AGILE 2

Neurotrade

2025.02.03

발표자 : 박현준

Contents

AGILE 2

타임라인

2/17/2025 2/21/2025 2/28/2025 3/10/2025 3/17/2025 3/17/2025 3/20/2025

단위 프로젝트

강사님과 회의 **금욜 1** **금욜 2** **에자일 2**

2.5주 (여러분이 일 할 수 있는 시간) 1주

단위 프로젝트

여러분이 한 것 종합

2차 에자일까지 3.5주 남음

AGILE 2

방향

중간회의 (여러분과 같이 앞에서 한 것) + 제가 따로 계속 강사님과 소통한 부분 정리

자동매매 구현에 있어서의 기술적 한계 존재

a. 유저 업비트 API를 서비스가 DB에 가지고 있어도 되는가?

i. 강사님: 웹은 백테스팅만 하고 유저들한테는 .exe로 다운로드 할 수 있는 프로그램을 줘라

ii. 나: 아니 그렇게 하려 했는데 웹 하라면서요

iii. 강사님: 그럼 가지고 있어 괜찮아!

iv. 나: ???????

b. 자동매매 구현이 얼마나 가능한가?

i. 나: 돌리려면 2중 3중 루프로 돌아가는데 (느린데), 멀티쓰레딩 해야함? @shared_task이런걸로 Redis, Celery (멀티쓰레딩 프레임워크) 써야함?

ii. 강사님: ㅇㅇ 맞음 방법 있음 그런서 쓰는 거 전공자는 다 하잖아요?

