비트코인 트레이딩 알고리즘

201735902 고승표

목차

₿ 프로젝트 주제 선정 이유

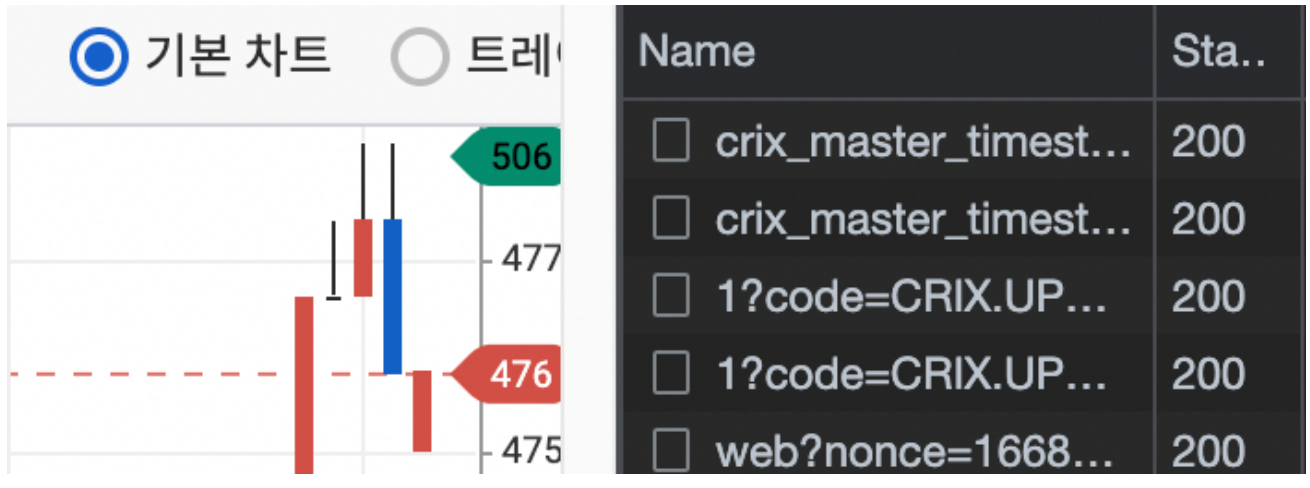
기본적인 지표와 데이터를 활용해
나만의 알고리즘을 개발 할 수 있고
현실에 접목해 쉽게 체감 할 수 있다.

프로젝트 목표

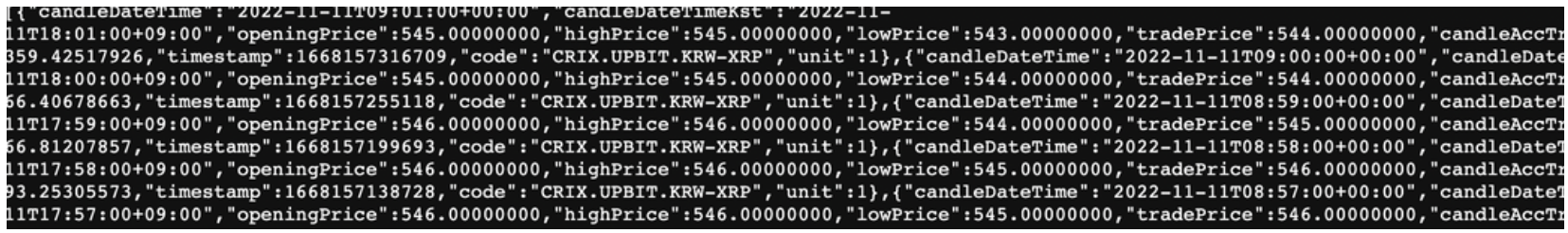

주문
호가
차트
시세
정보
주문
6,350
틱
5분
일
주
월
단순 MA 5 20 30 60 120
10,000
8,800
7,575
5,150
2,725
314
15일 16:30 (KST)
거래량 단순 MA 5 10 20
64,306.105
42,870.726
21,435.347
5,573.006
97.62
RSI 14
70
78.22
58.81
30
39.41

