# **Queries and Doubts**

1. What is filler data?

Ans: dummy data

1. can we use news data or other data other than book and trade data?

Ans: No because we do not know the exact time! time\_id is *shuffled*

1. why do some stocks miss data or their analysis in graphs is not showing?

Ans: Those stocks have missing data.

1. When calculating realised volatility did you first ffill for the missing seconds in book data?

Ans: Actually, forward fill does not affect the volatility calculation because log (s\_t2/s\_t1) = log(1) = 0 in the ffill period. Prices/WAP remain the same. It doesn't affect volatility.

1. Are the bid\_price1 and bid\_price1 from different time\_ids and stocks comparable? E.g. if bid\_price1 = 0.9 in stock\_id = 10 and time\_id = 5 equal to bid\_price1 = 0.9 in stock id = 20 and time\_id = 11 equal? Similar question for ask\_price1 ?
   1. Is the price in book data and trade data comparable? Have they been normalised together or separately. If normalised together then they are comparable if separately then they are NOT comparable.

Possible ans: <https://www.kaggle.com/competitions/optiver-realized-volatility-prediction/discussion/249474>

Normalisation is done separately for time, stock id so different time, stock id prices should not be comparable, but prices across the same stock id for same time id should be comparable as they would have the same mean and std? -> makes sense

1. Probably together based on inference from the discussion thread -> you mean book and trade data are comparable for the same stock\_id and time\_id right?

yep

Thanks @fegetable for your reply 🙂. But then, how can we compare the calculated realized volatility using the WAP formula across time periods of the same stock? If the bid/ask prices are not comparable across different time\_ids then how is the calculated volatility comparable?

I was wondering about this also haha, <https://www.kaggle.com/c/optiver-realized-volatility-prediction/discussion/267327>

This thread in particular:

--“Here it seems to mean that all price start at 1, so probably dividing each time seris by the initial price. Global price have been deanonymised by looking at ticks.”

-- “Thanks for commenting. Dividing time series by a global (per stock) value, which happens to be the price at some time point, makes some sense, however I observed that that value is revisited often, see notebook above. This suggests that the divisor is rather local than global, which could mean that every time slot was independently "normalized".

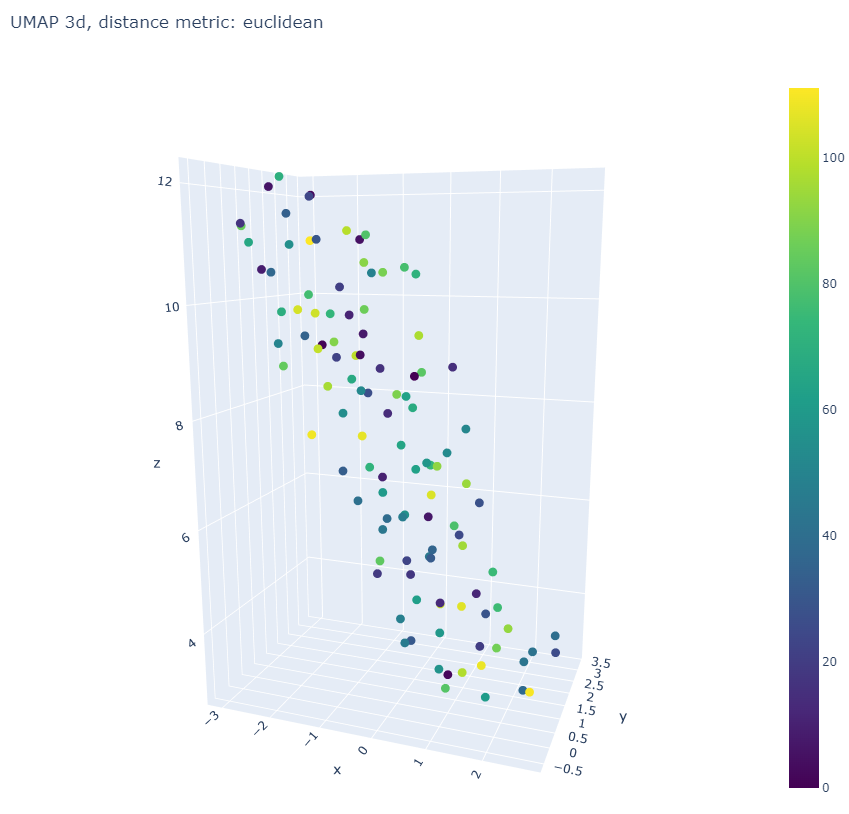
How about the standard deviation? Multiplication/Division by a constant changes it accordingly but we actually want the volatility to be consistent over time slots and stocks, do we?”

