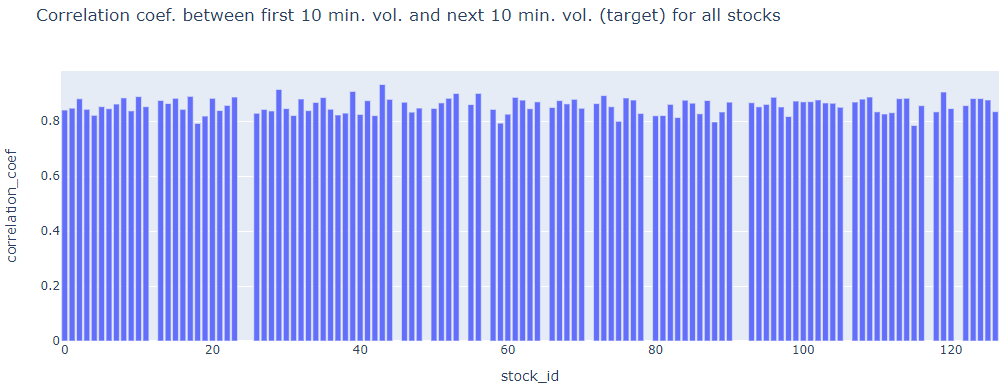
# **Key Insights that are important**

Record Key Insights that cannot be missed!

## 1) Correlation between real. Vol in first 10 mins and next 10 mins (target) for all stocks

****

## past 10 min real vol is a good predictor of future 10 min real vol.

##### **Check correlation between**

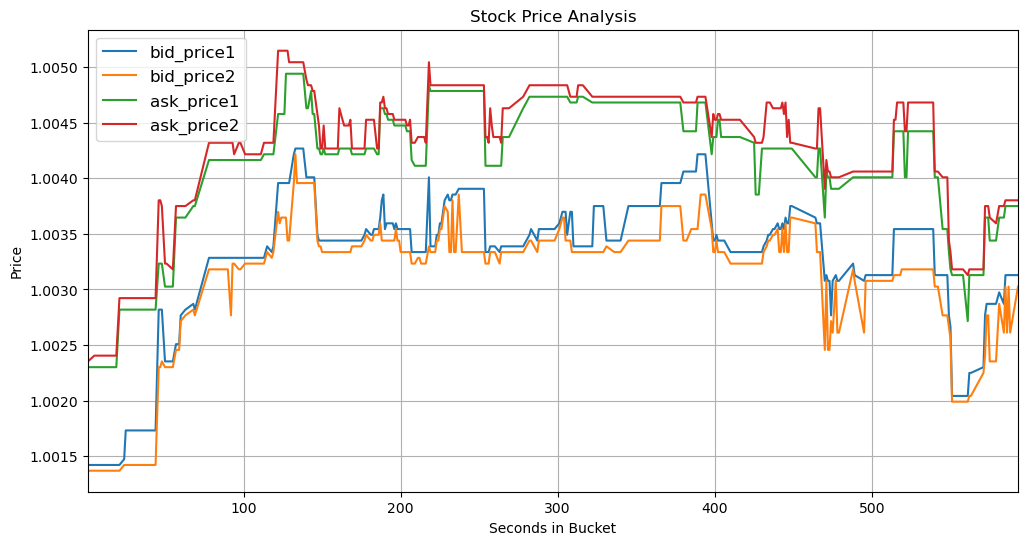
##### **i) target (2nd 10 mins vol.) and trade execution stock price at the available times.**

Aggregate **trade execution price using the log returns and realised volatility formulas for each time id. Plot scatterplot this against the corresponding target for all time id.**