₿ 학습 데이터

- 시가
- 종가
- 저가
- 고가
- 거래량



최초 데이터



수집 대상 지정

```
coin_name = "KRW-XRP"
start_time = "2022-11-13T00:00:00" #오늘날짜
base_url = "https://crix-api-cdn.upbit.com/v1/crix/candles/minutes/1?code=CRIX.UPBIT.{} \
           &count=400&to={}.000Z"
cols = ['timestamp','openingPrice', 'highPrice', 'lowPrice', 'tradePrice','candleAccTradeVolume']
df_out = pd.DataFrame() #비어있는 데이터 프레임 생성
```

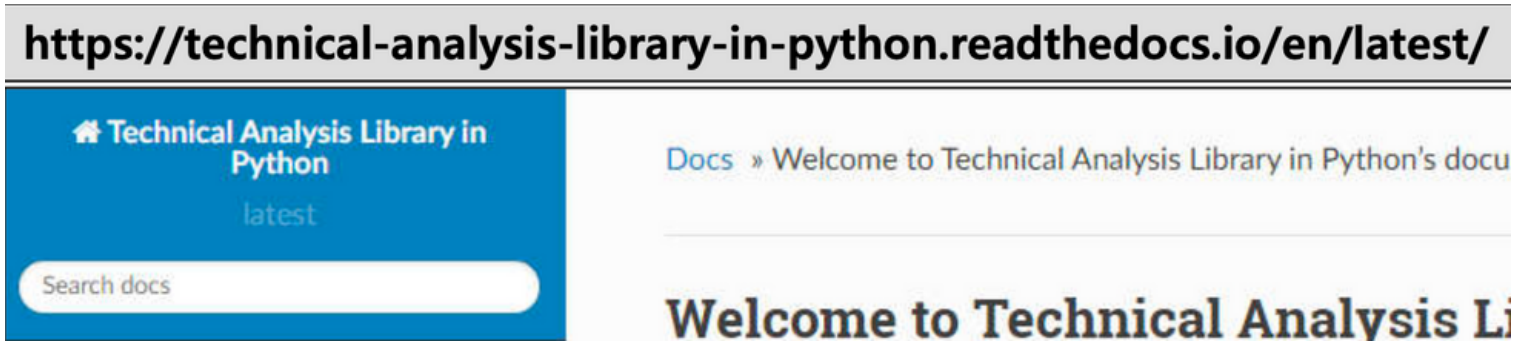
데이터 수집하기

```
for i in range(0, 500):
    url = base_url.format(coin_name, start_time)
    webpage = requests.get(url)
    df_temp = pd.read_json(webpage.content)

    df_temp_data = df_temp[cols]
    df_out = df_out.append(df_temp_data) #데이터 프레임에 데이터 추가

    temp_date = df_temp_data.tail(1)['timestamp'].dt.strftime('%Y-%m-%dT%H:%M:%S')
    start_time = temp_date.values[0]

    #사람같이 행동하기
    wait_time = random.choice([1.2, 1.4, 1.6, 1.8])
    time.sleep(wait_time)
#     print(start_time, end=', ')
    print(i, end=', ') #progessing

df_out.reset_index()
df_out.to_csv("./data/{}.csv".format(coin_name), index=False)
```

수집된 데이터 파일

	A	B	C	D	E	F	G
1	mestamp	openingPrice	highPrice	lowPrice	tradePrice	candleAccTradeVolume	
2	022-11-13 (	506	506	505	506	189184.691	
3	022-11-12 2	505	506	505	506	41075.1806	
4	#########	505	506	505	505	92763.9418	
5	#########	505	506	505	505	53007.4645	
6	#########	504	506	504	506	159032.746	
7	#########	505	506	504	505	62284.3067	
8	#########	506	506	504	505	176121.637	
9	#########	505	506	505	505	94998.6106	