-- “Dividing all prices by a constant (per stock and per time bucket) value doesn't change the returns and the volatility: The return is defined as the ratio between two subsequent prices and this ratio is invariant under the chosen transformation.

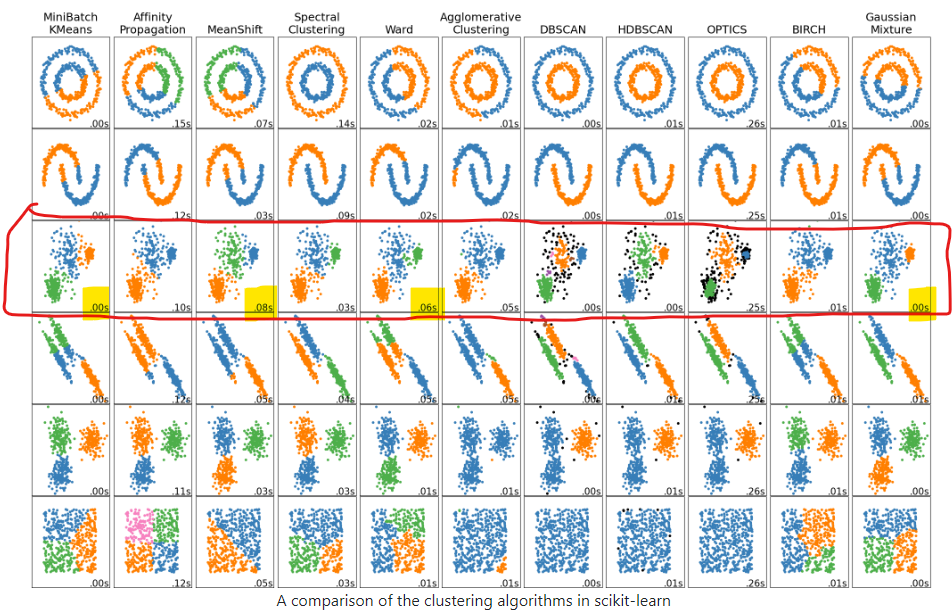
If you look at the logs: dividing the prices by a constant is equivalent to subtracting a constant from all logs of prices. The log return (as difference of two subsequent logs of prices) doesn't change under this transformation.”

From the discussion above, im understanding that since normalisation is done by dividing by a constant, the calculation of log returns and therefore realised vol for every individual stock and time id is not affected.

@fegetable, good job! Log( s\_t2/k / s\_t1/k ) is same as Log( s\_t2 / s\_t1 ) for all time\_id. We don't care about different k in different time\_id as they always get cancelled out.

6. What clustering algo. can separate the stocks into clusters (similar stocks based on summary stats. features) if stocks’ target volatility (in 2nd 10 mins) are distributed like below? Each colour represents one of the 112 stocks. The 3 dimensional view is after reducing 7 dimensions of ['mean\_vol','std\_vol','min\_vol','p25\_vol', 'median\_vol', 'p75\_vol','max\_vol'] using UMAP algo.

