

git 주소 : <https://github.com/calpi0074/ai-and-cryptocurrency-projects-2022-spring>

그룹원 : 김선태, 유혁준

코드:

```
import pandas as pd
import math

file_path = "2024-05-01-upbit-BTC-book.csv"
df = pd.read_csv(file_path)
df['timestamp'] = pd.to_datetime(df['timestamp'])
start_time = "2024-05-01 00:00:01"
end_time = "2024-05-01 23:59:59"
filter = (df['timestamp'] >= start_time) & (df['timestamp'] <=
end_time)
filtered_df = df.loc[filter]
results = []
var = {'_flag': True}

def cal_mid_price(gr_bid_level, gr_ask_level, group_t):
    level = 15
    gr_rB = gr_bid_level.head(level)
    gr_rT = gr_ask_level.head(level)
    if len(gr_bid_level) > 0 and len(gr_ask_level) > 0:
        bid_top_price = gr_bid_level.iloc[0].price
        bid_top_level_qty = gr_bid_level.iloc[0].quantity
        ask_top_price = gr_ask_level.iloc[0].price
        ask_top_level_qty = gr_ask_level.iloc[0].quantity
        mid_price = (bid_top_price + ask_top_price) * 0.5
```

```
        return (mid_price, bid_top_price, ask_top_price,
bid_top_level_qty, ask_top_level_qty)
    else:
        return (-1, -1, -2, -1, -1)
```

```
def live_cal_book_i_v1(param, gr_bid_level, gr_ask_level, var,
mid):
```

```
    mid_price = mid
    ratio = param[0]
    level = param[1]
    interval = param[2]
    _flag = var['_flag']
    if _flag:
        var['_flag'] = False
        return 0.0
    quant_v_bid = gr_bid_level.quantity ** ratio
    price_v_bid = gr_bid_level.price * quant_v_bid
    quant_v_ask = gr_ask_level.quantity ** ratio
    price_v_ask = gr_ask_level.price * quant_v_ask
    askQty = quant_v_ask.values.sum()
    bidPx = price_v_bid.values.sum()
    bidQty = quant_v_bid.values.sum()
    askPx = price_v_ask.values.sum()
    bid_ask_spread = interval
    book_price = 0
    if bidQty > 0 and askQty > 0:
        book_price = (((askQty * bidPx) / bidQty) + ((bidQty *
askPx) / askQty)) / (bidQty + askQty)
        indicator_value = (book_price - mid_price) /
bid_ask_spread
    return indicator_value
```

```

def get_diff_count_units(diff):
    _count_1 = _count_0 = _units_traded_1 = _units_traded_0 =
    _price_1 = _price_0 = 0
    if diff is not None:
        diff_len = len(diff)
        if diff_len == 1:
            row = diff.iloc[0]
            if row['type'] == 1:
                _count_1 = row['count']
                _units_traded_1 = row['units_traded']
                _price_1 = row['price']
            else:
                _count_0 = row['count']
                _units_traded_0 = row['units_traded']
                _price_0 = row['price']
        elif diff_len == 2:
            row_1 = diff.iloc[1]
            row_0 = diff.iloc[0]
            _count_1 = row_1['count']
            _count_0 = row_0['count']
            _units_traded_1 = row_1['units_traded']
            _units_traded_0 = row_0['units_traded']
            _price_1 = row_1['price']
            _price_0 = row_0['price']
    return (_count_1, _count_0, _units_traded_1,
    _units_traded_0, _price_1, _price_0)

```

```

def live_cal_book_d_v1(param, gr_bid_level, gr_ask_level, diff,
var, mid_price):
    ratio = param[0]
    level = param[1]