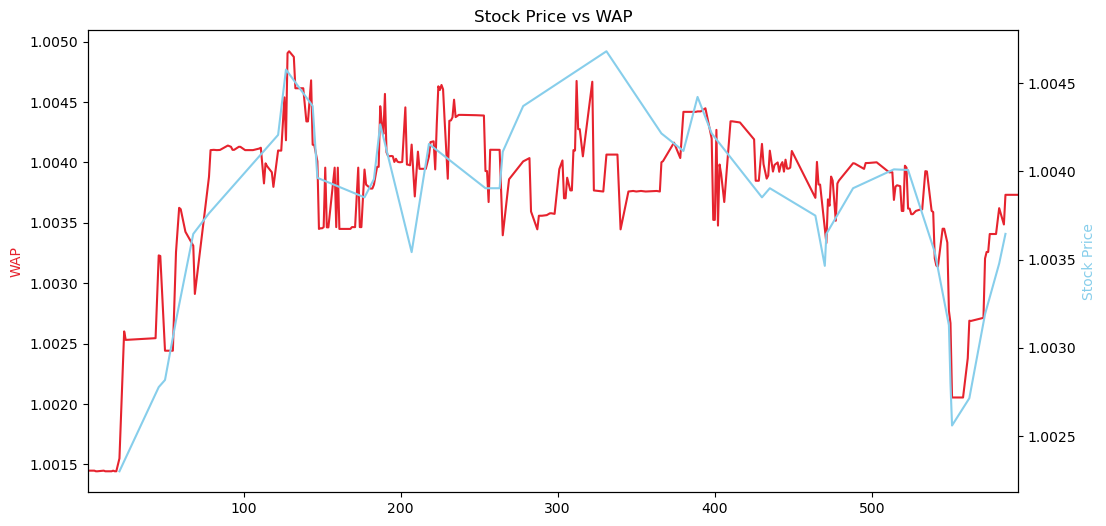
****

****

## 2) Bid Price VS Ask Price (For Stock 0 at time id 5):

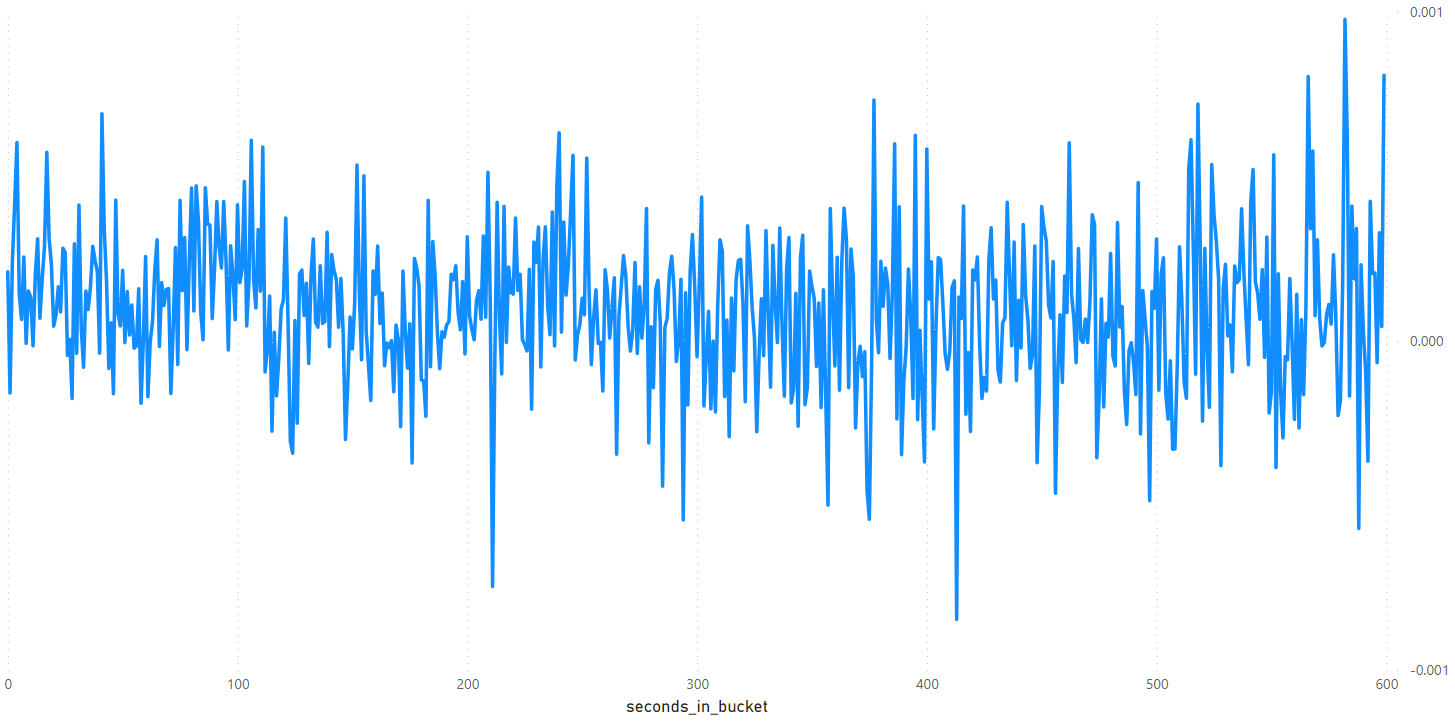
****

## 3) Stock Price VS WAP (For Stock 0 at time id 5):



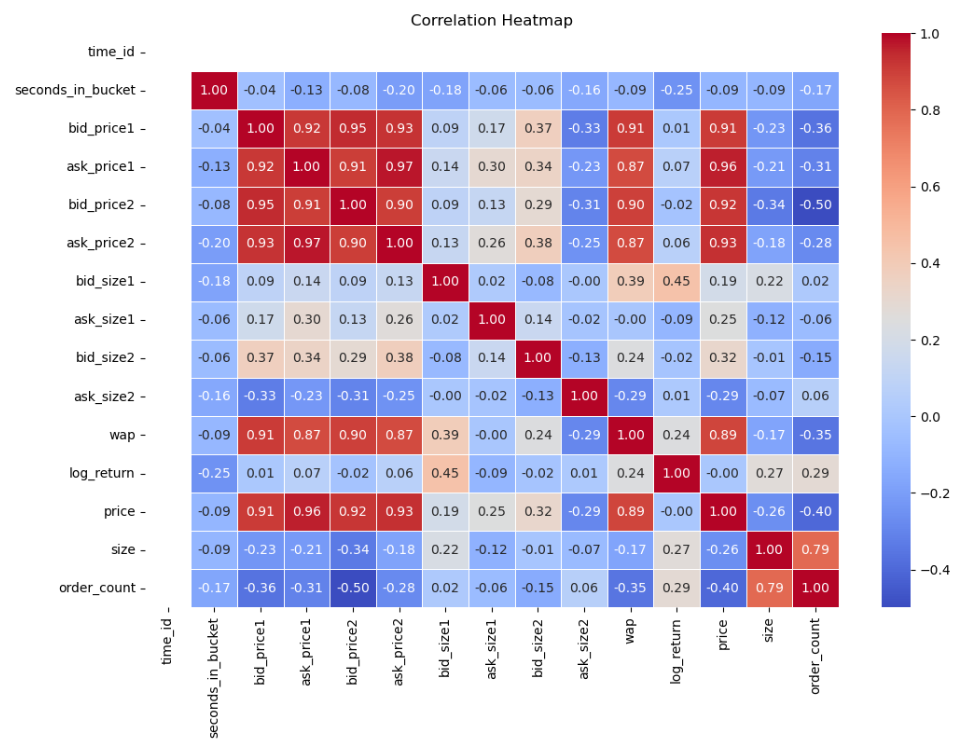
* Bid price and ask price form the lower and upper limit for WAP. WAP is correlated with the actual trade execution price (stock price) as shown above. Could we say that high volatilities in calculated WAP could indicate a high volatility in stock price?

## 4) Difference between Stock Price change and WAP for stock id 0 for aggregated all time id (Please verify this):



This represents the mean difference between the aggregated average of WAP across all time IDs and the price for stock ID 0. As we, can observe there is not much difference between both trends so we can safely say that the **correlation between WAP and Stock price (from trade data) is High.**

## 5) Correlation Heat Map (For Stock 0 at time id 5):



The correlation heatmap presented above reveals that our stock demonstrates a robust positive correlation with [bid\_price1, ask\_price1, bid\_price2, ask\_price2, wap], with an approximate 90% correlation coefficient. Conversely, there is a notable negative correlation of around 40% with the [order\_count] variable.

## **6) Target realized volatility clustering analysis**

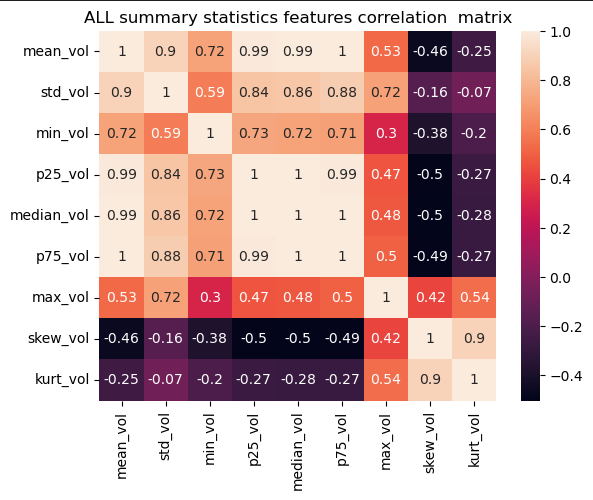
**Analysis file :** target\_eda\_across\_stocks.ipynb

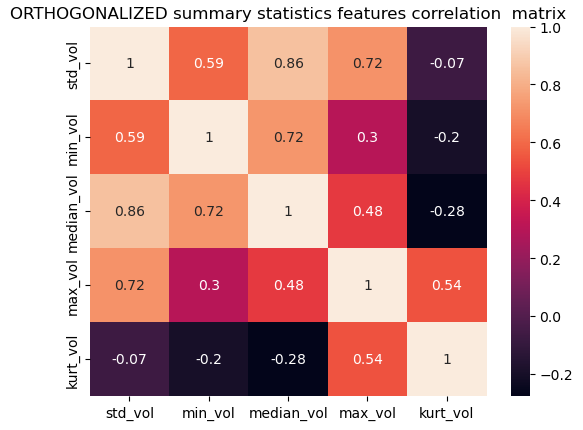
**Objective:** Group stocks that are similar to each other. The stocks that are similar can help to predict each other when used as covariates.