수집된 데이터 가공



TA 라이브러리 사용 데이터 가공

↓

단순 이동평균선 (SMA), 지수 이동평균선 (EMA),
이동평균선 (MACD) 등

```
##### 지수 이동평균 계산

from ta.trend import EMAIndicator
df['ema7'] = EMAIndicator(df['c'], window=7).ema_indicator()
df['ema25'] = EMAIndicator(df['c'], window=25).ema_indicator()
df['ema99'] = EMAIndicator(df['c'], window=99).ema_indicator()
df.head(10)
```

	t	o	h	l	c	v	sma7	sma14	sı
199999	2022-06-27 02:59:58.157	475	475	474	475	20126.578879	NaN	NaN	
199998	2022-06-27 03:00:51.753	475	475	474	475	29957.209487	NaN	NaN	
199997	2022-06-27 03:01:58.524	475	475	473	473	444851.548350	NaN	NaN	

가공된 데이터 파일

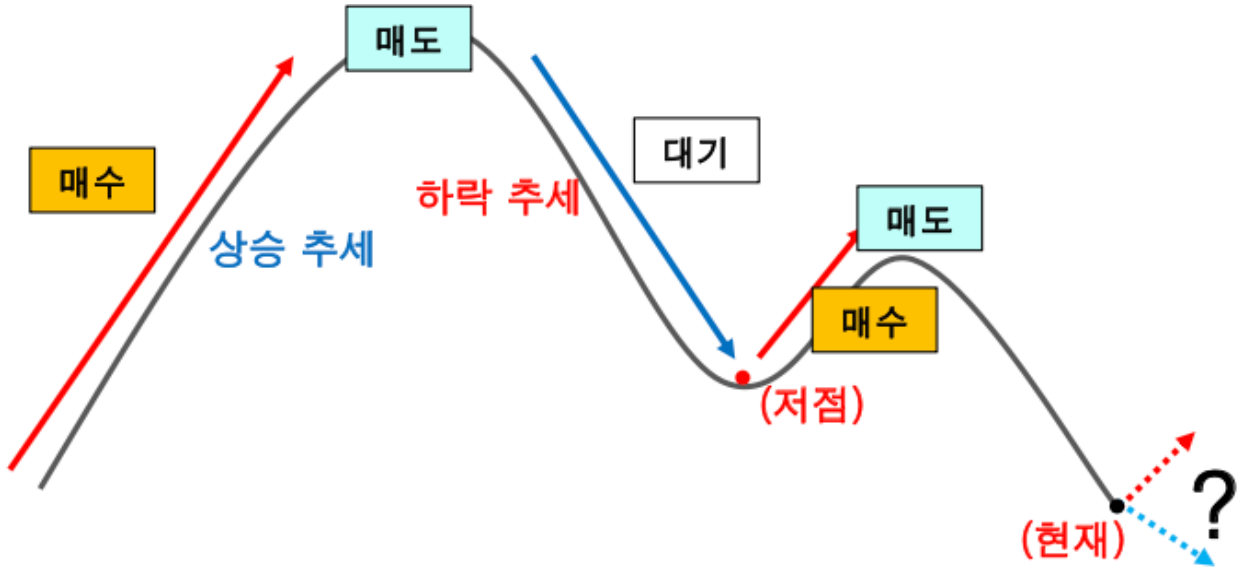
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
t	o	h	l	c	v	sma7	sma14	sma99	wma7	wma25	wma99	ema7	ema25	ema99	macd	macd_s	macd_d	rsi7
2021-04-01 (	705	705	700	700	2366999.77	704.571429	707.64	695.292929	704.071429	707.753846	698.364242	704.455179	705.100281	697.824277	0.67073043	2.25466021	-1.5839298	27.7991235
##########	700	700	698	699	1245075.18	703.857143	707.72	695.363636	702.678571	707.089231	698.438384	703.091384	704.631028	697.847792	0.09168903	1.82206597	-1.7303769	25.8045515
2021-04-01 (	699	700	693	694	1735347	702.142857	707.6	695.393939	700.214286	706.033846	698.411111	700.818538	703.813257	697.770836	-0.7618812	1.30527654	-2.0671577	18.1909419
2021-04-01 (	694	700	693	698	1843323.95	701.142857	707.68	695.474747	699.178571	705.295385	698.463232	700.113904	703.366083	697.775419	-1.102862	0.82364883	-1.9265109	35.855108
##########	698	700	697	700	1003725.21	700.285714	707.84	695.565657	698.892857	704.704615	698.553737	700.085428	703.107154	697.819911	-1.1978999	0.41933908	-1.617239	43.0305781
##########	699	701	699	700	1449010.12	699.285714	707.68	695.656566	698.821429	704.101538	698.642424	700.064071	702.868142	697.863513	-1.2587085	0.08372957	-1.342438	43.0305781