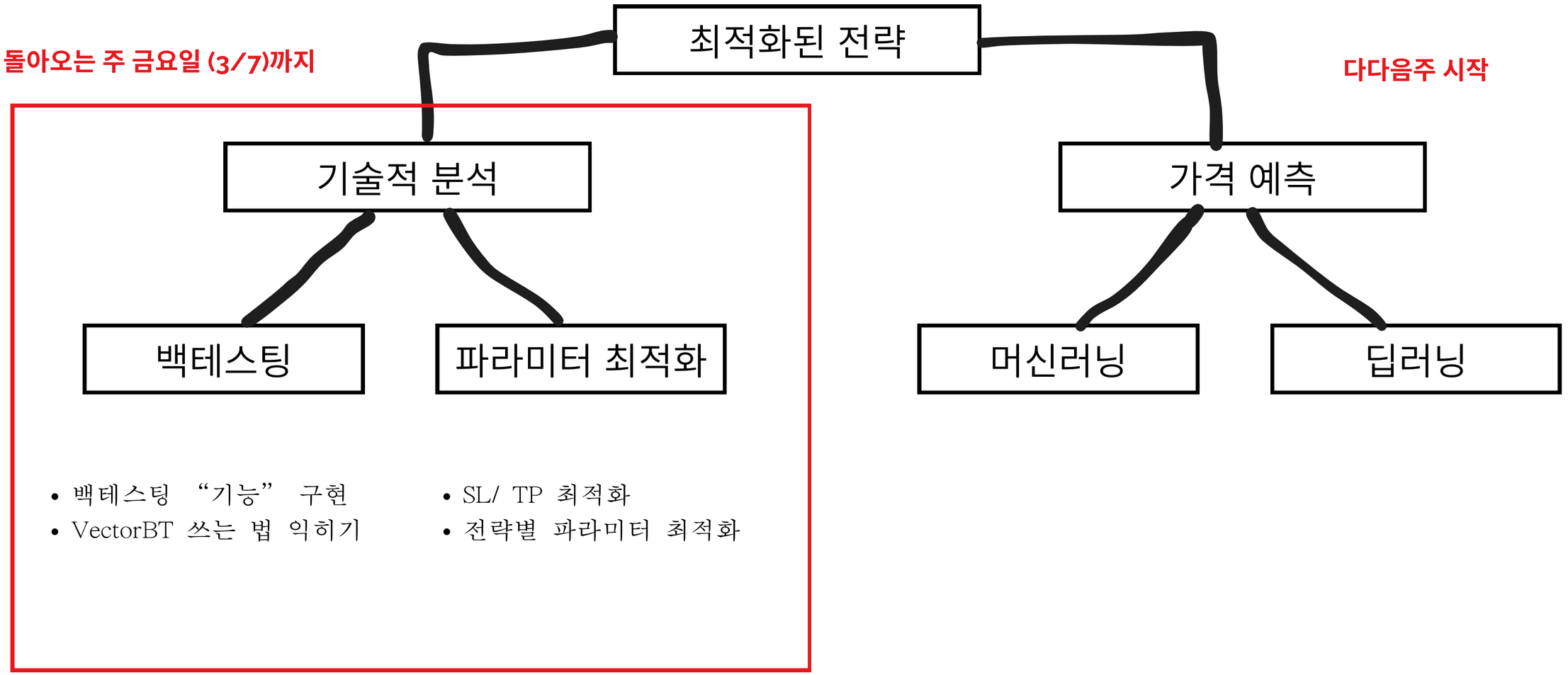
iii. 나: 아닌데용 ㅠ

iv. 강사님: 유감

강사님과 내린 결론: 2차 에자일에는 백테스팅 및 AI 모델에 집중

3차에 봐서 자동매매 구현(할거임) 및 기타 수정

백테스팅 및 AI 모델 구현 로드맵



백테스팅 구현 및 머신러닝/딥러닝 적용

기술적 분석의 최적화와 시계열 모델 머신러닝/딥러닝은 다르다!

기술적 분석 최적화

예시:

RSI전략

기본 세팅

SL/TP 설정
1. 10프로 손해보면 팔겠음
2. 30프로 이익보면 팔겠음

전략 설정
1. RSI 가 30이하로 내려가면 과매도로 보고 매수
2. RSI 가 70이상으로 올라가면 과매수로 보고 매도

최적화:
1. 몇프로 업다운하면 팔아야 최대수익?
2. 30/70 일때 수익율이 제일 좋음? 29/71은? -> RSI가 몇일때 사고 팔아야 최대수익?

1초요약: “최대수익 나오는 법” 예측

머신러닝/딥러닝 시계열 모델

머신러닝 (Sliding Window/ Walk-Forward 방법론)
1. 1월 - 3월 “가격" 으로 4월 예측 후 모델 학습
2. 2월 - 4월 “가격” 으로 5월 예측 후 모델 학습
3. 3월 - 5월 “가격” 으로 6월 예측 후 모델 학습
4. 무한 반복 (데이터 끝나는 곳 까지)
5. 이러면 window size는 2달, 예측범위는 1달

feature: 시가-고가-저가-종가 차트 (OHLC), 거래량(Volume)
데이터 학습끝난곳이 9월 마지막주라면 10월 한달 예측

딥러닝 LSTM (똑같이 window쓰는데 딥러닝 버전)

LSTM 간단 설명 링크

1초요약: 미래 “가격” 예측

백테스팅 구현 및 머신러닝/딥러닝 적용

기술적 분석의 최적화와 시계열 모델 머신러닝/딥러닝은 다르다!

기술적 분석 최적화

예시:

RSI전략

기본 세팅

SL/TP 설정
1. 10프로 손해보면 팔겠음
2. 30프로 이익보면 팔겠음

전략 설정
1. RSI 가 30이하로 내려가면 과매도로 보고 매수
2. RSI 가 70이상으로 올라가면 과매수로 보고 매도

최적화:
1. 몇프로 업다운하면 팔아야 최대수익?
2. 30/70 일때 수익율이 제일 좋음? 29/71은? -> RSI가 몇일때 사고 팔아야 최대수익?

