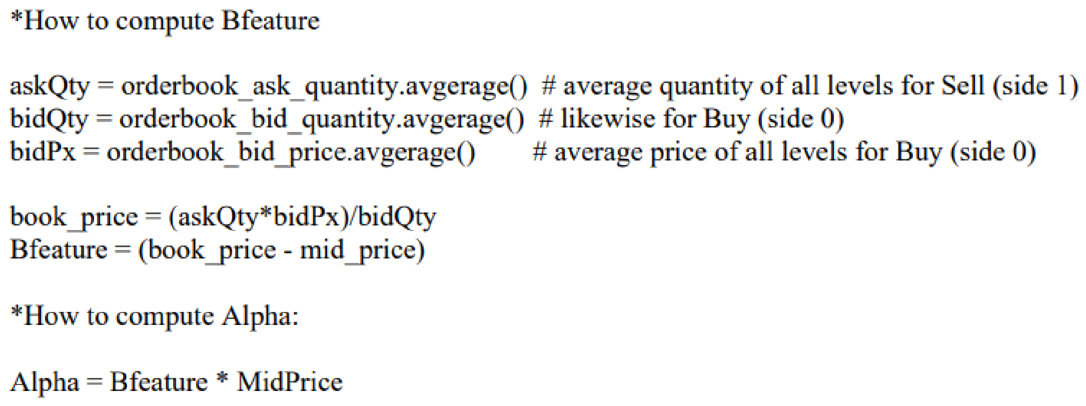
Midterm exam

2018007692 lee ji yun

Task 1, 2, and 3 were solved, and No. 4 was not solved because there were many things such as test schedules and reservations. Among the problems, Task3 seems to be confusing whether the word ‘average price of all levels for Buy and Sell’ means to extract values from the entire order book data or to extract values separately from the corresponding order book data at each time. So Task3 created two versions of the code.
Thank you.

Task 1,2,3번을 풀었고, 4번은 시험일정, 예비군 등 많은 일이 있어서 풀지 못했습니다. 문제들 중에 Task3은 난이도가 어렵다기 보다는 average price of all levels for Buy,Sell 부분에서 all levels라는 말뜻이 오더북 전체 데이터에서 값을 추출하라는 것인지, 각 시간마다 해당되는 오더북의 데이터에서만 각각 따로 값을 추출하라는 말인지 헷갈리게 되어있는 것 같습니다. 그래서 Task3은 두가지 버전으로 코드를 만들었습니다.

감사합니다.



*How to compute Bfeature

```
askQty = orderbook_ask_quantity.avgerage()  # average quantity of all levels for Sell (side 1)
bidQty = orderbook_bid_quantity.avgerage()  # likewise for Buy (side 0)
bidPx = orderbook_bid_price.avgerage()       # average price of all levels for Buy (side 0)

book_price = (askQty*bidPx)/bidQty
Bfeature = (book_price - mid_price)
```

*How to compute Alpha:

Alpha = Bfeature * MidPrice

# Untitled2

November 5, 2022

[29]:
```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
df=pd.read_csv('C:/Users/dlwld/Desktop/    /2019-05-trade.csv')

#Task1
def findExactProfit():
    i=0
    Sell=0
    Buy=0
    while(i<len(df)):
        if df['side'][i]==1:
            Sell+=df['quantity'][i]*df['price'][i]
        if df['side'][i]==0:
            Buy+=df['quantity'][i]*df['price'][i]
        i+=1
    Total=Sell-Buy
    Total=math.floor(Total*10000)/10000
    return Total
print(findExactProfit())
```

```
18152538.9211
```