**Clustering based on :**

1. Target realized volatility Summary statistical features (i.e. Distributions) : [mean\_vol,std\_vol,min\_vol,p25\_vol,median\_vol,p75\_vol,max\_vol]
2. Temporal target realized volatility correlation : pearson correlation between stocks for all time\_ids.

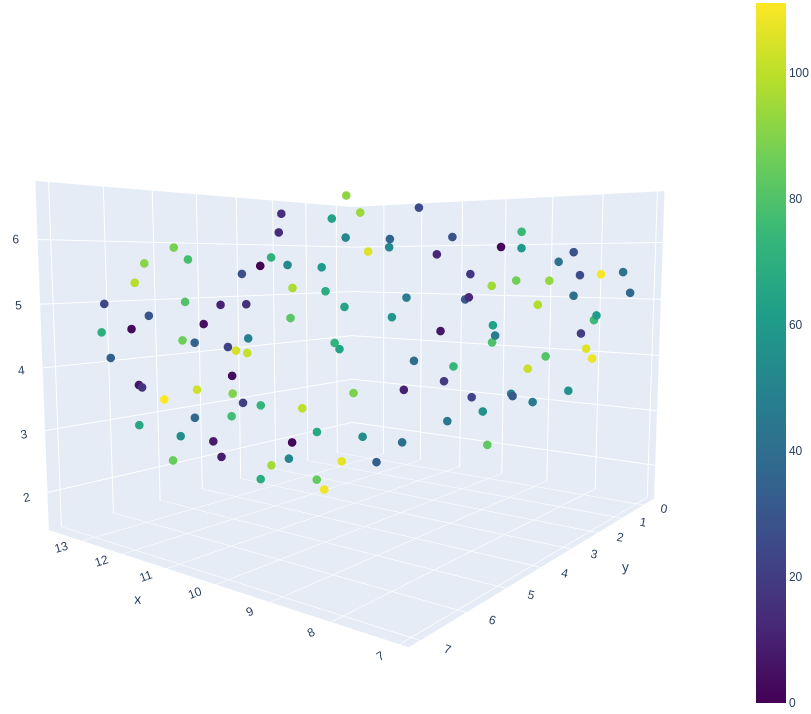
**a) Clustering on Summary statistical features**

1. Perform feature orthogonalization by removing correlated features 



For large matrices PCA can also be used for feature orthogonolizaiton at the cost of losing intrepretatvility of principal components

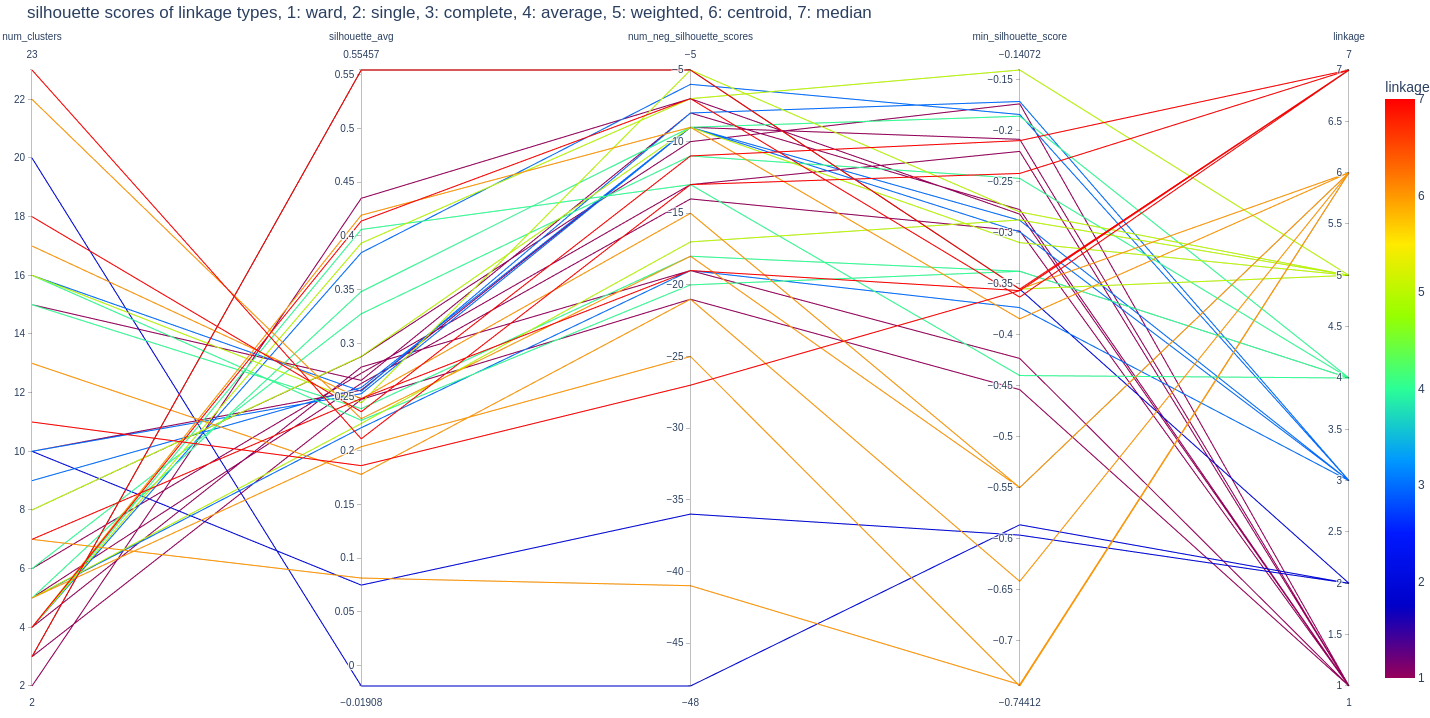
1. Perform Dimensionality reduction to visualise how data is distributed in order to choose the appropriate clustering algorithm



Which clustering algo. is better? K-means or Agglo. Hier. clustering?