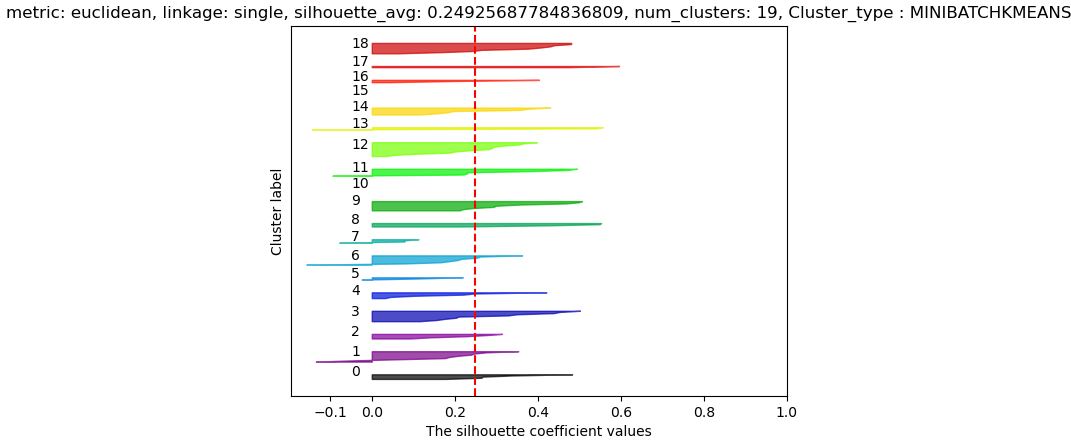
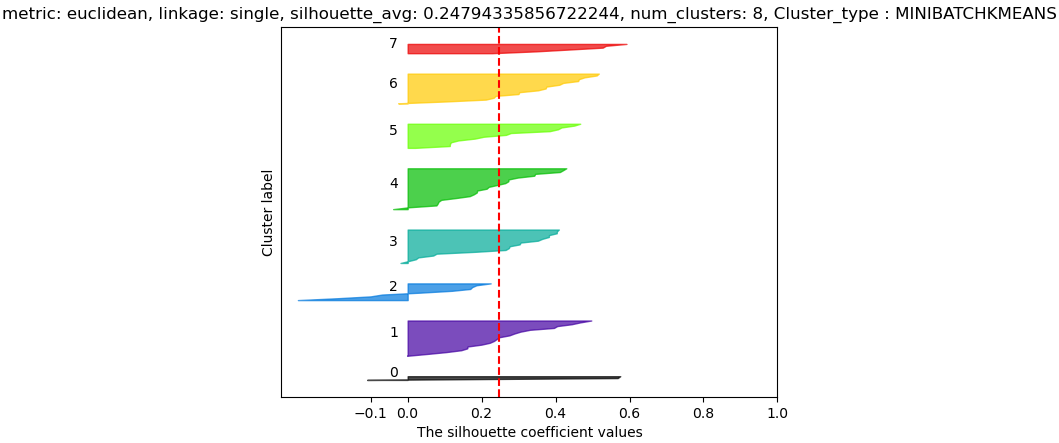
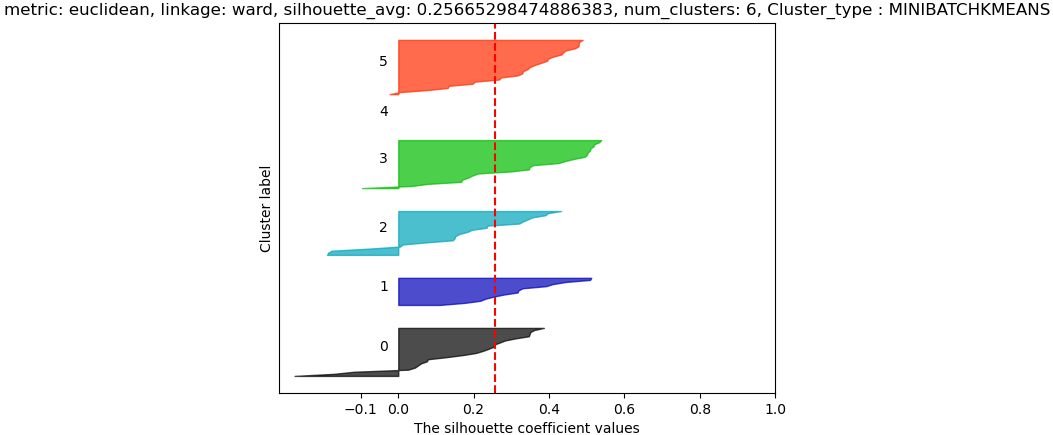
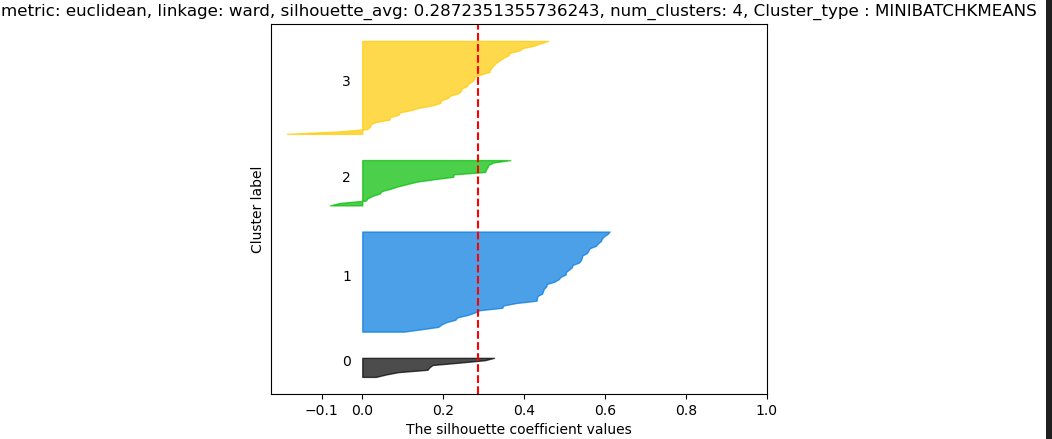
