```
Git push 에러로 인해 새로운 repo 에 데이터만 push 했습니다.
새로운 repo : https://github.com/wannacrypto/AI_crypto_data

융합전자공학부 16 학번 김선호
정보시스템학과 20 학번 김요섭

import pandas as pd
import time
import os
import datetime
import math
from tqdm import tqdm

pd.set_option('display.max_columns', 100)

def midprice():
    orderbook_df = pd.read_csv("./orderbook_merge_data/mod_orderbook.csv")

    time_stamp_list = []
    mid_price_list = []

    for i in tqdm(range(int(len(orderbook_df)/10))):
        df_order = orderbook_df.loc[10*i:(10*i)+9].reset_index(drop=True)
        time_stamp_list.append(df_order.iloc[0]['timestamp'])

        df_bid = df_order.loc[0:4]
        df_ask = df_order.loc[5:9]

        df_bid = df_bid.sort_values(by=['price'], axis=0,
ascending=False).reset_index(drop=True)
        df_ask = df_ask.sort_values(by=['price'], axis=0,
ascending=True).reset_index(drop=True)

        top_bid = df_bid.iloc[0]['price']
        top_ask = df_ask.iloc[0]['price']

        mid_price = (top_bid+top_ask)/2
        mid_price_list.append(mid_price)

    timestamp_series = pd.Series(time_stamp_list)
    mid_price_list = pd.Series(mid_price_list)

    result_df = pd.concat([timestamp_series,mid_price_list],axis=1)

    result_df.columns = ['timestamp','mid_price']

    print(result_df)
```

```
    result_df.to_csv('./result/midprice.csv',sep=',',index=False)


def Book_I():
    orderbook_df = pd.read_csv("./orderbook_merge_data/mod_orderbook.csv")

    ratio = 0.2
    level = 5
    interval = 1

    time_stamp_list = []
    book_i_list = []

    for i in tqdm(range(int(len(orderbook_df)/10))):
        askqty = bidqty = askpx = bidpx = book_p = 0

        df_order = orderbook_df.loc[10*i:(10*i)+9].reset_index(drop=True)
        time_stamp_list.append(df_order.iloc[0]['timestamp'])

        df_bid = df_order.loc[0:4]
        df_ask = df_order.loc[5:9]

        df_bid = df_bid.sort_values(by=['price'], axis=0,
ascending=False).reset_index(drop=True)
        df_ask = df_ask.sort_values(by=['price'], axis=0,
ascending=True).reset_index(drop=True)

        top_bid = df_bid.iloc[0]['price']
        top_ask = df_ask.iloc[0]['price']

        mid_price = (top_bid+top_ask)/2


        if(mid_price != 0):


            for j in range(int(len(df_bid))):
                bidqty += df_bid.iloc[j]['quantity']**ratio
                bidpx +=
df_bid.iloc[j]['price']*(df_bid.iloc[j]['quantity']**ratio)

            for k in range(int(len(df_ask))):
                askqty += df_ask.iloc[j]['quantity']**ratio
                askpx +=
df_ask.iloc[j]['price']*(df_ask.iloc[j]['quantity']**ratio)

            book_p = ( ((askqty*bidpx)/bidqty) + ((bidqty*askpx)/askqty) )/
(bidqty+askqty)
```

```
            book_i = (book_p - mid_price)/interval
            book_i_list.append(book_i)

        else:
            book_i_list.append(0)

    timestamp_series = pd.Series(time_stamp_list)
    book_i_series = pd.Series(book_i_list)

    result_df =
pd.concat([timestamp_series,book_i_series],axis=1).reset_index(drop=True)

    result_df.columns = ['timestamp','book-imbalance-0.2-5-1']

    print(result_df)

    result_df.to_csv('./result/book_i.csv',sep=',',index=False)

def Book_D():
    ratio = 0.2
    level = 5
    interval = 1
    decay = math.exp(-1.0/interval)

    time_stamp_list = []
    book_d_list = []

    cur_askqty = cur_asktop = cur_bidqty = cur_bidtop = 0
    prev_askqty = prev_asktop = prev_bidqty = prev_bidtop = 0

    bidsideadd = 0
    bidsidecount = 0
    bidsidedelete = 0
    bidsideflip = 0

    asksideadd = 0
    asksidecount = 0
    asksidedelete = 0
    asksideflip = 0

    bidsidetrade = 0
    asksidetrade = 0

    orderbook_df = pd.read_csv("./orderbook_merge_data/mod_orderbook.csv")
    trade_df = pd.read_csv("./trade_merge_data/mod_trade.csv")

    #0,1,2,...