다음주 금요일까지 할 것 (쉬움)

머신러닝/딥러닝 시계열 모델

머신러닝 (Sliding Window/ Walk-Forward 방법론)
1. 1월 - 3월 "가격" 으로 4월 예측 후 모델 학습
2. 2월 - 4월 "가격" 으로 5월 예측 후 모델 학습
3. 3월 - 5월 "가격" 으로 6월 예측 후 모델 학습
4. 무한 반복 (데이터 끝나는 곳 까지)
5. 이러면 window size는 2달 예측범위는 1달

feature: 시가-고가-저가-종가 차트 (OHLC), 거래량(Volume)
데이터 학습끝난곳이 9월 마지막주라면 10월 한달 예측

딥러닝 LSTM (똑같이 window쓰는데 딥러닝 버전)

LSTM 간단 설명 링크

****여러분 의견 필요 추후 회의**
다다음주 시작 (중간)

FAQ

Q: 기술적 분석이랑 가격 예측이랑 뭐가 다른지 모르겠어요

A: 기술적 분석은 차트를 기반으로 여러가지 신호 (거래량/이동평균선/상대강도지수 등등)들을 활용하여 언제 사고 팔아야 하는지 확인
가격예측은 가격 캔들 (시가 고가 종가 등)을 분석하여 미래에 이 가격이 될 것이다를 예측
근본적으로 기술적 분석은 언제 사고 팔아야할지 알려줄 수 있으나 미래 가격은 알려주지 못함 vs 머신러닝 모델들로 가격 예측 가능

Q: 지금 하는 건 기술적 분석중에서도 "지표분석" 인데, 캔들 패턴 분석이나 저항/지지선은 안하시나요?

A: 여쭤보신다면 기술적분석에 어느 정도 이해하신다는 가정하에 말씀드리겠습니다. 말씀주신 다른 기술적 분석 기술들은 1. 유저에게 파라미터로 받기 어렵거나, 2. 말씀주신대로 "패턴" 이기 때문에 LSTM 같은 딥러닝 시계열 모델에 더 어울립니다. 지지저항선을 유저에게 받기 애매하고, "정석 패턴" (헤드엔 숄더, 이중 천장/바닥, 깃발, 쐐기형, 삼각수렴, 기영이 머리 등) 또는 "정석 캔들" (양봉 3개 이후 장대양봉, 음양봉 사이 십자가 패턴 등)은 아직 기술적, 개념적으로 많이 부족하다고 생각하여 가장 숫자로 환산하기 쉬운 지표 분석을 채택하게 되었습니다.

Q: 기술적 분석 파라미터 최적화도 머신러닝으로 할 수 있지 않나요?

A: 네. 기술적 분석에 들어가는 파라미터 (언제사? 언제 팔아? 이동평균선이 언제일때 사? 상대강도가 언제일때 사?) 등등의 최적화도 머신러닝/ 딥러닝으로 할 수 있습니다. VectorBT에는 라이브러리 함수로 이런 최적의 파라미터 (머신러닝에서의 하이퍼 파라미너 튜닝 + 그리드 서치/ 랜덤서치) 를 찾아주는게 이미 구현이 되어 있어 편하게 모델을 학습시키시지 않고 최적화가 가능합니다. 본인의 인생 난이도가 아직 쉬움이시라면 머신러닝/딥러닝으로 구현하셔도 좋습니다. VectorBT에 있는 최적화 함수들은 Brute Force / 랜덤 서치로 구현되어 있으며 Walk-Forward로 최적화 하셔도 무방합니다.

Q: 기술적 분석이랑 머신러닝/딥러닝 돌린 것이랑 어떻게 합치나요?

A: 기본적으로 기술적분석에서 신호를 보내고, 머신러닝/딥러닝 돌린 모델도 같은 방향을 시사한다면 거래가 체결되는 방향으로 생각하고 있습니다. 예시로, 상대강도지수가 30이하로 떨어져 과매도 상태를 시사한다면 -> 지금 사야한다면, 머신러닝/딥러닝도 가격 상승을 바라보는지 예측해보고 같은 방향이라면 삽니다. 반대방향이라면? 비율로 누가 더 강한 신호인지: 세팅을 기술적 분석 5, 가격 5로 해놓았다면 거래가 체결되지 않고, 6:4로 해놓았다면 강한 쪽의 결정대로. 기술적 분석에 힘을 실어줄 것이냐 가격예측을 더 중요시 할 것이냐 역시 머신러닝/딥러닝으로 최적의 비율을 찾는 (기술적 분석 6.29 대 가격 3.71 비율로 했을때 수익률이 최대이더라) 일이 될 수 있습니다.