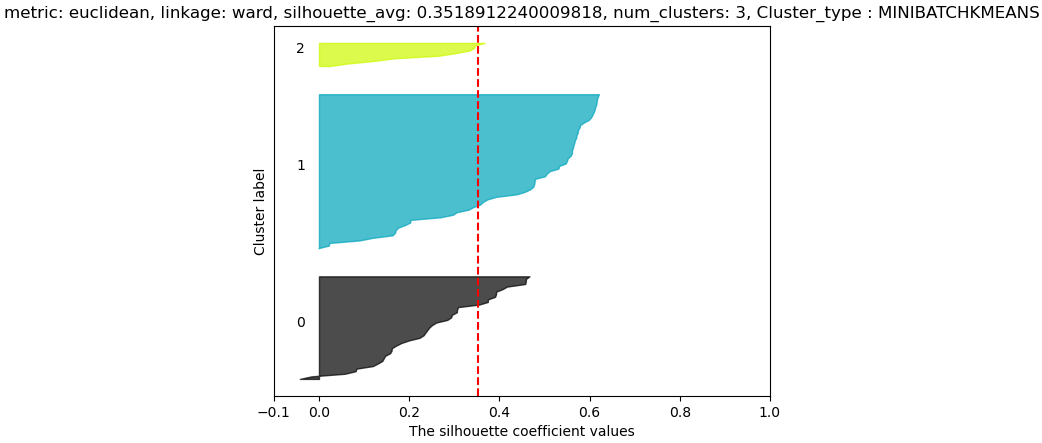
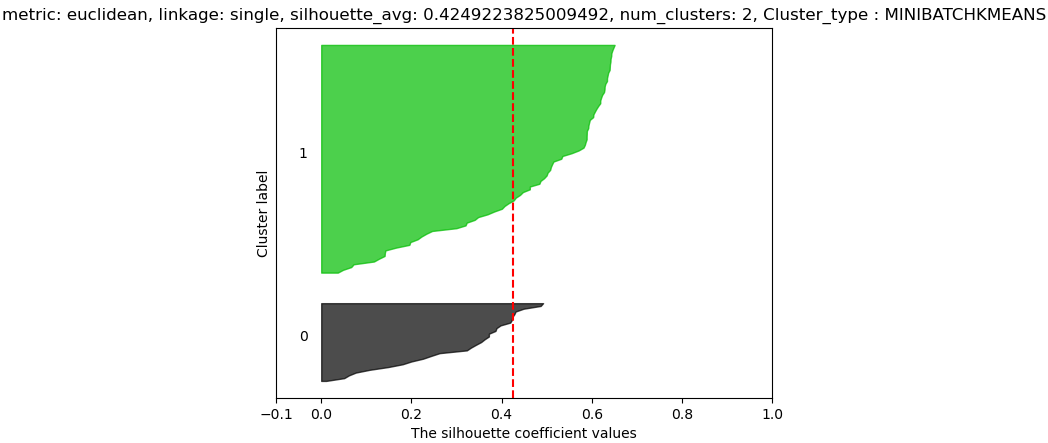
Possible answer: <https://scikit-learn.org/stable/modules/clustering.html>