₿ 알고리즘 선택 및 튜닝

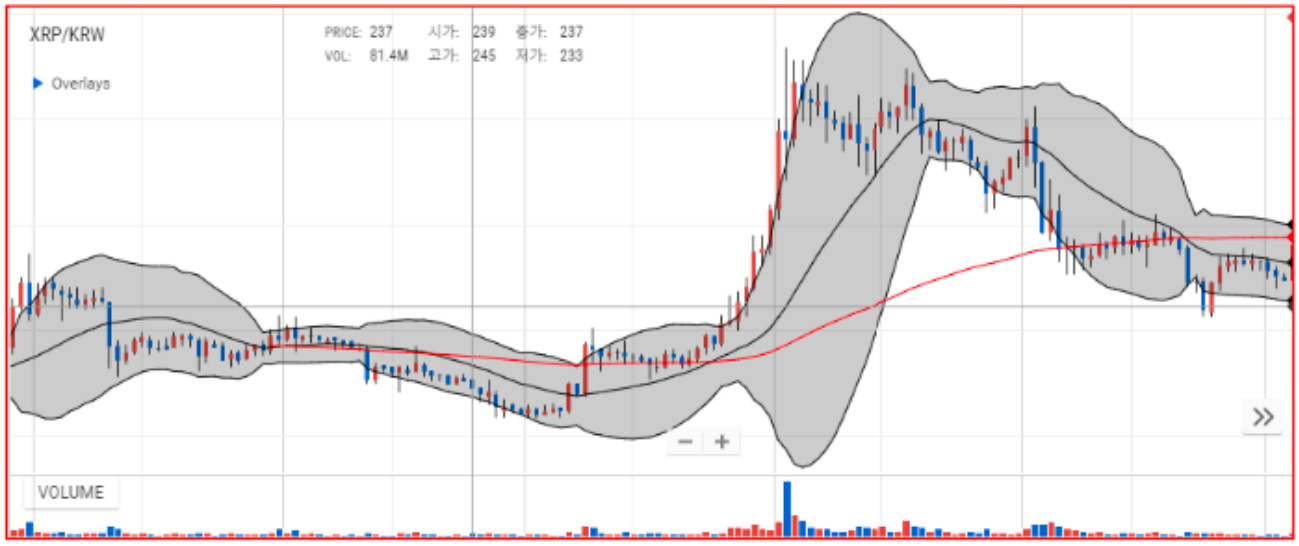
- 추세 추종 전략 알고리즘
- 역추세 전략 알고리즘
- 볼린저밴드를 활용한 알고리즘
- 추세 적응형 전략 알고리즘

추세 추종 전략

추세를 따라가는 매매법
상승 추세일 때 매수, 하락 추세로 전환시 매도
현재 추세가 전환점이 오지 않는 한 당분간 지속된다는 논리에 바탕
가격이 지속적으로 상승하는 상승장에서 좋은 수익률을 보여줌
중장기 투자에 적합