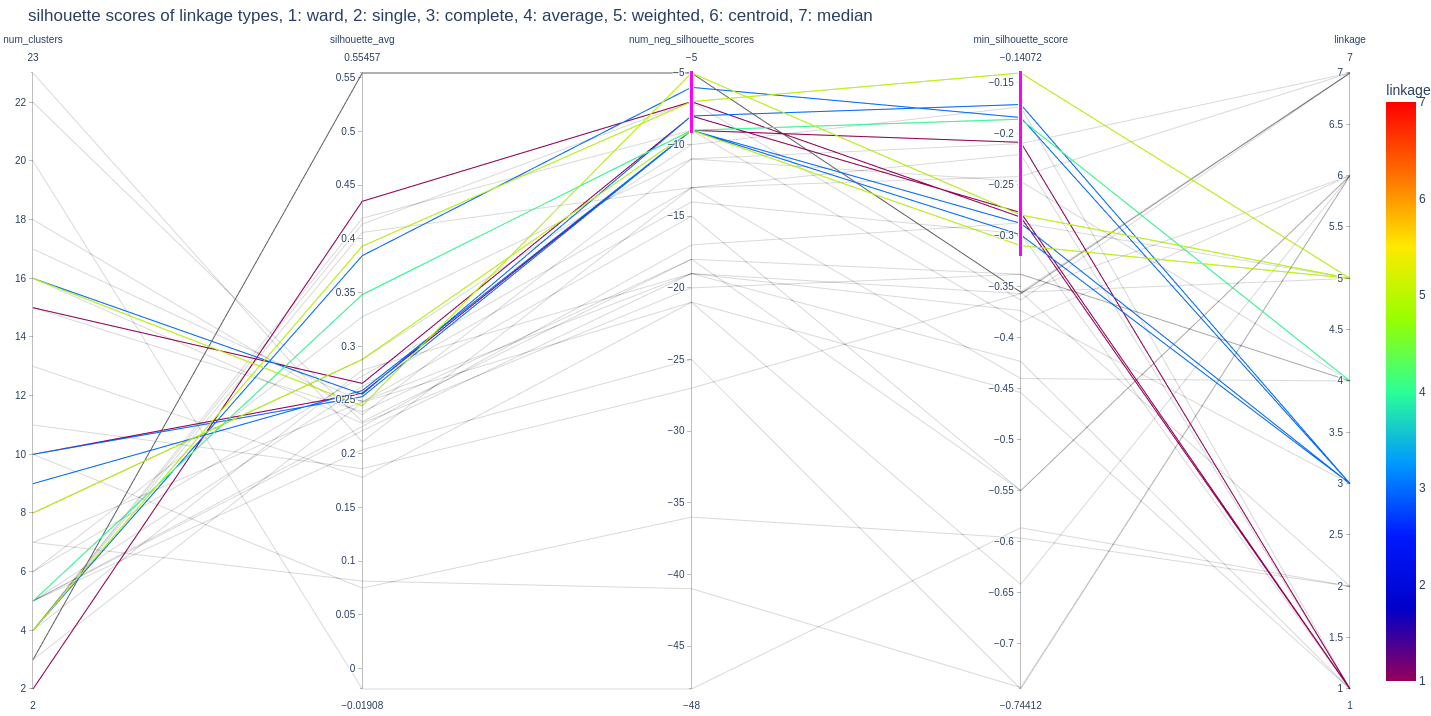
Ans: Ultimately, silhouette scores decide but we can look at the data distribution as well and make an educated guess. Smaller dataset (that have non-spherical data distributions) Hier. clustering is better than k-means. k-means is faster on larger datasets. K-means if number of clusters is known. K-means works better if clusters are well linearly separated.

1. Perform clustering using various clustering algorithms and hyperparameters and check which one maximises the silhouette average score.



Select based on high average sil. Score and fewer number of negative sil. Score data points, select smaller minimum silhouette score.

Those stocks that have negative silhouette scores are like outliers and they can each be in their own cluster.



Chosen number of clusters: 2,4,5,8,9,10,15,16

All distances are euclidean.

Cluster parameters:

Num\_clusters = 2, linkage = ward, threshold =2.5

Num\_clusters = 15, linkage = ward,threshold = 0.5

Num\_clusters = 10, linkage = ward,threshold = 0.7

Num\_clusters = 15, linkage = ward,threshold = 0.5

Num\_clusters = 4, linkage = complete,threshold = 1

Num\_clusters = 9, linkage = complete,threshold = 0.6

Num\_clusters = 10, linkage = complete,threshold = 0.5

Num\_clusters = 16, linkage = complete,threshold = 0.4

Num\_clusters = 30, linkage = complete,threshold = 0.25

Num\_clusters = 5, linkage = average,threshold = 0.5

Num\_clusters = 20, linkage = average,threshold = 0.25

Num\_clusters = 3, linkage =weighted,threshold = 0.7

Num\_clusters = 4, linkage =weighted,threshold = 0.6

Num\_clusters = 8, linkage =weighted,threshold = 0.4

Num\_clusters = 16, linkage =weighted,threshold = 0.3

Num\_clusters = 3, linkage =centroid,threshold = 0.7

Num\_clusters = 4, linkage =centroid,threshold = 0.6

Num\_clusters = 3, linkage =median,threshold = 0.7

Num\_clusters = 4, linkage = median,threshold = 0.6

Further shortlist from above.

Just choose any one color from each color. Others can be ignored.

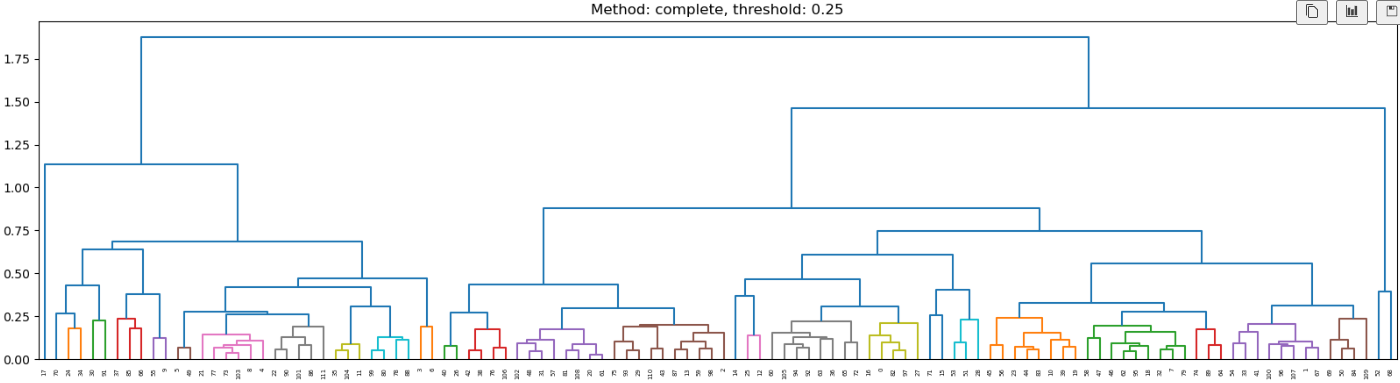
We select:

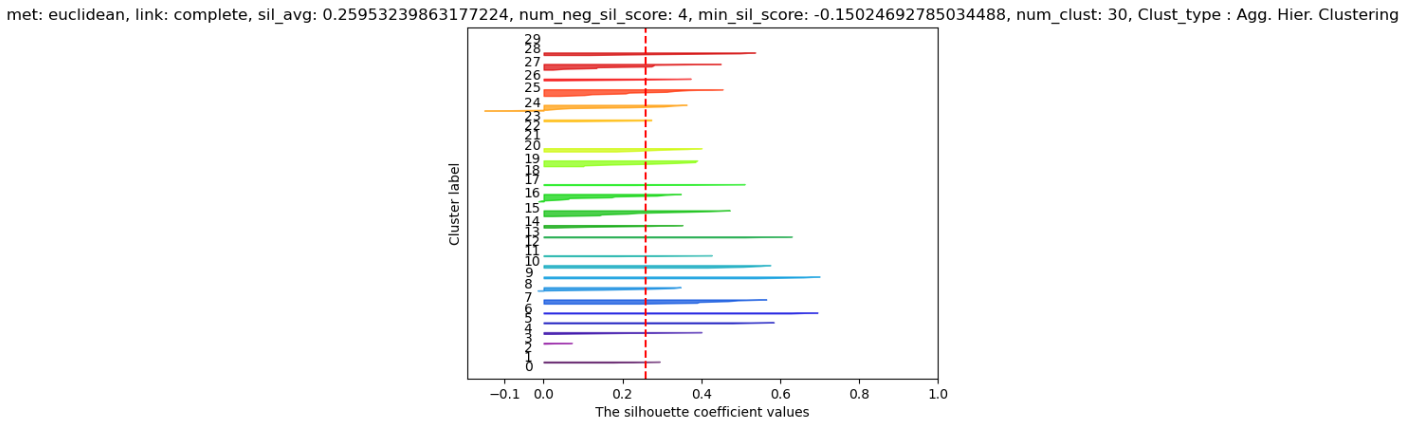
Num\_clusters = 4, linkage = complete,threshold = 1

Num\_clusters = 10, linkage = complete,threshold = 0.5

Num\_clusters = 16, linkage = complete,threshold = 0.4

Num\_clusters = 30, linkage = complete,threshold = 0.25





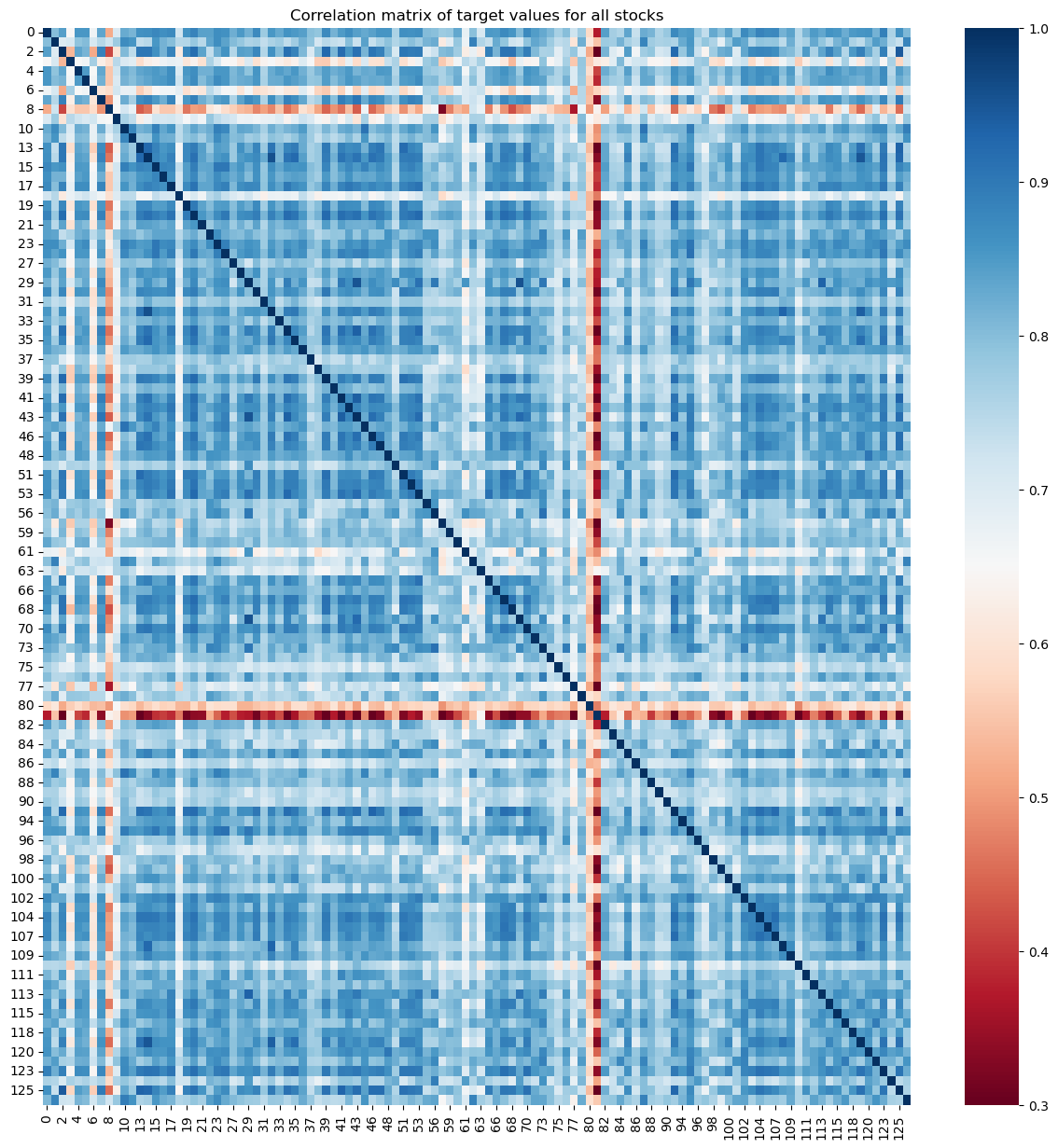
Note that the stock numbers shown in dendrogram is not actual stock number but just going from 0 to 111.

**use this to cluster first\_10\_min\_vol\_df feature using cluster\_agg( ) function**

**b) Clustering on Temporal target realized volatility correlation**

**Reference:** <https://www.kaggle.com/code/sgalella/correlation-heatmaps-with-hierarchical-clustering>

1. Find ALL the unique time\_ids from all stocks. Create a new target dataframe (train\_common\_time\_ids\_df ) with all these time\_ids for all the stocks. Use forward and backward fills to populate missing values.
2. Find correlation matrix of this train\_common\_time\_ids\_df
3. Plot the (unclustered /ordered stocks) correlation matrix



No negative or zero correlation. Correlation starts at 0.3.