Because our data distribution (image 3d projected to 2d) looks like the 3rd row above I will try minibatch kmeans,spectral clustering, meanshift, ward and gaussian mixture model clustering algos. WARD is very goo

Yep sounds good, seems like k-means is quite suitable

In file target\_eda\_across\_stocks.ipynb Tried all clustering but WARD is the best. Kmeans is second best. Updated in Key Insights file.



7) features derived from bidsize/asksize were affected by stock splits, may not be reliable as data did not account for stock splits?

In a stock split **the number of outstanding shares increases** and the price per share decreases proportionately, so stock split equally affects bidprice/askprice ??

Ahh yes, i meant like bidsize or asksize 😅not division haha, sorry for the confusion

Sorry, I did not fully understand this, perhaps we can discuss it in the meeting. 😅

8) temp\_df = temp\_df.reindex(unique\_time\_ids).ffill().bfill() ## forward and backward fill the missing values so that data is available at all time\_id

would bfill introduce look ahead bias

So when both ffill() and bfill() are used together like above ffill() is used all the time except for time\_ids that are even earlier than where data is available I have confirmed this with a toy example below. Notice only time id = 1 is backward filled others are all ffill()

Ahh okay thanks!

E,g. df = pd.DataFrame({'t':[2,6,11], 'v':[3,1,4]}).set\_index('t')

df



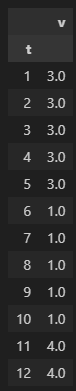
df = pd.DataFrame({'t':[2,6,11], 'v':[3,1,4]}).set\_index('t')

df

u\_t = np.array([1,2,3,4,5,6,7,8,9,10,11,12])

df = df.reindex(u\_t).ffill().bfill()

df



9) Verify that all time\_ids in train.csv and book\_train.parquet match for all the stocks

Yes they match!

Verified in data\_munging.ipynb

9) Verify that all time\_ids in train.csv and trade\_train.parquet match for all the stocks

Following do NOT match!

stock id 18 time ids do not match

missing in train\_st\_time\_ids []

missing in book\_train\_time\_ids [8524]

stock id 31 time ids do not match

missing in train\_st\_time\_ids []

missing in book\_train\_time\_ids [985, 3987, 5539, 5629, 6197, 8753, 8840, 9208, 12011, 13377, 13663, 15010, 20017, 22498, 28186, 32174]

stock id 37 time ids do not match

missing in train\_st\_time\_ids []

missing in book\_train\_time\_ids [62]

stock id 103 time ids do not match

missing in train\_st\_time\_ids []

missing in book\_train\_time\_ids [9664]

Need to use ffill()

10) Why take arctanh(C) of the correlation coefficient matrix?