```

```
interval = param[2]
decay = math.exp(-1.0 / interval)
_flag = var['_flag']
prevBidQty = var.get('prevBidQty', 0)
prevAskQty = var.get('prevAskQty', 0)
prevBidTop = var.get('prevBidTop', 0)
prevAskTop = var.get('prevAskTop', 0)
bidSideAdd = var.get('bidSideAdd', 0)
bidSideDelete = var.get('bidSideDelete', 0)
askSideAdd = var.get('askSideAdd', 0)
askSideDelete = var.get('askSideDelete', 0)
bidSideTrade = var.get('bidSideTrade', 0)
askSideTrade = var.get('askSideTrade', 0)
bidSideFlip = var.get('bidSideFlip', 0)
askSideFlip = var.get('askSideFlip', 0)
bidSideCount = var.get('bidSideCount', 0)
askSideCount = var.get('askSideCount', 0)
curBidQty = gr_bid_level['quantity'].sum()
curAskQty = gr_ask_level['quantity'].sum()
curBidTop = gr_bid_level.iloc[0].price
curAskTop = gr_ask_level.iloc[0].price
if _flag:
    var['prevBidQty'] = curBidQty
    var['prevAskQty'] = curAskQty
    var['prevBidTop'] = curBidTop
    var['prevAskTop'] = curAskTop
    var['_flag'] = False
    return 0.0
if curBidQty > prevBidQty:
    bidSideAdd += 1
    bidSideCount += 1
if curBidQty < prevBidQty:
```

```

        bidSideDelete += 1
        bidSideCount += 1
    if curAskQty > prevAskQty:
        askSideAdd += 1
        askSideCount += 1
    if curAskQty < prevAskQty:
        askSideDelete += 1
        askSideCount += 1
    if curBidTop < prevBidTop:
        bidSideFlip += 1
        bidSideCount += 1
    if curAskTop > prevAskTop:
        askSideFlip += 1
        askSideCount += 1
    (_count_1, _count_0, _units_traded_1, _units_traded_0,
    _price_1, _price_0) = diff
    bidSideTrade += _count_1
    bidSideCount += _count_1
    askSideTrade += _count_0
    askSideCount += _count_0
    if bidSideCount == 0:
        bidSideCount = 1
    if askSideCount == 0:
        askSideCount = 1
    bidBookV = (-bidSideDelete + bidSideAdd - bidSideFlip) /
    (bidSideCount ** ratio)
    askBookV = (askSideDelete - askSideAdd + askSideFlip) /
    (askSideCount ** ratio)
    tradeV = (askSideTrade / askSideCount ** ratio) -
    (bidSideTrade / bidSideCount ** ratio)
    bookDIndicator = askBookV + bidBookV + tradeV
    var['bidSideCount'] = bidSideCount * decay

```

```

var['askSideCount'] = askSideCount * decay
var['bidSideAdd'] = bidSideAdd * decay
var['bidSideDelete'] = bidSideDelete * decay
var['askSideAdd'] = askSideAdd * decay
var['askSideDelete'] = askSideDelete * decay
var['bidSideTrade'] = bidSideTrade * decay
var['askSideTrade'] = askSideTrade * decay
var['bidSideFlip'] = bidSideFlip * decay
var['askSideFlip'] = askSideFlip * decay
var['prevBidQty'] = curBidQty
var['prevAskQty'] = curAskQty
var['prevBidTop'] = curBidTop
var['prevAskTop'] = curAskTop
return bookDIndicator

```

```

for gr_o in filtered_df.groupby('timestamp'):
    gr_bid_level = gr_o[1][gr_o[1].type == 0]
    gr_ask_level = gr_o[1][gr_o[1].type == 1]
    mid_price, bid, ask, bid_qty, ask_qty =
cal_mid_price(gr_bid_level, gr_ask_level, gr_o)
    param = [0.2, 5, 1]
    diff = get_diff_count_units(gr_o[1])
    book_imbalance = live_cal_book_i_v1(param, gr_bid_level,
gr_ask_level, var, mid_price)
    book_D = live_cal_book_d_v1(param, gr_bid_level,
gr_ask_level, diff, var, mid_price)
    results.append([gr_o[1].iloc[0]['timestamp'], mid_price,
book_imbalance, book_D])

```

```

result_df = pd.DataFrame(results, columns=['timestamp',
'mid_price', 'book_imbalance', 'book_D'])
result_df.to_csv("2024-05-01-00:00~24:00-upbit-BTC-

```

feature.csv", index=False)