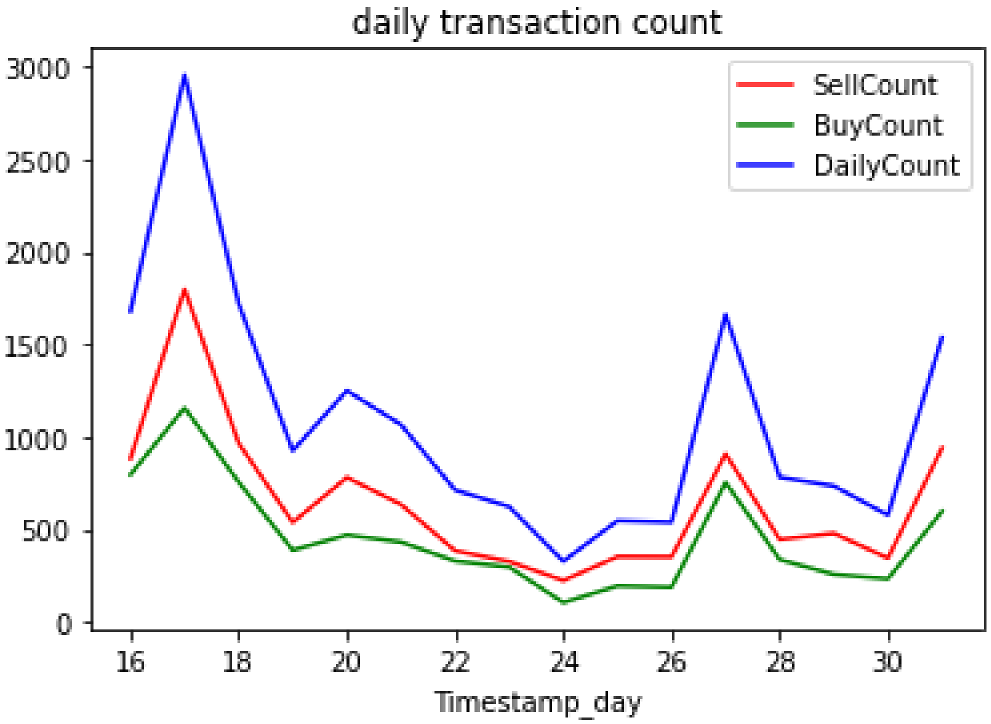
[30]:
```
#Task2
def findSellCount(a):
    l=[]
    for i in range(0,len(a)):
        j=0
        s=0
        while(j<len(df)):
            if(df['timestamp_days'][j]==a[i])&(df['side'][j]==1):
                s+=1
            j+=1
        l.append(s)
    return l

def findBuyCount(a):
    l=[]
    for i in range(0,len(a)):
        j=0
        s=0
        while(j<len(df)):
            if (df['timestamp_days'][j]==a[i])&(df['side'][j]==0):
                s+=1
            j+=1
        l.append(s)
    return l

df['timestamp_days']=pd.to_datetime(df['timestamp']).dt.day
days=[16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]
Sell=findSellCount(days)
Buy=findBuyCount(days)
Transaction=[x+y for x,y in zip(Sell, Buy)]

task2_list=[
    ['timestamp_days',days],
    ['Sell',Sell],
    ['Buy',Buy],
    ['Sum',Transaction]
]
df2=pd.DataFrame.from_dict(dict(task2_list))
df2
plt.title('daily transaction count')
plt.plot(days,Sell,'r',label='SellCount')
plt.plot(days,Buy,'g',label='BuyCount')
plt.plot(days,Transaction,'b',label='DailyCount')
plt.xlabel('Timestamp_day')
plt.ylabel('')
plt.legend()
plt.show
```

```
[30]: <function matplotlib.pyplot.show(close=None, block=None)>
```



