
    after_balance = get_redis(stock + 'balance_channel')
    print("eBest api에서 받은 잔고는 {}입니다".format(after_balance))

    _, now_close, __, __ = read_close_date_stock(stock, last_date)
    print('[시스템] 마지막 증가는 {}입니다'.format(now_close))
    AAtime = datetime.now().strftime('%H:%M:%S')
    print('[시스템] 현재시각 {}'.format(AAtime))
    print('[Redis] real_close를 기다리는 중입니다')
    real_close = get_redis(stock + 'real_close')

    print('실제 체결 단가는 {}입니다'.format(real_close))

    performance_print = (now_close - int(real_close)) / now_close * 100
    performance_rate = abs((now_close - int(real_close)) / now_close * 100)

    chan_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    PV = int(after_balance) + int(real_close) * int(acc)
    chan_redis(stock + 'timepy', chan_time)
    chan_redis(stock + 'pv', PV)
    insert_pv(stock, chan_time, PV)

    BBB = datetime.now().strftime('%H:%M:%S')
    print('[시스템] 현재시각은 {}입니다 20초에 predict 시작합니다'.format(BBB))
```

Part 3 강화학습 모델링 - 자동 매매 Redis

```
if bsy==0: # 매수
    trading_unit= int(((prob-0.50)/0.01)+1)
    trading_unit= max(trading_unit,1)
    trading_unit= min(trading_unit,100) # max 100
    print('*****')
    print('{} 종목 {}개 매수합니다'.format(stock,int(trading_unit)))
    acc = acc+1
    action='2'

elif bsy==1: # 매도
    trading_unit= int(((prob-0.50)/0.01)+1)
        # if trading <1
    trading_unit= max(trading_unit,1)
        # if trading >100
    trading_unit= min(trading_unit,100) # max acc or 100
        # if trading > acc
    print('*****')
    print('{} 종목 {}개 매도합니다'.format(stock,int(trading_unit)))
    acc = acc-1
    action='1'

elif bsy==2: # 관망
    print('*****')
    print('{} 종목 헐드합니다'.format(stock))
    trading_unit=0
    action='0'
    acc = acc
```

```
##### 매매하는 부분
if action =='2':
    ACCESS_TOKEN=eba.make_token()
    now_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    # trading_unit=get_redis(stock+'trading_channel')
    print('현재시각 {} : {}주 매수합니다'.format(now_time,int(trading_unit)))
    print('=====')
    log_it(now_time, '매수')
    eba.order(ACCESS_TOKEN,'A'+stock,'buy',int(trading_unit))
    insert_pv(now_time,'매수')
    time.sleep(0.1)

if action =='1':
    ACCESS_TOKEN=eba.make_token()
    now_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    # trading_unit=get_redis(stock+'trading_channel')
    try:
        eba.order(ACCESS_TOKEN,'A'+stock,'sell',int(trading_unit))
        print('=====')
        print('현재시각 {} : {}주 매도합니다'.format(now_time,int(trading_unit)))
        log_it(now_time, '매도')
        insert_pv(now_time,'매도')
    except:
        print('=====')
        print('현재시각 {} : {}주 매도합니다'.format(now_time,int(trading_unit)))
        log_it(now_time, '매도')
        insert_pv(now_time,'매도')

while action=='0':
    real_close = real_close1
    now_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print('\n')
    insert_pv(now_time,'관망')
    print('=====')
    print('현재시각 {} 헐드합니다'.format(now_time))

    print('[Redis] 보내는 잔고는 {}'.format(balance))
    send_redis(stock+'balance_channel', balance)
    log_it(now_time, balance)
    time.sleep(5)
    send_redis(stock+'real_close', real_close1)
    print('[Redis] 보내는 종가는 {}'.format(real_close1))
    log_it(now_time, real_close1)

    print("[Redis] action을 기다리는 중입니다")
    action = get_redis(stock+'action_channel')
    # trading_unit=get_redis(stock+'trading_channel')
    log_it(now_time, '관망')

    if action!='0':
        break
```

Part 3 강화 학습 모델링 - 시연 영상

The screenshot shows a Jupyter Notebook environment with two panes. The left pane displays a terminal window with a log of Redis actions and API calls, indicating a trading bot's activity. The right pane shows the Jupyter interface with code cells and their outputs, illustrating the execution of trading logic.

Terminal Log (Left Pane):