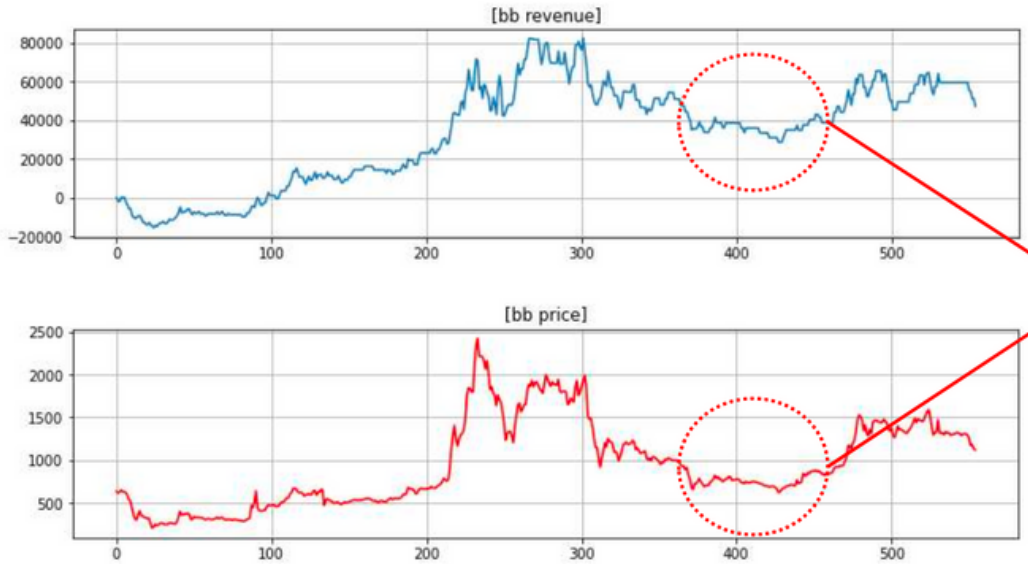
볼린저 밴드



상단, 중단, 하단 선(20일 단순 이동 평균선 +- 20일 표준 편차*2)을 이용해 밴드를 그리고 그 밴드 안에서 차트 움직임을 분석

상단선을 벗어날 경우 가격이 많이 오른 것으로 판단, 매도
하단선을 벗어날 경우 가격이 많이 내린 것으로 판단, 매수
상단선과 하단선 간격이 좁아질 경우 큰 가격 변동 예상, 관망

추세 적응형 알고리즘

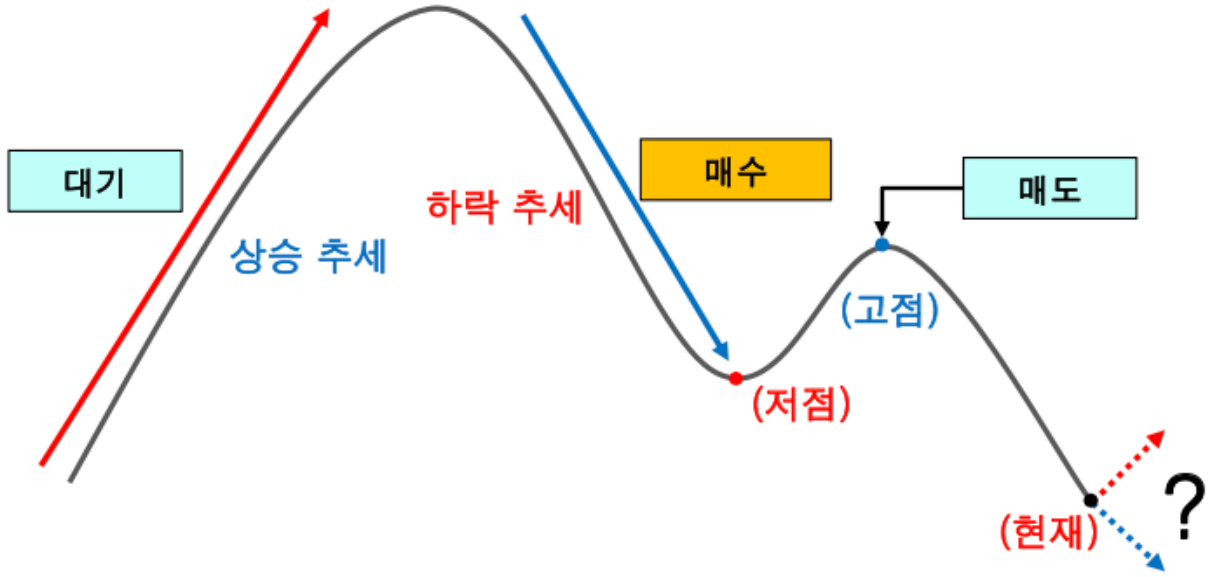


가격 하락 구간에서
손실을 최소화

볼린저밴드 알고리즘과 거의 같으나 장기 하락 추세일 때 손실을 줄이기 위해 매수 건수, 목표 수익률, 최대 손실률을 줄인다.