[54]:

```
#Task3
#orderbook
dfo=pd.read_csv('C:/Users/dlwld/Desktop/    /2019-05-17-BTC-orderbook.csv')
df=pd.read_csv('C:/Users/dlwld/Desktop/    /2019-05-trade.csv')

#trade  17
df['timestamp_days']=pd.to_datetime(df['timestamp']).dt.day
l=[]
for i in range(0,len(df)):
    if df['timestamp_days'][i]==17:
        l.append(i)
df2=df.loc[l]
# 17     timestamp
# df2=df1.drop_duplicates(['timestamp'])
df2=df2.drop(['fee','timestamp_days','amount'],axis='columns')


#
df2=df2.reset_index()
df2=df2.drop('index',axis=1)

dfo['timestamp'] = pd.to_datetime(dfo['timestamp'])
dfo1=dfo[dfo['timestamp'].dt.second==0]
```

```
# df2timestamp        (2019-05-17 00:00:00)

tmp=df2['timestamp']
tmp_val=tmp.values
timestamp_df2=tmp_val.tolist()

# dfo1timestamp           (2019-05-17 00:00:00)
ts=[]
dfo1=dfo1.reset_index(drop=True)
for i in range(0,len(dfo1)):
    tp=dfo1['timestamp'][i]
    ts.append(tp.strftime('%Y-%m-%d %H:%M'))
timestamp_dfo1=ts

dfo1['timestamp']=timestamp_dfo1


# # MidPrice

MidPrice=[]
TopLevelBuy=[]
TopLevelSell=[]
# top_buy_price
for i in range(0,len(df2)):
    for j in range(0,len(dfo1)):
        if timestamp_df2[i]==timestamp_dfo1[j]:
            if dfo1['type'][j]==0:
                TopLevelBuy.append(dfo1['price'][j])
                break

# top_sell_price
for i in range(0,len(df2)):
    for j in range(0,len(dfo1)):
        if timestamp_df2[i]==timestamp_dfo1[j]:
            if dfo1['type'][j]==1:
                TopLevelSell.append(dfo1['price'][j])
                break
MidPrice=[(x+y)/2 for x,y in zip(TopLevelBuy,TopLevelSell)]
df2['midprice']=MidPrice

# Bfeature,Alpha

# askQty,bidQty,bidPx,book_price
# groupy         ,
dfo_group=dfo.groupby('type').mean()
```

```
askQty=dfo_group['quantity'][1]
bidQty=dfo_group['quantity'][0]
bidPx=dfo_group['price'][0]
book_price=(askQty*bidPx)/bidQty
bfeature=[]
for i in range(0,len(MidPrice)):
    bfeature.append(book_price-MidPrice[i])

df2['bfeature']=bfeature



alpha=[(x*y) for x,y in zip(bfeature,MidPrice)]
df2['alpha']=alpha
df2=df2[['timestamp','quantity','price','midprice','bfeature','alpha','side']]

#
df2.to_csv(" new_2019_05_trade.csv", mode='w')

df2.head(20)
```

```
[54]:            timestamp  quantity    price   midprice      bfeature  \
      0   2019-05-17 00:00  0.057701  9449000  9449500.0  1.068276e+06
      1   2019-05-17 00:00  0.005000  9449000  9449500.0  1.068276e+06
      2   2019-05-17 00:00  0.127708  9449000  9449500.0  1.068276e+06
      3   2019-05-17 00:00  1.057672  9449000  9449500.0  1.068276e+06
      4   2019-05-17 00:00  0.068212  9449000  9449500.0  1.068276e+06
      5   2019-05-17 00:02  0.003361  9472000  9459500.0  1.058276e+06
      6   2019-05-17 00:02  0.022236  9472000  9459500.0  1.058276e+06
      7   2019-05-17 00:02  0.001468  9472000  9459500.0  1.058276e+06
      8   2019-05-17 00:03  0.000026  9474000  9473000.0  1.044776e+06
      9   2019-05-17 00:03  0.003637  9472000  9473000.0  1.044776e+06
      10  2019-05-17 00:03  0.010597  9472000  9473000.0  1.044776e+06
      11  2019-05-17 00:03  0.013971  9472000  9473000.0  1.044776e+06
      12  2019-05-17 00:03  0.006700  9472000  9473000.0  1.044776e+06
      13  2019-05-17 00:03  0.044558  9472000  9473000.0  1.044776e+06
      14  2019-05-17 00:03  0.023373  9472000  9473000.0  1.044776e+06
      15  2019-05-17 00:03  0.014385  9472000  9473000.0  1.044776e+06
      16  2019-05-17 00:05  0.110682  9474000  9477500.0  1.040276e+06
      17  2019-05-17 00:05  0.601963  9472000  9477500.0  1.040276e+06
      18  2019-05-17 00:05  0.065100  9471000  9477500.0  1.040276e+06
      19  2019-05-17 00:05  0.498738  9468000  9477500.0  1.040276e+06

                 alpha  side
      0   1.009468e+13     1
      1   1.009468e+13     1
      2   1.009468e+13     1
      3   1.009468e+13     1
```

```
4   1.009468e+13     1
5   1.001076e+13     1
6   1.001076e+13     1
7   1.001076e+13     1
8   9.897166e+12     1
9   9.897166e+12     1
10  9.897166e+12     1
11  9.897166e+12     1
12  9.897166e+12     1
13  9.897166e+12     1
14  9.897166e+12     1
15  9.897166e+12     1
16  9.859218e+12     1
17  9.859218e+12     1
18  9.859218e+12     1
19  9.859218e+12     1
```