```
4 [시그널] 현재시각은 14:37:32입니다
5 [시스템] 남은 잔고는 489153198입니다
6 [Redis] action을 기다리는 중입니다
7 [Redis] unit을 기다리는 중입니다
8 [API] 체결단가를 가져오는중..
9 [API] 체결 단가를 가져옵니다
10 수정된1 real_close:195600
11 [시스템] 현재시각 14:38:26, 1개를 매수합니다
12 모의투자 매수주문이 원로 되었습니다.
13 [API] 잔고를 가져옵니다
14 [API] 체결단가를 가져오는중..
15 [API] 체결 단가를 가져옵니다
16 수정된2 real_close:195600
17 real_close2로 들어옵니다
18 [Redis] 남은 잔고는 489153198입니다 보냅니다
19 [Redis] 체결단가는 195600입니다 학습하게 보냅니다
20
21 004170
22 [API] 잔고를 가져옵니다
23 [시스템] 현재시각은 14:38:33입니다
24 [Redis] action을 기다리는 중입니다
25 [Redis] unit을 기다리는 중입니다
26 [Redis] unit을 기다리는 중입니다
27 [API] 체결단가를 가져오는중..
28 [API] 체결 단가를 가져옵니다
29 수정된1 real_close:195600
30 [시스템] 현재시각 14:38:39, 1개를 매수합니다
31 모의투자 매수주문이 원로 되었습니다.
32 [API] 잔고를 가져옵니다
33 [API] 체결단가를 가져오는중..
34 [API] 체결 단가를 가져옵니다
35 수정된2 real_close:195600
36 real_close2로 들어옵니다
37 [Redis] 남은 잔고는 489153198입니다 보냅니다
38 [Redis] 체결단가는 195600입니다 학습하게 보냅니다
39
40 004170
41 [API] 잔고를 가져옵니다
42 [시스템] 현재시각은 14:38:46입니다
43 [시스템] 남은 잔고는 489155170입니다
44 [Redis] action을 기다리는 중입니다
45
```

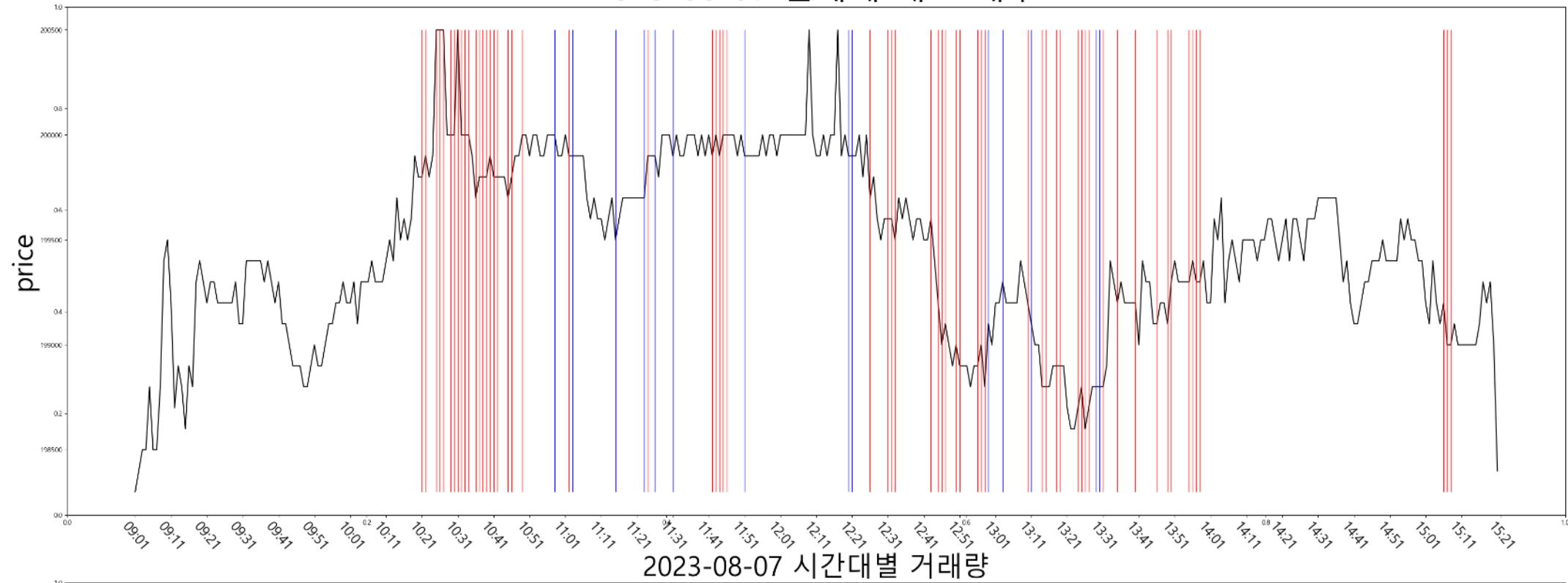
Jupyter Notebook (Right Pane):