1. Check that there is no NEGATIVELY correlated stock pairs.

If there are negatively correlated stock pairs they need to be SEPARATELY clustered from the positively correlated stock pairs.

In this dataset all are positively correlated. (usually the case if the time interval is long enough.)

1. For each stock sort the most correlated stocks. This can help to validate clusterings formed later on.
2. Perform agglomerative hierarchical clustering to cluster similar stocks based on pearson correlation. Stocks that realised volatility that move together are clustered together.

The linkage function takes the pairwise correlations between stocks. I.e. (N\*(N-1))/2 entries in the upper triangular part of the correlation matrix.

The methods used like single/complete/ward etc. are different ways to combine the indiv. stocks and then clusters of stocks until there is only 1 cluster left. <https://www.youtube.com/watch?v=8QCBl-xdeZI>

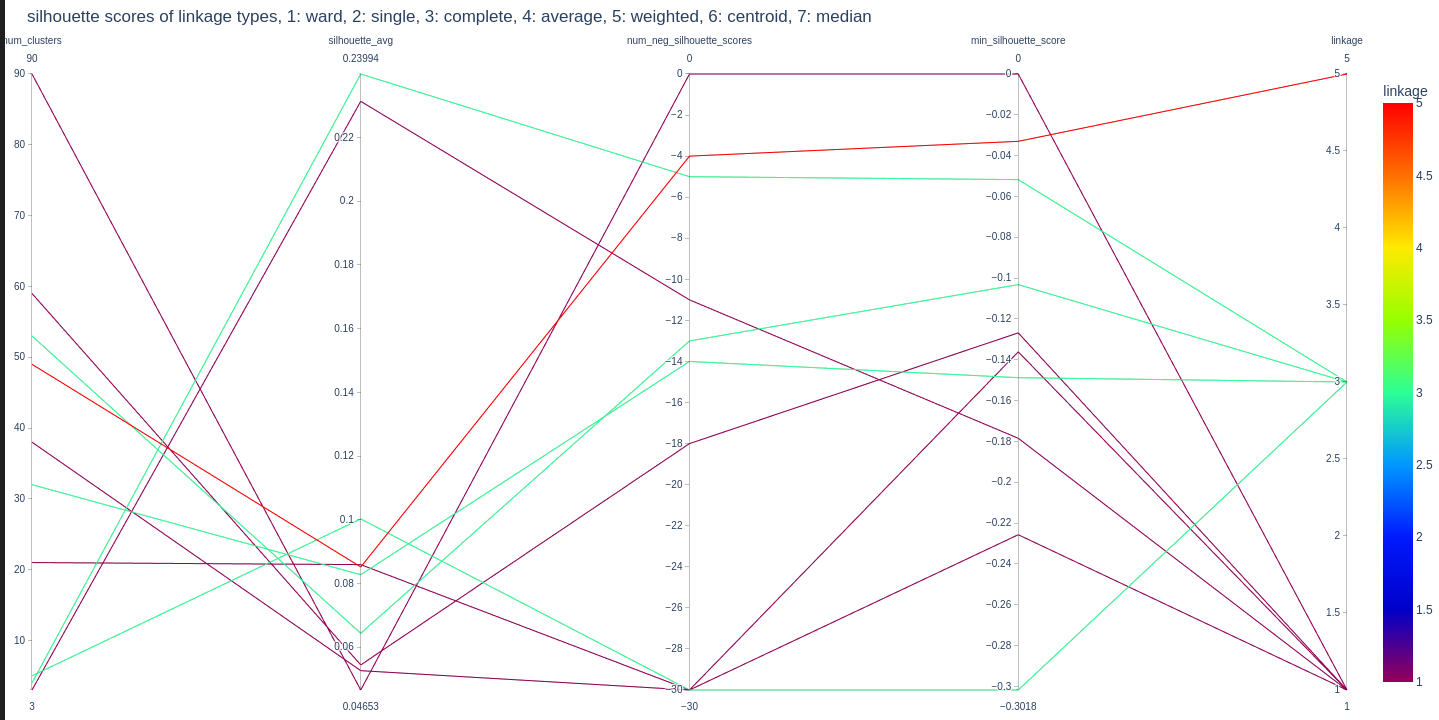
<https://www.youtube.com/watch?v=vg1w5ZUF5lA>,

This is shown on dendrogram. The y-axis is the distance we provided (i.e. 1 - pearson\_correlation ) for methods like single and complete BUT not for all methods e.g. average.

**Best methods: 1)** Ward, 2)Complete

Ward method is the best because it creates more compact clusters due to centroid formation and clusters combine based on new centroid formations. Other methods combining may note be good because they are combining based on edge point of the cluster alone whereas ward combines based on centroid which is better representative of the cluster than a single point i..e edge point.

WARD method: Try this first and see if prediction accuracy is good.



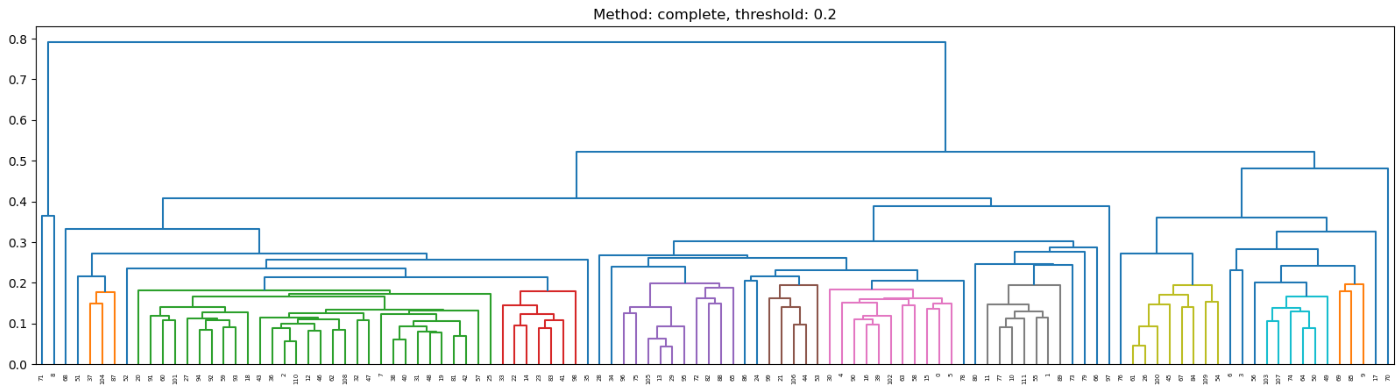
Cluster parameters:

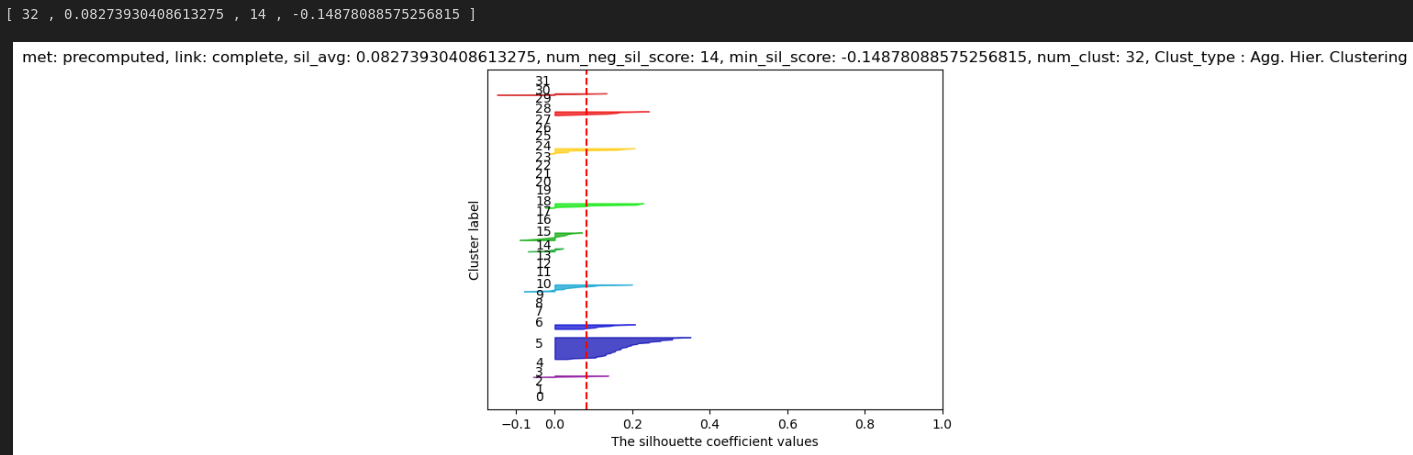
Num\_clusters = 4, linkage = complete, threshold = 0.45

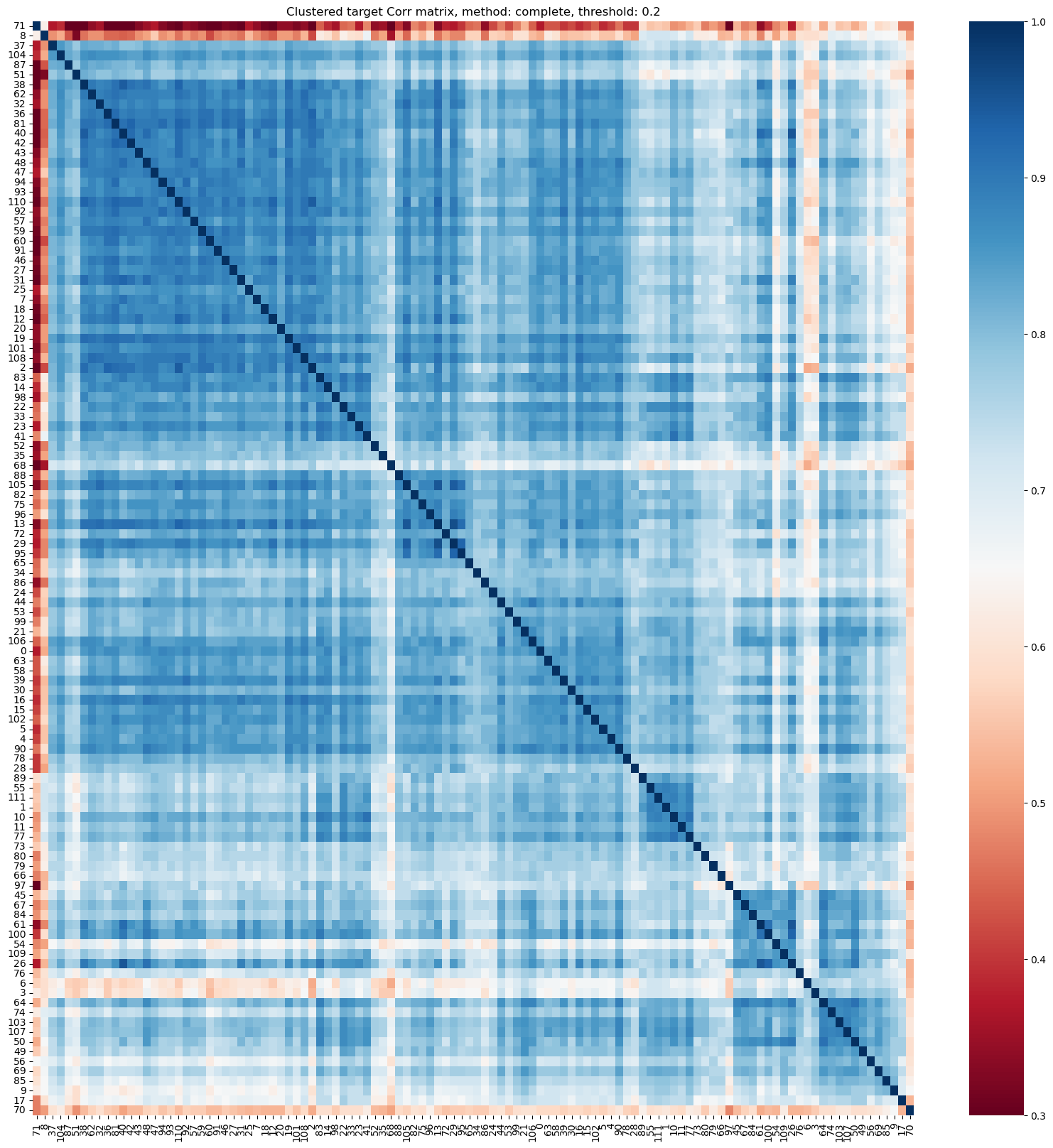
Num\_clusters = 32, linkage = complete, threshold = 0.2

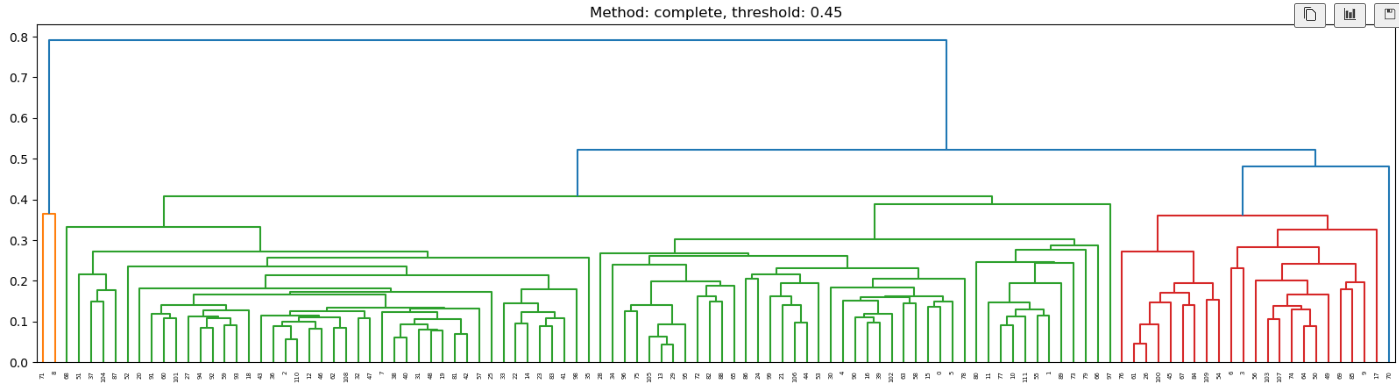
Num\_clusters =49, linkage = weighted, threshold = 0.15