[55]:
```
df2.tail(20)
```

[55]:
```
             timestamp  quantity    price   midprice      bfeature  \
2934  2019-05-17 23:39  0.761881  8658000  8653000.0  1.864776e+06
2935  2019-05-17 23:40  0.020298  8652000  8656500.0  1.861276e+06
2936  2019-05-17 23:40  0.022291  8643000  8656500.0  1.861276e+06
2937  2019-05-17 23:40  0.010000  8644000  8656500.0  1.861276e+06
2938  2019-05-17 23:41  0.000156  8650000  8646000.0  1.871776e+06
2939  2019-05-17 23:41  0.292016  8650000  8646000.0  1.871776e+06
2940  2019-05-17 23:42  0.539100  8642000  8644500.0  1.873276e+06
2941  2019-05-17 23:43  0.310700  8621000  8632500.0  1.885276e+06
2942  2019-05-17 23:43  0.019500  8620000  8632500.0  1.885276e+06
2943  2019-05-17 23:43  0.255600  8620000  8632500.0  1.885276e+06
2944  2019-05-17 23:48  0.371000  8640000  8642000.0  1.875776e+06
2945  2019-05-17 23:48  0.006948  8640000  8642000.0  1.875776e+06
2946  2019-05-17 23:48  0.006948  8640000  8642000.0  1.875776e+06
2947  2019-05-17 23:48  0.010000  8638000  8642000.0  1.875776e+06
2948  2019-05-17 23:49  0.055531  8648000  8641500.0  1.876276e+06
2949  2019-05-17 23:49  0.135370  8648000  8641500.0  1.876276e+06
2950  2019-05-17 23:49  0.196965  8640000  8641500.0  1.876276e+06
2951  2019-05-17 23:51  0.196900  8650000  8648500.0  1.869276e+06
2952  2019-05-17 23:58  0.078198  8599000  8600500.0  1.917276e+06
2953  2019-05-17 23:58  2.073502  8600000  8600500.0  1.917276e+06

             alpha  side
2934  1.613591e+13     1
2935  1.611214e+13     1
2936  1.611214e+13     0
2937  1.611214e+13     0
2938  1.618338e+13     1
```

```
2939  1.618338e+13     1
2940  1.619354e+13     1
2941  1.627465e+13     0
2942  1.627465e+13     0
2943  1.627465e+13     0
2944  1.621046e+13     1
2945  1.621046e+13     1
2946  1.621046e+13     1
2947  1.621046e+13     1
2948  1.621384e+13     1
2949  1.621384e+13     1
2950  1.621384e+13     0
2951  1.616644e+13     1
2952  1.648953e+13     0
2953  1.648953e+13     0
```