```
In [1]: runfile('C:\Users\user\PycharmProjects\d1_trade-real\Copy1.py', wdir='C:\Users\user\PycharmProjects\d1_trade-real')
In [2]: acc = 0
running stack acc
004170
[Redis] balance를 기다리는 중입니다
[Redis] effect api에서 가져온 현재 잔고는 489153198입니다
[시스템] 현재시각은 14:38:33입니다 203에 predict 시작합니다
[시스템] 초기 이후 잔고는 489153198입니다
Enabling Pytorch
[시스템] predict 완료
004170: 매수합니다
[Redis] balance를 기다리는 중입니다
[Redis] 첫거래후 잔고는 489153198입니다
[시스템] 현재시각은 14:38:33입니다
[Redis] real_close를 기다리는 중입니다
첫거래후 effect api에서 체결증기는 195600입니다

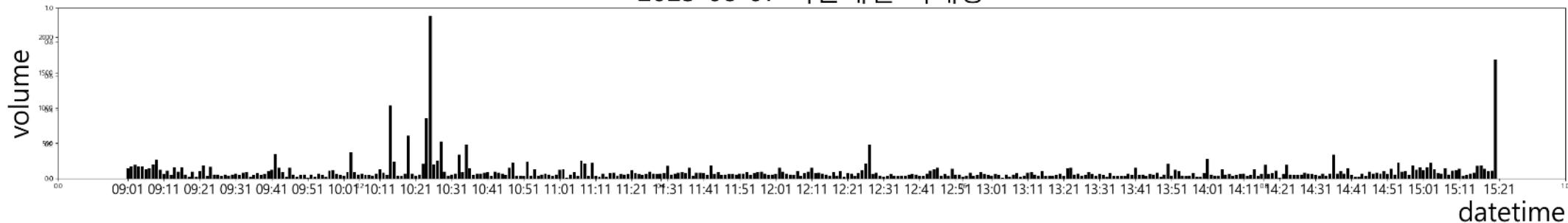
[시스템] 현재시각은 14:38:33입니다 203에 predict 시작합니다
[시스템] 수익률 0.0%가 나왔습니다 predict합니다
Enabling Pytorch
[시스템] predict 완료
004170: 풍락 매수합니다
[Redis] effect api에 action을 보냈습니다
[Redis] balance를 기다리는 중입니다
[Redis] effect api에서 말로 잔고는 489154870입니다
마지막 잔액은 195600입니다
[시스템] 현재시각 14:38:45
[Redis] real_close를 기다리는 중입니다
실제 체결 단가는 195600입니다
[시스템] 현재시각은 14:38:45입니다 303에 predict 시작합니다
```

Part 3 강화 학습 모델링 - 매매 내역

2023-08-07 신세계 매도 매수



2023-08-07 시간대별 거래량



Part 4

웹 구축

Web Development



Part4 웹 구축 - 회원가입



회원가입

프로필 이미지
 파일 선택 | 선택된 파일 없음

이름

아이디

비밀번호

비밀번호 표시:

가입하기

이미 계정이 있으십니까? [로그인하기](#)

[회원가입 화면]

컬럼명	#	Data Type	Not Null	Auto Increment	Key	디폴트
username	1	varchar(100)	[v]	[]		
imgnm	2	varchar(2000)	[]	[]		NULL
userid	3	varchar(500)	[v]	[]	PRI	
userpassword	4	varchar(500)	[v]	[]		

username	imgnm	userid	userpassword
유저	/images/사람.png	user	jeonchan

[DB에 저장된 정보 화면]

Part4 웹 구축 - 로그인



로그인

아이디

비밀번호

로그인 정보 저장

로그인

회원이 아니십니까? 계정 만들기

[로그인 화면]

localhost:8080 내용:
로그인에 성공하셨습니다.

확인

localhost:8080 내용:
로그인에 실패하셨습니다. 다시 로그인해주세요

확인

[로그인 성공 및 실패 확인 메시지 화면]

Part4 웹 구축 - 로그인