이 부분은 복잡하고 주관적이어서 이 방향이 맞는지/ 동의하시는지/ 틀린지 여러분의 의견이 필요하기에 추후에 회의를 해야할 것 같습니다.
현재 방향은 저희의 진행속도, 코딩 지식과 능력, 관심사를 종합하여 "적당히 임의로" 제가 결정했음을 알립니다.

AGILE 2

해야할 것

2/17/2025 2/21/2025 2/28/2025 3/10/2025 3/17/2025 3/17/2025 3/20/2025

단위 프로젝트

돌아오는 월욜 **강사님과 회의** **금욜 1** **금욜 2** **에자일 2**

3주차

- VetorBT 로 전략 구현 및 파라미터 최적화

4-5주차

- 모델 구현 및 기능 종합

AGILE 2

VECTORBT 소개

VECTORBT 소개

- TA 라이브러리 / Backtesting 라이브러리처럼 전략 구현 및 과거 데이터로 백테스팅 가능
- 백테스팅 결과 json형식으로 리턴 가능
- 백테스팅 결과 시각화 가능
- 백테스팅에 사용한 전략 기반으로 실제 거래 구현 가능

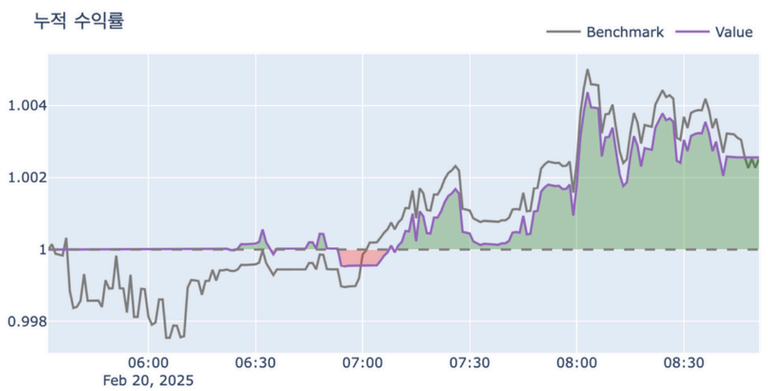
```python
# ✅ 백테스트 실행 함수
def run_backtest(market="KRW-BTC", short_window=50, long_window=100, init_cash=1000, unit=15, total_count=100000):
    df = fetch_upbit_data(market, unit, total_count)
    if df.empty:
        print("⚠️ 데이터가 없습니다. API 응답을 확인하세요.")
        return

    price = df["close"]

    # ✅ 이동평균선 기반 전략 적용
    fast_ma = vbt.MA.run(price, short_window).ma
    slow_ma = vbt.MA.run(price, long_window).ma

    entries = fast_ma > slow_ma  # 매수 시그널
    exits = fast_ma < slow_ma  # 매도 시그널

    # ✅ 포트폴리오 백테스트 실행
    pf = vbt.Portfolio.from_signals(price, entries, exits, init_cash=init_cash)
    print(pf.stats())
    # ✅ 포트폴리오 성과 지표 계산
    stats = pf.stats()

    print("\n📊 백테스트 결과 📊")

    # ✅ 시각화 (vectorbt 내장 함수 활용)
    pf.plot_value(title="포트폴리오 가치 변화").show()
    pf.plot_drawdowns(title="드로우다운 분석").show()
    pf.plot_trades(title="거래 내역").show()
    pf.plot_cum_returns(title="누적 수익률").show()
    pf.plot(title="plot").show()
    # pf.plot_candlestick(title="plot2").show()
    pf.plot_trades(title="plot3").show()

# ✅ 백테스트 실행
run_backtest(
    market="KRW-BTC",
    short_window=10,
    long_window=50,
    init_cash=1000,
    unit=15,
    total_count=200
)
```