[31]:

```python
#Task3 (other version - askQty,bidQty's definition is too ambiguous (average
# quantity of all levels for Sell).                ,
#
#          .))

dfo=pd.read_csv('C:/Users/dlwld/Desktop/    /2019-05-17-BTC-orderbook.csv')
df=pd.read_csv('C:/Users/dlwld/Desktop/    /2019-05-trade.csv')

#trade  17
df['timestamp_days']=pd.to_datetime(df['timestamp']).dt.day
l=[]
for i in range(0,len(df)):
    if df['timestamp_days'][i]==17:
        l.append(i)
df2=df.loc[l]
# 17     timestamp
# df2=df1.drop_duplicates(['timestamp'])
df2=df2.drop(['fee','timestamp_days','amount'],axis='columns')



#
df2=df2.reset_index()
df2=df2.drop('index',axis=1)

dfo['timestamp'] = pd.to_datetime(dfo['timestamp'])
dfo1=dfo[dfo['timestamp'].dt.second==0]

# df2timestamp        (2019-05-17 00:00:00)

tmp=df2['timestamp']
tmp_val=tmp.values
```

```
timestamp_df2=tmp_val.tolist()

# dfo1timestamp           (2019-05-17 00:00:00)
ts=[]
dfo1=dfo1.reset_index(drop=True)
for i in range(0,len(dfo1)):
    tp=dfo1['timestamp'][i]
    ts.append(tp.strftime('%Y-%m-%d %H:%M'))
timestamp_dfo1=ts

dfo1['timestamp']=timestamp_dfo1


# # MidPrice

MidPrice=[]
TopLevelBuy=[]
TopLevelSell=[]
# top_buy_price
for i in range(0,len(df2)):
    for j in range(0,len(dfo1)):
        if timestamp_df2[i]==timestamp_dfo1[j]:
            if dfo1['type'][j]==0:
                TopLevelBuy.append(dfo1['price'][j])
                break

# top_sell_price
for i in range(0,len(df2)):
    for j in range(0,len(dfo1)):
        if timestamp_df2[i]==timestamp_dfo1[j]:
            if dfo1['type'][j]==1:
                TopLevelSell.append(dfo1['price'][j])
                break
MidPrice=[(x+y)/2 for x,y in zip(TopLevelBuy,TopLevelSell)]
df2['midprice']=MidPrice


# Bfeature,Alpha

# askQty,bidQty,bidPx,book_price
askQty=[]
bidQty=[]
bidPx=[]
book_price=[]
Bfeature=[]
for i in range(0,len(df2)):
    Sum1=0
    Sum2=0
    Sum3=0
    Sum1_Count=0
    Sum2_Count=0
    for j in range(0,len(dfo1)):
        if timestamp_df2[i]==timestamp_dfo1[j]:
            if dfo1['type'][j]==1:
                Sum1+=dfo1['quantity'][j]
                Sum1_Count+=1
            else:
                Sum2+=dfo1['quantity'][j]
                Sum3+=dfo1['price'][j]
                Sum2_Count+=1
    askQty.append(Sum1/Sum1_Count)
    bidQty.append(Sum2/Sum2_Count)
    bidPx.append(Sum3/Sum2_Count)
    book_price.append(((Sum1/Sum1_Count)*(Sum3/Sum2_Count))/(Sum2/Sum2_Count))

Bfeature=[(x-y) for x,y in zip(book_price,MidPrice)]
df2['bfeature']=Bfeature

alpha=[(x*y) for x,y in zip(Bfeature,MidPrice)]
df2['alpha']=alpha

df2=df2[['timestamp','quantity','price','midprice','bfeature','alpha','side']]
df2.head(20)
```