역추세 전략 알고리즘

역 추세 전략은 가격이 하락할 때 매수
가격이 일정 부분 상승하면 다시 매도
가격이 박스권에서 횡보하는 시장에서 수익률이 높음, 초단기 투자에 적합



역추세 전략 알고리즘 모의 투자

```
#단계별 구매 수량
def get_buy_amt_list(buy_amt_unit, buy_cnt_limit, increace_rate):
    buy_amt = 0
    buy_amt_list = [0.0]
    for idx in range(0, buy_cnt_limit):
        temp_amt = buy_amt_unit + buy_amt * increace_rate
        buy_amt = round(buy_amt + temp_amt, 4)
        buy_amt_list.append(buy_amt)
    return buy_amt_list

#손실 최소화 실현 금액 계산
def get_max_loss(close, buy_amt_unit, buy_cnt_limit, increace_rate, max_loss_rate):
    buy_amt = 0 #누적 구매 수량
    buy_price = 0 #누적 구매 금액
    for idx in range(0, buy_cnt_limit):
        temp_amt = buy_amt_unit + buy_amt * increace_rate
        buy_price = round(buy_price + close * temp_amt, 4)
        buy_amt = round(buy_amt + temp_amt, 4)
    return round(buy_price * max_loss_rate, 4)
```

```
revenue_rate = 0.014  #익절 비율
max_loss_rate = 0.2   #손절 비율: 20%
increace_rate = 0.2
buy_cnt_limit = 7     #최대 오픈 건수
buy_amt_unit = 20    #최소 오픈 수량

trade_fee = 0.001 #거래수수료
close = 480
buy_amt_list = get_buy_amt_list(buy_amt_unit, buy_cnt_limit, increace_rate)
max_loss = get_max_loss(close, buy_amt_unit, buy_cnt_limit, increace_rate, max_loss_rate)
buy_amt_list
```

```
[0.0, 20.0, 44.0, 72.8, 107.36, 148.832, 198.5984, 258.3181]
```

```
buy_cnt = 0
buy_price = 0 #누적 구매 금액
buy_amt = 0 #누적 구매 수량
revenue = 0 #전체 이익금
revenue_t = 0 #단계별 이익금
buy_cnt_tot = 0
# df = df_org.iloc[df_org.shape[0]-144000:,]
df = df_org.iloc[df_org.shape[0]-144000:df_org.shape[0]-124000,]

df = df_org.iloc[df_org.shape[0]-144000:,] #1일 1440분, 144000분 -> 100일
for i in range(0, df.shape[0]-1):
    close1 = round(df.iloc[i:i+1,]['c'].values[0],4)
    close2 = round(df.iloc[i+1:i+2,]['c'].values[0],4)
    wma7 = round(df.iloc[i:i+1,]['wma7'].values[0],4) #7분 가중이동 평균
    wma99 = round(df.iloc[i:i+1,]['wma99'].values[0],4) #99분 가중이동평균
    vwap = round(df.iloc[i:i+1,]['vwap'].values[0],4) # 거래량이동평균
```

1분전 종가, 현재 종가, 가중이동 평균, 거래이동평균 선언

```
    #stop loss
    loss = buy_price - close2*buy_amt
    if loss > max_loss:
        revenue_t = close2*buy_amt - buy_price - buy_price * trade_fee
        revenue = round(revenue + revenue_t,4)
        print("*[{}] stop loss:{} revenue:{} ".format(i, round(revenue_t,4), round(revenue,4)))
        buy_cnt = 0
        buy_amt = 0
        buy_price = 0
        continue
```