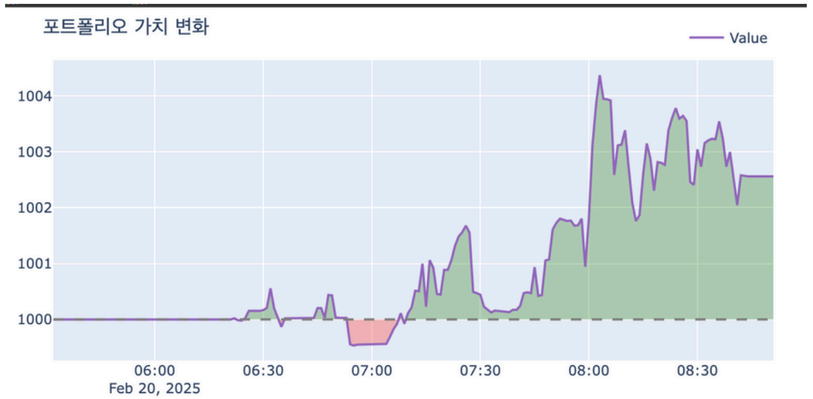
백테스팅 예시 (이동평균선 50일선/ 100일선 교차 전략)

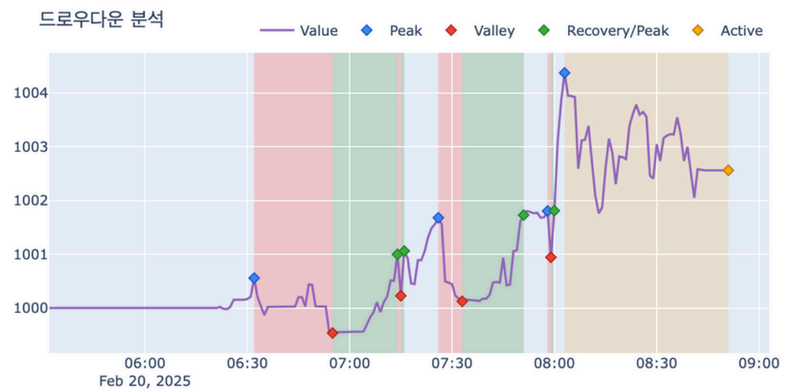
자료제공: 편성민님

Start	2025-02-20 05:32:00
End	2025-02-20 08:51:00
Period	0 days 03:20:00
Start Value	1000.0
End Value	1002.561166
Total Return [%]	0.256117
Benchmark Return [%]	0.251471
Max Gross Exposure [%]	100.0
Total Fees Paid	0.0
Max Drawdown [%]	0.259893
Max Drawdown Duration	0 days 00:48:00
Total Trades	2
Total Closed Trades	2
Total Open Trades	0
Open Trade PnL	0.0
Win Rate [%]	50.0
Best Trade [%]	0.300038
Worst Trade [%]	-0.04379
Avg Winning Trade [%]	0.300038
Avg Losing Trade [%]	-0.04379
Avg Winning Trade Duration	0 days 01:40:00
Avg Losing Trade Duration	0 days 00:37:00
Profit Factor	6.848808
Expectancy	1.280583
Sharpe Ratio	30.854727
Calmar Ratio	319205.581098
Omega Ratio	1.179519
Sortino Ratio	43.675179