[31]:

```
           timestamp  quantity    price   midprice      bfeature  \
0   2019-05-17 00:00  0.057701  9449000  9449500.0 -2.699152e+06
1   2019-05-17 00:00  0.005000  9449000  9449500.0 -2.699152e+06
2   2019-05-17 00:00  0.127708  9449000  9449500.0 -2.699152e+06
3   2019-05-17 00:00  1.057672  9449000  9449500.0 -2.699152e+06
4   2019-05-17 00:00  0.068212  9449000  9449500.0 -2.699152e+06
5   2019-05-17 00:02  0.003361  9472000  9459500.0 -8.083382e+06
6   2019-05-17 00:02  0.022236  9472000  9459500.0 -8.083382e+06
7   2019-05-17 00:02  0.001468  9472000  9459500.0 -8.083382e+06
8   2019-05-17 00:03  0.000026  9474000  9473000.0 -7.796323e+05
9   2019-05-17 00:03  0.003637  9472000  9473000.0 -7.796323e+05
10  2019-05-17 00:03  0.010597  9472000  9473000.0 -7.796323e+05
11  2019-05-17 00:03  0.013971  9472000  9473000.0 -7.796323e+05
12  2019-05-17 00:03  0.006700  9472000  9473000.0 -7.796323e+05
13  2019-05-17 00:03  0.044558  9472000  9473000.0 -7.796323e+05
14  2019-05-17 00:03  0.023373  9472000  9473000.0 -7.796323e+05
15  2019-05-17 00:03  0.014385  9472000  9473000.0 -7.796323e+05
16  2019-05-17 00:05  0.110682  9474000  9477500.0  2.145082e+06
17  2019-05-17 00:05  0.601963  9472000  9477500.0  2.145082e+06
18  2019-05-17 00:05  0.065100  9471000  9477500.0  2.145082e+06
```

```
19  2019-05-17 00:05  0.498738  9468000  9477500.0  2.145082e+06

           alpha  side
0  -2.550564e+13     1
1  -2.550564e+13     1
2  -2.550564e+13     1
3  -2.550564e+13     1
4  -2.550564e+13     1
5  -7.646475e+13     1
6  -7.646475e+13     1
7  -7.646475e+13     1
8  -7.385457e+12     1
9  -7.385457e+12     1
10 -7.385457e+12     1
11 -7.385457e+12     1
12 -7.385457e+12     1
13 -7.385457e+12     1
14 -7.385457e+12     1
15 -7.385457e+12     1
16  2.033002e+13     1
17  2.033002e+13     1
18  2.033002e+13     1
19  2.033002e+13     1
```

[32]:

```
df2.tail(20)
```

[32]:

```
             timestamp  quantity    price   midprice      bfeature  \
2934  2019-05-17 23:39  0.761881  8658000  8653000.0  1.561910e+06
2935  2019-05-17 23:40  0.020298  8652000  8656500.0 -2.612918e+06
2936  2019-05-17 23:40  0.022291  8643000  8656500.0 -2.612918e+06
2937  2019-05-17 23:40  0.010000  8644000  8656500.0 -2.612918e+06
2938  2019-05-17 23:41  0.000156  8650000  8646000.0 -4.118015e+06
2939  2019-05-17 23:41  0.292016  8650000  8646000.0 -4.118015e+06
2940  2019-05-17 23:42  0.539100  8642000  8644500.0 -1.709677e+06
2941  2019-05-17 23:43  0.310700  8621000  8632500.0  1.094520e+06
2942  2019-05-17 23:43  0.019500  8620000  8632500.0  1.094520e+06
2943  2019-05-17 23:43  0.255600  8620000  8632500.0  1.094520e+06
2944  2019-05-17 23:48  0.371000  8640000  8642000.0 -2.713221e+06
2945  2019-05-17 23:48  0.006948  8640000  8642000.0 -2.713221e+06
2946  2019-05-17 23:48  0.006948  8640000  8642000.0 -2.713221e+06
2947  2019-05-17 23:48  0.010000  8638000  8642000.0 -2.713221e+06
2948  2019-05-17 23:49  0.055531  8648000  8641500.0 -2.382326e+06
2949  2019-05-17 23:49  0.135370  8648000  8641500.0 -2.382326e+06
2950  2019-05-17 23:49  0.196965  8640000  8641500.0 -2.382326e+06
2951  2019-05-17 23:51  0.196900  8650000  8648500.0 -7.486342e+05
2952  2019-05-17 23:58  0.078198  8599000  8600500.0 -1.669804e+06
2953  2019-05-17 23:58  2.073502  8600000  8600500.0 -1.669804e+06
```

```
              alpha  side
2934  1.351520e+13     1
2935 -2.261873e+13     1
2936 -2.261873e+13     0
2937 -2.261873e+13     0
2938 -3.560435e+13     1
2939 -3.560435e+13     1
2940 -1.477930e+13     1
2941  9.448447e+12     0
2942  9.448447e+12     0
2943  9.448447e+12     0
2944 -2.344766e+13     1
2945 -2.344766e+13     1
2946 -2.344766e+13     1
2947 -2.344766e+13     1
2948 -2.058687e+13     1
2949 -2.058687e+13     1
2950 -2.058687e+13     0
2951 -6.474563e+12     1
2952 -1.436115e+13     0
2953 -1.436115e+13     0
```

```
[ ]:
```