스탑 로스 설정
총구매가격 - 현재가치*구매량 > MAX_LOSS를
넘을 경우 손절

```
    #take profit
    tp_revenue = close2*buy_amt - (buy_price + buy_price*revenue_rate)
    if buy_cnt > 0 and tp_revenue > 0:
        revenue_t = close2*buy_amt - buy_price - buy_price * trade_fee
        revenue = round(revenue + revenue_t,4)
#         print("*[{}] take profit:{} revenue:{} ".format(i, revenue_t, revenue))
        buy_cnt = 0
        buy_amt = 0
        buy_price = 0
        continue
```

이익 실현 설정

종가*구매량 - (총구매가 + 목표 수익)

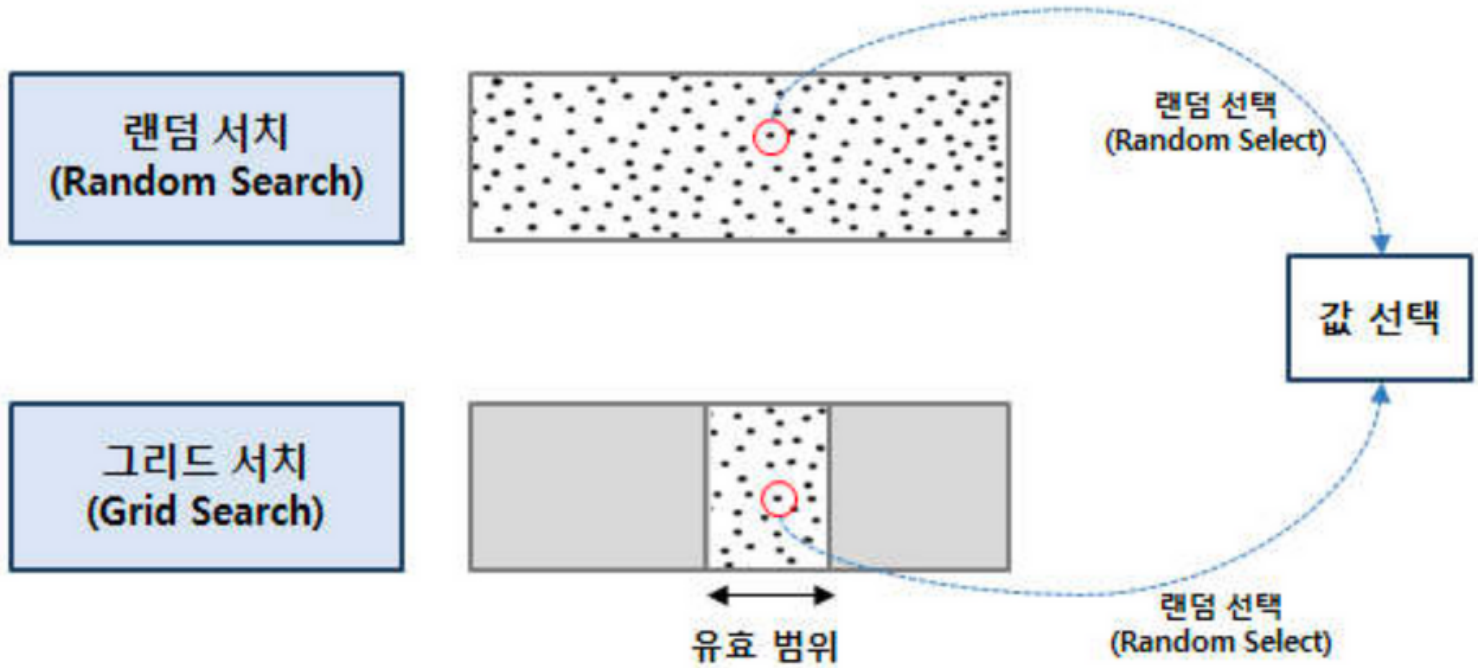
```
*[71227] stop loss:-21859.7842 revenue:34349.3408
*[75584] stop loss:-20936.3751 revenue:19891.7441
*[99391] stop loss:-21258.8006 revenue:17681.8855
*[110469] stop loss:-21562.2511 revenue:7329.1382
*[137019] stop loss:-20960.9051 revenue:1682.0156
*[137955] stop loss:-21453.6948 revenue:-12989.2228
*[139332] stop loss:-20983.1613 revenue:-26213.6166
```