```
if (check == 1) {
    HttpSession session = request.getSession();
    session.setAttribute("userid", userid);
    session.setAttribute("userpassword", userpassword);
    red.addFlashAttribute("msg", "<script>alert('로그인에 성공하셨습니다.');//</script>");
} else {
    red.addFlashAttribute("msg", "<script>alert('로그인에 실패하셨습니다. 다시 로그인해주세요');//</script>");
    return "redirect:/rainbowcompany/signin";
}
```

[Session에 로그인 정보 저장 코드 화면]

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    HttpServletRequest req = (HttpServletRequest) request;
    HttpServletResponse res = (HttpServletResponse) response;
    HttpSession session = req.getSession();
    String logincheck = (String) session.getAttribute("userid");
    if(logincheck == null) {
        res.sendRedirect("/rainbowcompany/main");
    }else {
    }
    chain.doFilter(request, response);
}
```

[Filter를 통해 로그인 회원만 페이지 이동 가능 코드 화면]

Part4 웹 구축 - Redis & Websocket

```
@Component
public class RedisMessageListener implements MessageListener {

    private final SimpMessagingTemplate messagingTemplate;

    @Autowired
    public RedisMessageListener(SimpMessagingTemplate messagingTemplate) {
        this.messagingTemplate = messagingTemplate;
    }

    @Override
    public void onMessage(Message message, byte[] pattern) {
        String topic = new String(message.getChannel());
        String value = new String(message.getBody());
        messagingTemplate.convertAndSend("/stock/" + topic, value);
    }
}
```

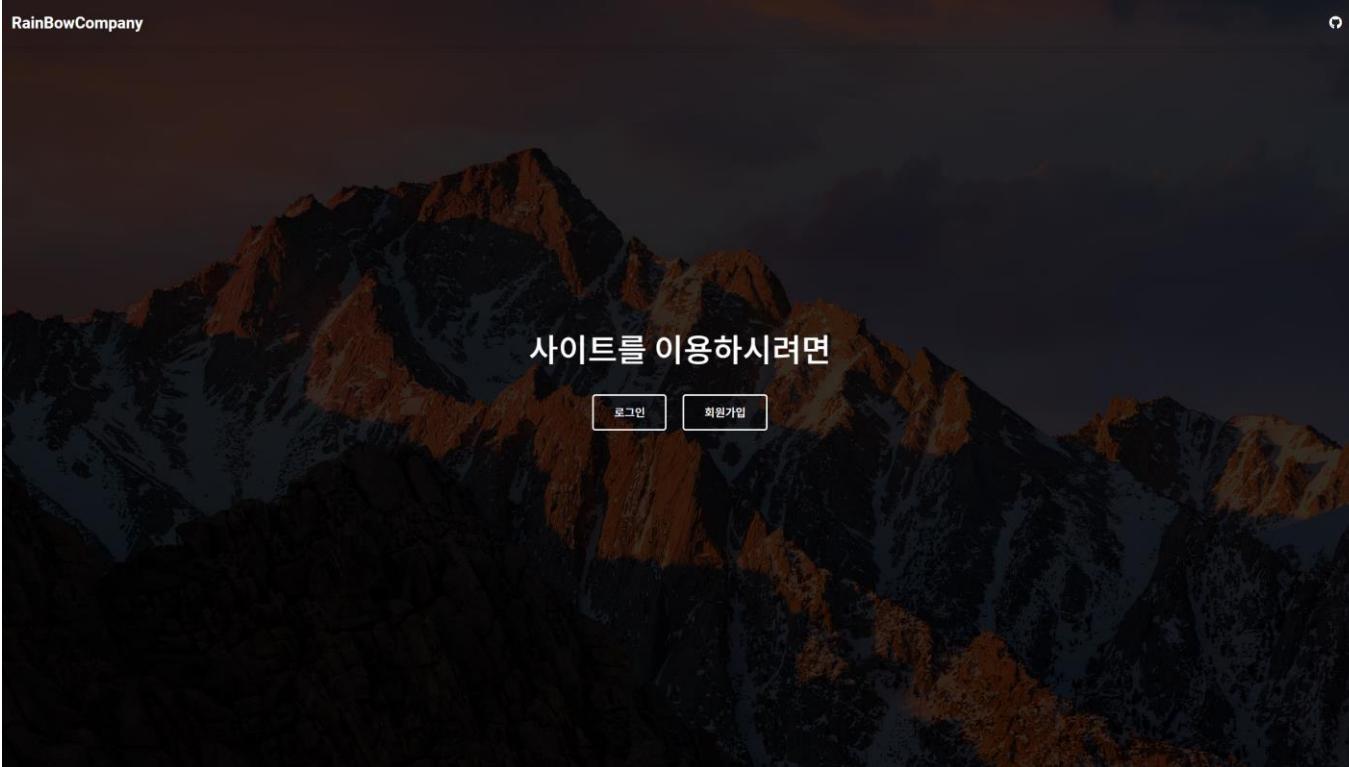
[Redis 실시간 연결 코드]

```
CONNECT stomp.min.js:8
2667.07
<<< MESSAGE stomp.min.js:8
destination:/stock/exchange
sd
content-
type:text/plain;charset=UTF-
8
subscription:sub-2
message-id:pndve4pi-1142
content-length:6
1290.0
<<< MESSAGE stomp.min.js:8
destination:/stock/exchange
my
content-
type:text/plain;charset=UTF-
8
subscription:sub-3
message-id:pndve4pi-1143
content-length:5
179.7
>
```