ANS: fisher z transformation, it normalizes the distribution of C, make it more symmetric and stabilize the variance as correlation is bounded by [-1,1]. Hypothesis tests can be performed on normal distributions using z scores.

<https://blogs.sas.com/content/iml/2017/09/20/fishers-transformation-correlation.html>

11) ERROR from aggreagtion Code without using any for loop:

book\_wap\_log\_return\_stats\_df['wap\_max'] = book\_wap\_log\_returns\_df.groupby(['st\_id','time\_id'])['wap'].max().values

MemoryError: Unable to allocate 1.24 GiB for an array with shape (166824357,) and data type int64

Ans: Groupby is a parallel operation so If RAM is not allocated memory out of memory issues occur on laptop. Only can use a for loop to sequentially go through the stock id. This takes less memory for a single stock at a time in memory.

12) What is the physical meaning of the following code?

# filter out the extremely high and low prices of wap1\_log\_price by amplifying with postiive and negative exponential of wap1\_log\_price

# apply time weighted average to the amplified wap1\_log\_price

# what may be the physical meaning?

book\_n\_trade\_data['wap1\_log\_price\_amp\_max\_wavg'] = np.log( bucketized\_time\_weighted\_avg\_data(np.array(book\_data['seconds\_in\_bucket']),

np.array(book\_data['time\_id']),

np.exp( 4000\*np.array(book\_data['wap1\_log\_price'])),

np.ones((book\_data.shape[0])),

20, 30, ids.shape[0]) )/4000

book\_n\_trade\_data['wap1\_log\_price\_amp\_min\_wavg'] = -np.log( bucketized\_time\_weighted\_avg\_data(np.array(book\_data['seconds\_in\_bucket']),

np.array(book\_data['time\_id']),

np.exp(-4000\*np.array(book\_data['wap1\_log\_price'])),

np.ones((book\_data.shape[0])),

20, 30, ids.shape[0]) )/4000

# amplification of the difference between max and min

book\_n\_trade\_data['wavg\_wap1\_log\_price\_amp\_diff'] = np.exp(book\_n\_trade\_data['wap1\_log\_price\_amp\_max\_wavg'] - book\_n\_trade\_data['wap1\_log\_price\_amp\_min\_wavg'])

13) How to interpret higher levels of liquidity such as?

df\_book['liquidity2'] = (

df\_book['bid\_vol1']/( 1000\*(df\_book['wapq2'] - df\_book['log\_bid1']) )\*\*2

+ df\_book['bid\_vol2']/( 1000\*(df\_book['wapq2'] - df\_book['log\_bid2']) )\*\*2

+ df\_book['ask\_vol1']/( 1000\*(df\_book['wapq2'] - df\_book['log\_ask1']) )\*\*2

+ df\_book['ask\_vol2']/( 1000\*(df\_book['wapq2'] - df\_book['log\_ask2']) )\*\*2

)

df\_book['liquidity2f1'] = (

df\_book['bid\_vol1']/( 1000\*(df\_book['wap1'] - df\_book['log\_bid1']) )\*\*2

+ df\_book['ask\_vol1']/( 1000\*(df\_book['wap1'] - df\_book['log\_ask1']) )\*\*2

)

df\_book['liquidity3'] = (

df\_book['bid\_vol1']/( 1000\*(df\_book['wapq3'] - df\_book['log\_bid1']) )\*\*3

+ df\_book['bid\_vol2']/( 1000\*(df\_book['wapq3'] - df\_book['log\_bid2']) )\*\*3

- df\_book['ask\_vol1']/( 1000\*(df\_book['wapq3'] - df\_book['log\_ask1']) )\*\*3

- df\_book['ask\_vol2']/( 1000\*(df\_book['wapq3'] - df\_book['log\_ask2']) )\*\*3

)

Etc..

14) Why is mean-centering done using m + m.T and not just m alone?