    for i in tqdm(range(int(len(orderbook_df)/10))):
```

```
        order_10level_df =
orderbook_df.loc[10*i:(10*i)+9].reset_index(drop=True)
        ts = order_10level_df.iloc[0]['timestamp']
        time_stamp_list.append(ts)

        df_bid = order_10level_df.loc[0:4]
        df_ask = order_10level_df.loc[5:9]

        df_bid = df_bid.sort_values(by=['price'], axis=0,
ascending=False).reset_index(drop=True)
        df_ask = df_ask.sort_values(by=['price'], axis=0,
ascending=True).reset_index(drop=True)

        cur_bidtop = df_bid.iloc[0]['price']
        cur_bidqty = df_bid['quantity'].sum()
        cur_asktop = df_ask.iloc[0]['price']
        cur_askqty = df_ask['quantity'].sum()

        if (i == 0):
            bookd = 0
            prev_askqty = cur_askqty
            prev_asktop = cur_asktop
            prev_bidqty = cur_bidqty
            prev_bidtop = cur_bidtop
            continue

        if(cur_bidqty>prev_bidqty):
            bidsideadd += 1
            bidsidecount  += 1

        if(cur_bidqty<prev_bidqty):
            bidsidedelete += 1
            bidsidecount  += 1

        if(cur_askqty>prev_askqty):
            asksideadd += 1
            asksidecount  += 1

        if(cur_askqty<prev_askqty):
            asksidedelete += 1
            asksidecount  += 1

        if(cur_bidtop<prev_bidtop):
            bidsideflip += 1
            bidsidecount  += 1

        if(cur_asktop>prev_asktop):
            asksideflip += 1
```

```
            asksidecount  += 1

        temp_df = trade_df[trade_df['timestamp'] == ts]

        if(not(temp_df.empty)):
            try:
                bidcnt = temp_df['type'].value_counts()[0]
            except:
                bidcnt = 0

            try:
                askcnt = temp_df['type'].value_counts()[1]
            except:
                askcnt = 0

        else:
            bidcnt = 0
            askcnt = 0

        bidsidetrade += bidcnt
        bidsidecount += bidcnt
        asksidetrade += askcnt
        asksidecount += askcnt

        bidbookv = (bidsideadd - bidsidedelete - bidsideflip) /
(bidsidecount**ratio)
        askbookv = (asksidedelete-asksideadd+asksideflip) /
(asksidecount**ratio)
        tradev = (asksidetrade/asksidecount**ratio)-
(bidsidetrade/bidsidecount**ratio)

        bookDindicator = askbookv+bidbookv+tradev

        book_d_list.append(bookDindicator)

        bidsideadd *= decay
        bidsidecount *= decay
        bidsidedelete *= decay
        bidsideflip *= decay

        asksideadd *= decay
        asksidecount *= decay
        asksidedelete *= decay
        asksideflip *= decay

        bidsidetrade *= decay
        asksidetrade *= decay
```

```
        prev_bidqty = cur_bidqty
        prev_bidtop = cur_bidtop
        prev_askqty = cur_askqty
        prev_asktop = cur_bidtop

    timestamp_series = pd.Series(time_stamp_list)
    book_d_series = pd.Series(book_d_list)

    result_df =
pd.concat([timestamp_series,book_d_series],axis=1).reset_index(drop=True)

    result_df.columns = ['timestamp','book-delta-v1-0.2-5-1']

    print(result_df)

    result_df.to_csv('./result/book_d.csv',sep=',',index=False)



def merge():
    m_df = pd.read_csv("./result/midprice.csv")
    i_df = pd.read_csv("./result/book_i.csv")
    d_df = pd.read_csv("./result/book_d.csv")
    print(d_df['book-delta-v1-0.2-5-1'],i_df['book-imbalance-0.2-5-
1'],m_df['mid_price'],m_df['timestamp'])
    result_df = pd.concat([d_df['book-delta-v1-0.2-5-1'],i_df['book-imbalance-
0.2-5-1'],m_df['mid_price'],m_df['timestamp']],axis=1)
    result_df = result_df.reset_index(drop=True)
    result_df.to_csv('./result/2022-11-25_26_27-upbit-btc-krw-
feature.csv',sep=',',index=False)

# midprice()
# Book_I()
Book_D()
time.sleep(2)
merge()
```