[Web에서 개발자 도구를 통한 Redis / Websocket 연결 화면]

```
Opening Web stomp.min.js:8
Socket...
Web Socket stomp.min.js:8
Opened...
>>> CONNECT stomp.min.js:8
accept-version:1.1,1.0
heart-beat:10000,10000
<<< stomp.min.js:8
CONNECTED
version:1.1
heart-beat:0,0
```

Part4 웹 구축 - 홈페이지 시연



김서영

김현수

라수정

오정화

전원석

전찬

형수진

Part4 웹 구축 - 홈페이지 예시

The screenshot displays a web-based financial trading platform with the following sections:

- Header:** RAINBOW COMPANY logo, 오후 01:47:23 유저님 환영합니다, user icon.
- Sidebar:** 코스피, 환율, 종목 차트, 계좌정보 (highlighted), AI 추천 투자 차트, 주요 종목 뉴스, 소개.
- Top Content Area:** **계좌정보 및 사용자 정보** table:

사용자 정보	계좌번호	잔고평가금액	투자손익금액	예수금	D+1예수금	D+2예수금	순익률
유저 user	555018903-01	76,478,700원	1,995,177원	452,459,815원	426,657,258원	425,628,646원	2.67%
- Middle Content Area:** **실시간 매매 정보** table:

주문시간	체결시간	주문번호	종목명	체결가	체결수량	주문유형
없음	없음	없음	없음	없음	없음	없음

검색: 10 개씩 보기 1 - 1 (총 1 건) 이전 1 다음
- Bottom Content Area:** **매매내역** table:

주문시간	체결시간	주문번호	종목명	체결가	체결수량	주문유형
2023-07-27 13:51:33	2023-07-27 13:51:37	13605	신세계	191500.00	1	(KSE)현금매도
2023-07-27 13:58:08	2023-07-27 13:58:13	13769	신세계	191100.00	1	(KSE)현금매수
2023-07-27 14:08:45	2023-07-27 14:09:13	14024	신세계푸드	39300.00	1	(KSE)현금매수
2023-07-27 14:09:59	2023-07-27 14:16:07	14068	신세계푸드	39250.00	1	(KSE)현금매수
2023-07-27 14:13:51	2023-07-27 14:16:07	14161	신세계푸드	39250.00	1	(KSE)현금매수

Part 5

개선사항 및 결론

Room for improvement & Conclusion



Part5 개선사항 및 결론

001 >> 시간 부족

프로젝트에 주제 선정에서 많은 시간을 소모하여 수집 및 개발 시간이 부족.
증시 거래시간 제약 조건으로 인한 코드 검증 시간 부족 .

002 >> 개념 학습

강화학습 개념에 대한 이해시간이 부족하여 개발하는데 어려움 겪음.
이러한 경험을 토대로 공부와 개발을 병행하여 완성도 있는 프로그램 개발을 위한 노력을 하고자 함.

003 >> 부가 기능 구현

고객의 편의성을 고려한 부가적인 기능(* 수익률 실시간 알림 기능)을 시간 부족으로 구현하지 못함 .
향후엔 모바일 애플리케이션을 기획하고 개발하여 고객의 편의성을 개선하고자 함 .



RAINBOW COMPANY
infinite of possibilities