dtype: object



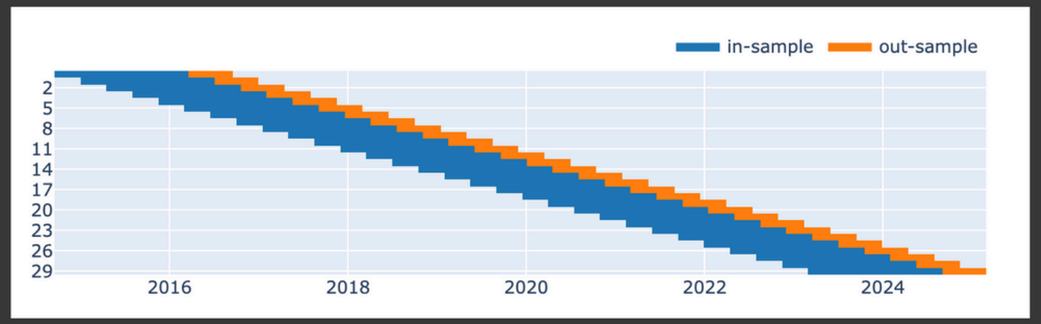


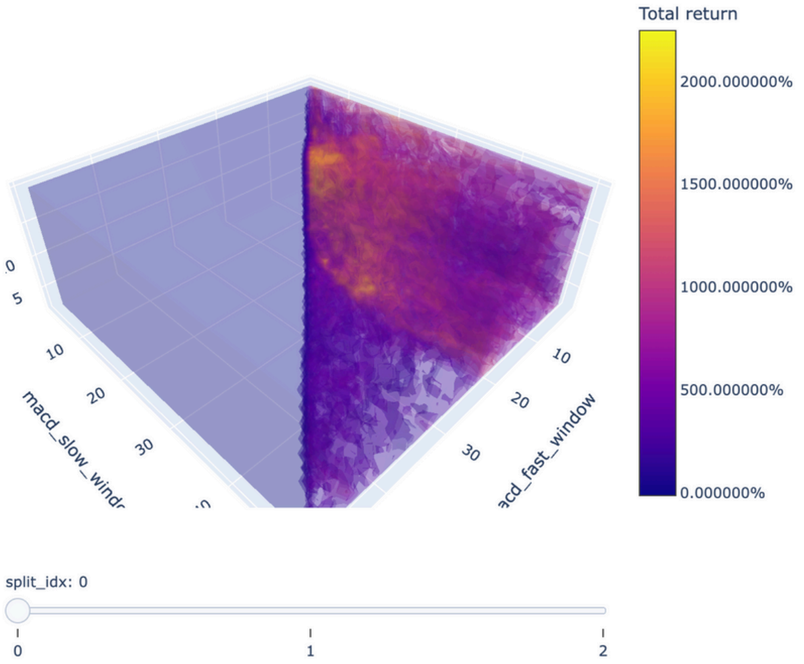


백테스팅 결과

자료제공: 편성민님

```
def roll_in_and_out_samples(price, **kwargs):
    return price.vbt.rolling_split(**kwargs)
roll_in_and_out_samples(price, **split_kwargs, plot=True, trace_names=['in-sample', 'out-sample']).show()
```





백테스팅 최적화 (머신러닝)
(**전략 최적화 아님)

롤링윈도우/ 히트맵 등 이미 라이브러리 안에 함수로 있음

자료제공: 편성민님

참조링크:

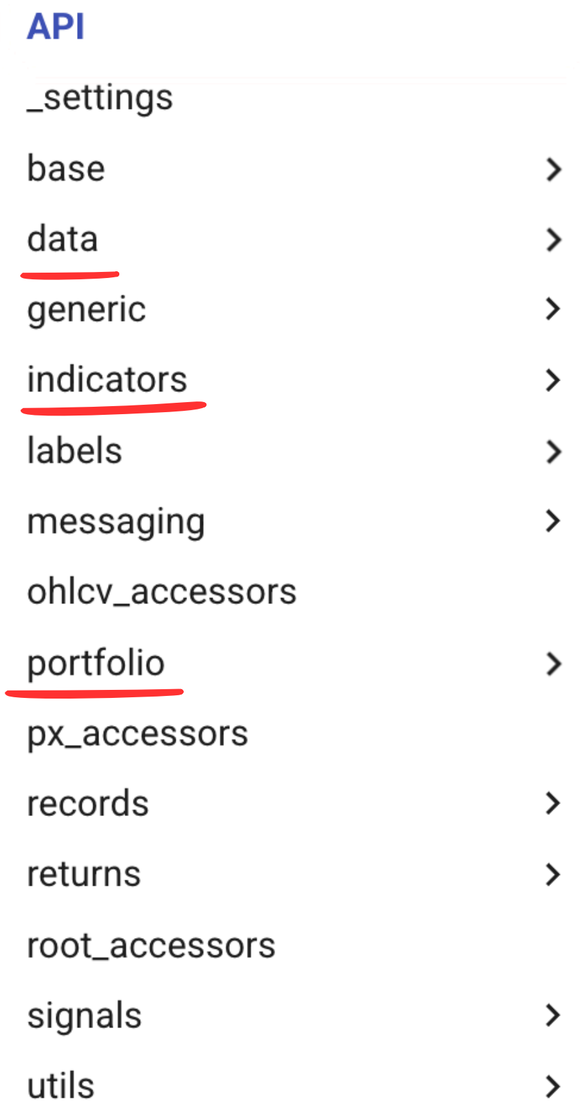
기본 사용법

이동평균선 백테스팅 해보기

SL/TP 백테스팅 해보기

Walk-Forward 최적화

모든 라이브러리 함수 문서



눈여겨볼섹션

문서가 아주 깁니다. 절대 다 읽지 마시고,

1. 컨셉만 이해하고 챗지피티/커서/코파일럿한테 물어본다.
2. 안되거나 이상한 함수를 쓰면 (이상한 문법을 사용하면)저 웹사이트를 참조하게 한다
3. 그래도 안되면 안되는 곳만 읽어본다

번외: 난 정말 궁금하다 하시면 다 읽으셔도 됩니다.

AGILE 2

역할분담

개발 개요

할 것 1:백테스팅 3시간 (벡터비티 이해 + 함수 구현)

1. 모든 함수(Django)는 자신의 파라미터를 프론트엔드에서 받는다 (request.data.backtesting_data)
    a. 저한테 무슨 파라미터가 필요한지 알려줘야함 (프론트엔드)
2. VectorBT를 통해 프론트엔드에서 받은 파라미터로 백테스팅을 돌리고 결과를 반환한다

할 것 2: 최적화 5시간 (최적화 방법 생각 + 구현)

1. 만든 백테스팅 함수를 기반으로 최적의 파라미터를 찾는다.

개발환경

지금 당장 프론트가 없기 때문에 코랩에서 하셔도 좋고 (추천) 파이썬에서 하셔도 좋고, 능력이 되신다면 장고에서 바로 하셔도 됩니다. 하지만 결국 장고에서 돌아가야한다는 사실을 유념하시고 개발부탁드립니다.

예시: 이동평균선 교차 전략 백테스팅

```python
@api_view(['POST'])
def perform_backtesting_MovingAverage(request):
    try:
        #request에서 프론트엔드에서 받은 파라미터 초기화 -> 개발할대는 request를 하드코딩해서 되는지 확인
        data = request.data
        symbol = data.get('symbol', 'BTC-USD')
        start_date = datetime.strptime(data['start_date'], '%Y-%m-%d')
        end_date = datetime.strptime(data['end_date'], '%Y-%m-%d')
        take_profit = float(data['take_profit']) / 100
        stop_loss = float(data['stop_loss']) / 100
        short_ema = int(data['short_ema'])
        long_ema = int(data['long_ema'])

        # 역사적 데이터 로드 (여러분께 맡김-> 어차피 테스트용이고 csv파일 다운받아셔 하셔도 좋고 편성민님께서 만드신 업비트에서 데이터 가져오는 함수 그대로 쓰셔도 좋음 (django에서 backtesting/views.py 참조))
        df = yf.download(symbol, start=start_date, end=end_date)
        close = df['Close']

        # 장/단기 이동평균선 계산
        short_ema = vbt.MA.run(close, window=short_ema, short_name='short')
        long_ema = vbt.MA.run(close, window=long_ema, short_name='long')

        # 이동평균선으로 매매전략 구현
        entries = short_ema.ma_crossed_above(long_ema) #단기 이동평균선이 장기이동평균선 상향돌파시 무조건 사기
        exits = short_ema.ma_crossed_below(long_ema) #단기 이동평균선이 장기이동평균선 하향돌파시 무조건 팔기

        # vectorbt에 전략 먹이기
        pf = vbt.Portfolio.from_signals(
            close,
            entries,
            exits,
            tp_stop=take_profit,
            sl_stop=stop_loss,
            freq='1D',
            init_cash=10000,
            fees=0.001
        )

        # 전략 성과
        stats_df = pf.stats() # 전략 성과 파라미터 (수익률, 드로다운, 자산변동 등)
        pf.plot().show() # 전략 성과 시각화
```

