Git push 에러로 인해 새로운 repo에 데이터만 push했습니다.

새로운 repo : https://github.com/wannacrypto/AI\_crypto\_data

융합전자공학부 16학번 김선호

정보시스템학과 20학번 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()