# mean centering

m = np.mean(C, 1, keepdims=True)

C = C - (m + m.T)

C[a,a] = np.mean(C)

15) why is pca components scaled by square **root** of singular values and not by the square of singular values (i.e. eigenvalues)? Could it be a mistake? Principal components are eigenvectors.

scaled\_pcs = pca.components\_ \* pca.singular\_values\_[:,np.newaxis]\*\*.5

16) What is the point of clustering over principal components if they are orthogonal to each other?

Ans: Improved Cluster Separation:

In some cases, clustering on principal components can lead to better separation of clusters, as the principal components are constructed to be uncorrelated and, therefore, may highlight different aspects of the data.

17) Visualise the first 3 principal components of the scaled\_pcs to select the clustering algorithm and compare the visualization of the first 3 principal components of the scaled\_pcs with the dim. reduced UMAP of scaled\_pcs matrix.

18) why take square root of trade count?

# average of squre root of trade\_count over all buckets

final\_features['root\_trade\_count'] = np.log( np.nanmean(train\_buckets['trade\_count']\*\*.5, 2, keepdims=True))

Might be to give less weight to larger values

19) What are these features?

#

final\_features['v1liq2projt5'] = np.log( ( np.mean( liquidity2\_wavg[:,:, : 5]\*\*(1/8), 2, keepdims=True)\*\*8

/ np.mean( liquidity2\_wavg[:,:,28: ] , 2, keepdims=True) )\*\*(1/2) )

20) what is the meaning of following features?

#

final\_features['v1liq2sprojt10f25'] = np.log( np.median(

np.mean(liquidity2\_wavg[:,:,:10]\*\*.125, (2),keepdims=True)\*\*8/

np.mean(liquidity2\_wavg[:,:,25: ]\*\*.125, (2),keepdims=True)\*\*8

, 1, keepdims=True)\*\*(1/2) )

21) how does having time weighted spread and inverse spread together help? Won’t it cause multicollinearity issues?

Large spreads might take longer to close hence time weighted spreads give a lot of weight to large spread values while inverse spread normalizes spread values so both might not be as correlated (maybe), probably needs to be tested

22) In trade book why does jager not cosider the trade order\_count ? he only uses trade size.

trade\_data['trade\_volume'] = trade\_data['size']\*trade\_data['price']

Is this the line referred to

23) would not the positive and negative correlations of different clusters, k cancel out and reduce the score?

score=0

nc= np.max(p)+1

for k in range(nc):

Q = C[p==k,:][:,p==k]

score += np.mean(Q)/nc

print('SCORE', score)

23) Why are there so many transformations of trade volume feature?

* Sqrt\_trade\_volume\_buks
* Cube\_root\_volume\_buks
* volume\_p2/3\_buks
* quart\_root\_volume\_buks

Possible Ans: Transformation of Volume maybe due to square root law of the market impact??

<https://www.reddit.com/r/algotrading/comments/kuupuz/square_root_law_of_the_market_impact_simplest/>

24) Why use agglomerative hierarchical clustering and not K-means?

* does not assume that clusters have a spherical shape or are of equal size, unlike K-means. K-means is sensitive to the initial placement of centroids and may not perform well when clusters have irregular shapes or different sizes.
* does not require specifying the number of clusters in advance, whereas K-means does
* useful for exploring the relationships between different levels of clustering. This hierarchical structure allows you to see how smaller clusters are grouped into larger ones.
* Agglomerative hierarchical clustering is generally more robust to outliers than K-means. Outliers in K-means can significantly affect the positions of centroids, leading to suboptimal clustering.
* Agglomerative hierarchical clustering allows for the use of various distance metrics

25) What type of linkage would be ideal after looking at the dendrogram? Should clusters have an equal number of stocks in them or can they be skewed?

* Equal Size Clusters: If your application requires clusters of approximately equal size, you might prefer linkage methods like Ward's that tend to produce more balanced clusters.
* Unequal Size Clusters: In some cases, the natural structure of the data may result in clusters of different sizes. Single or complete linkage may be more appropriate if you are interested in capturing elongated or irregularly shaped clusters.