기본 파라미터

이건 전략마다 다름. 나한테 줘야함

본인의 전략이 뭐하는 전략인지 모르면 각자 알아보기 바람
-> 전략 안에서 어떻게 할지는 알아서

예시: 이동평균선 교차 전략 최적화

```python
@api_view(['POST'])
def optimized_MovingAverage(request):
    try:
        #파라미터 받은 것 초기화
        data = request.data
        symbol = data.get('symbol', 'BTC-USD')
        start_date = datetime.strptime(data['start_date'], '%Y-%m-%d')
        end_date = datetime.strptime(data['end_date'], '%Y-%m-%d')
        take_profit = float(data['take_profit']) / 100
        stop_loss = float(data['stop_loss']) / 100

        # 역사적 데이터 로드
        df = yf.download(symbol, start=start_date, end=end_date)
        close = df['Close']

        # 그리드서치 할 범위 정의
        short_windows = list(range(5, 51, 5))
        long_windows = list(range(10, 201, 10))

        # 그리드서치로 모든 조합 확인
        ma = vbt.MA.run_combs(
            close,
            window=short_windows,
            window2=long_windows,
            short_names=['short', 'long'],
            param_product=True  # Creates all combinations of parameters
        )

        # 전략 정의
        entries = ma.short_ma.crossed_above(ma.long_ma)
        exits = ma.short_ma.crossed_below(ma.long_ma)

        # 정의한 전략으로 모든 백테스팅 실행
        pf = vbt.Portfolio.from_signals(
            close,
            entries,
            exits,
            tp_stop=take_profit,
            sl_stop=stop_loss,
            freq='1D',
            init_cash=10000,
            fees=0.001
        )

        # 한개 조합의 수익률 / 샤프지수 등 계산
        metrics = pd.DataFrame({
            'total_return': pf.total_return(),
            'sharpe_ratio': pf.sharpe_ratio(),
            'max_drawdown': pf.max_drawdown(),
            'win_rate': pf.win_rate()
        })

        # 조합 저장
        metrics['short_window'] = [p[0] for p in ma.window_idx]
        metrics['long_window'] = [p[1] for p in ma.window2_idx]

        # 샤프지수로 오름차순 정렬로 가장 좋은 파라미터 찾기
        metrics = metrics.sort_values('sharpe_ratio', ascending=False)
        best_params = metrics.iloc[0]

        # 가장 좋은 파라미터로 백테스팅 실행
        best_ma = vbt.MA.run(
            close,
            window=[int(best_params['short_window']), int(best_params['long_window'])],
            short_names=['short', 'long']
        )

        best_entries = best_ma.short_ma.crossed_above(best_ma.long_ma)
        best_exits = best_ma.short_ma.crossed_below(best_ma.long_ma)

        best_pf = vbt.Portfolio.from_signals(
            close,
            best_entries,
            best_exits,
            tp_stop=take_profit,
            sl_stop=stop_loss,
            freq='1D',
            init_cash=10000,
            fees=0.001
        )

        #성과 확인
        best_pf.stats()
        best_pf.plot()
```

결과보고

1. 백테스팅 방법과 최적화 방법은 너무나 많습니다. 어떻게 구현하셨는지(변수를 뭐 사용하셨는지, 루프를 어떻게 돌리셨는지), 어떤 방법론 (walk-forward, grid search, random-search, 기타)을 쓰셨는지 묻지 않겠습니다.
2. 단 백테스팅 기본 파라미터 (백테스팅 범위, st/tp 등) 은 쟝고에서 돌릴 것을 유념해 개발해주세요. 백테스팅 결과는 무조건 portfolio.stats() 랑 portfolio.plot()입니다.
3. 기본 백테스팅 파라미터 아무거나넣어서 결과 (stats, plot 등) 와 파라미터 최적화된 백테스팅 결과랑 비교하여 파라미터 최적화된게 더 낫다는 것을 보여만 주시면 돌아오는 주 끝나십니다.

샤프지수 1.5이상, 수익률 30프로 이상을 “목표”로 최적화해주세요.

편성민: 이동평균선 교차

조현정: 상대강도지수 (RSI)

원유형, 전서빈: 볼린져밴드

가장 유명한 전략들로 제가 임의로 정했습니다.

까지가 AGILE 2 3주차

4,5,6주차까지 있음

단위프로젝트랑 겹쳐서 바쁘신 것 압니다만 잘 부탁드립니다.

NEUROTRADE

팀장 박현준