약 -26000원 손절

알고리즘 개선 및 모의 투자 VER.2

그리드 서치

랜덤 서치	무작위로 값을 선택해서 대입해 보는 것
그리드 서치	기준선 즉 범위를 정해 놓고 그 범위 안에서 값을 선택하는 것



```python
def random_select():
    config_data = {
        'revenue_rate':rand.uniform(0.005,0.015),
        'max_loss_rate':rand.uniform(0.05,0.3),
        'increace_rate':rand.uniform(0.1,0.4),
        'buy_cnt_limit':rand.uniform(5,20),
        'buy_amt_unit':rand.uniform(15,40)
    }
    return config_data
```

이익률, 손실률, 구매 증가량, 구매 최대치, 1회 구매량을 범위를 정해 랜덤한 값으로 수익률을 계산해본다.

```python
for i in range(50):
    config_data = random_select()
    revenue = run_test(config_data)
    print("*config:{} result:{}".format(config_data, revenue))
    result = []
    result.append(config_data)
    result.append(revenue)
    results.append(result)
```

총 50번 랜덤한 값으로 계산해보았다.

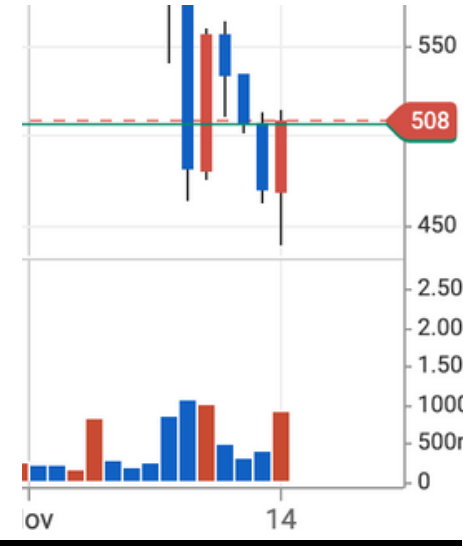
```
Out[4]: [[{'revenue_rate': 0.008884690841510733,
           'max_loss_rate': 0.09312280196131158,
           'increace_rate': 0.16316799099492513,
           'buy_cnt_limit': 18.886217544914498,
           'buy_amt_unit': 33.64605522724095},
          9190.865592032847],
```

위와 같은 비율로 설정한 경우 1차에서 -26000원의 손실이 약 9200 수익으로 바뀐 것을 볼 수 있다

₿ 프로젝트 결과

```
*[20619] stop loss:-124111.1628 revenue:9852.4283
*[31259] stop loss:-119352.3565 revenue:-17684.7879
*[60599] stop loss:-124746.2893 revenue:11798.5843
*[64934] stop loss:-125326.0645 revenue:99407.0022
*[68766] stop loss:-124796.3333 revenue:164640.4876
*[71048] stop loss:-125211.5607 revenue:238869.3853
*[72897] stop loss:-124983.0276 revenue:207609.817
*[75231] stop loss:-122153.0288 revenue:250029.5944
*[77329] stop loss:-121251.9669 revenue:141397.435
*[77942] stop loss:-121883.0895 revenue:35264.9706
*[84022] stop loss:-125889.1009 revenue:63543.1957
*[96292] stop loss:-129997.6926 revenue:226472.2956
*[99419] stop loss:-123175.363 revenue:122596.3101
```

첫 모의테스트 변수를 그리드 서치를 통해 얻은 비율로 수정 후 -26000원 손절을 120000원 수익으로 바꾼 모습이다.



HTTPS://GITHUB.COM/KSP0321/TRADING.GIT