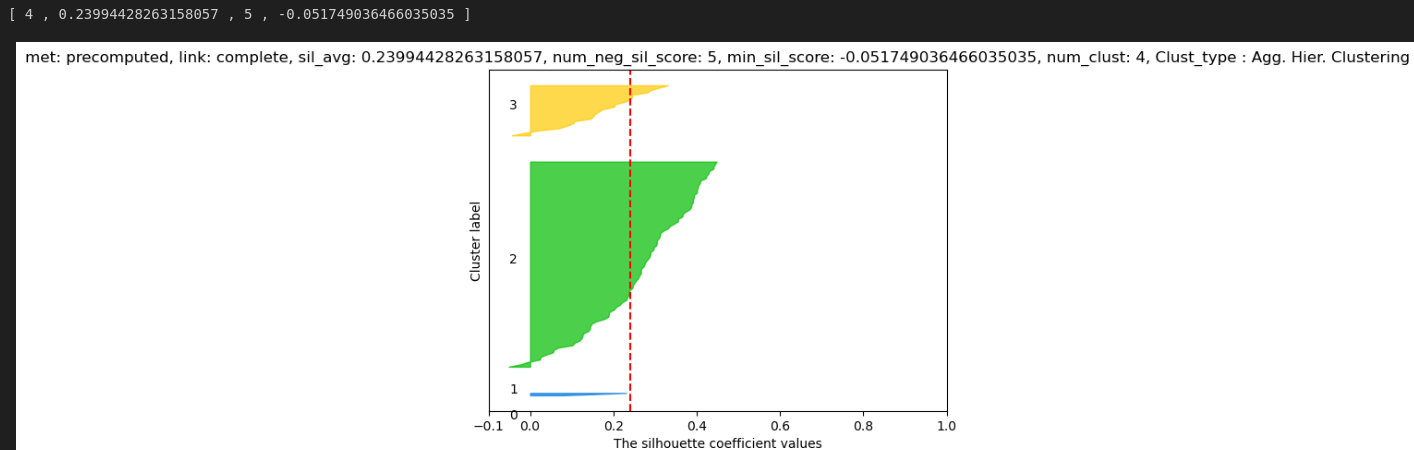
Num\_clusters =90, linkage = ward, threshold = 0.1

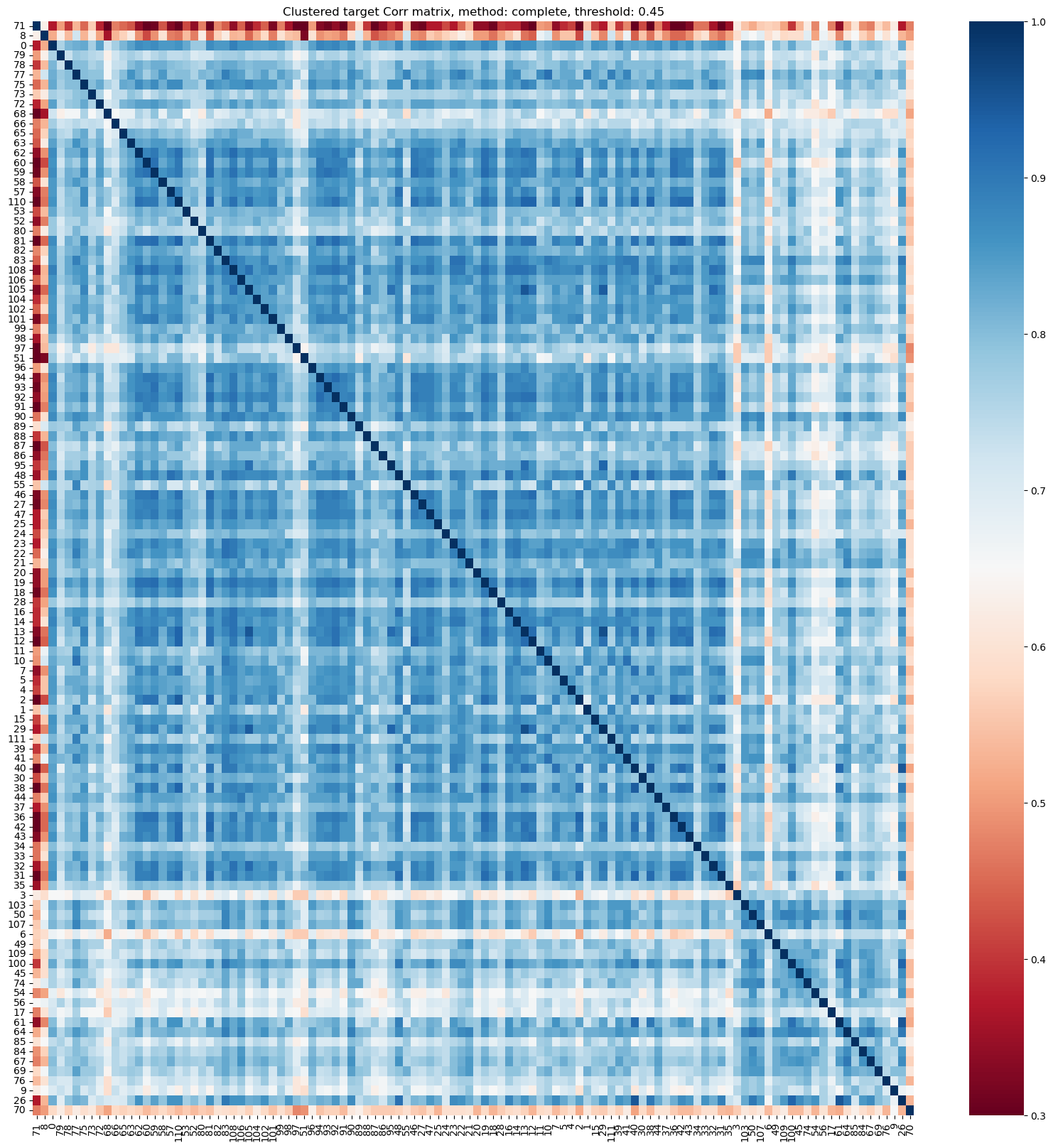


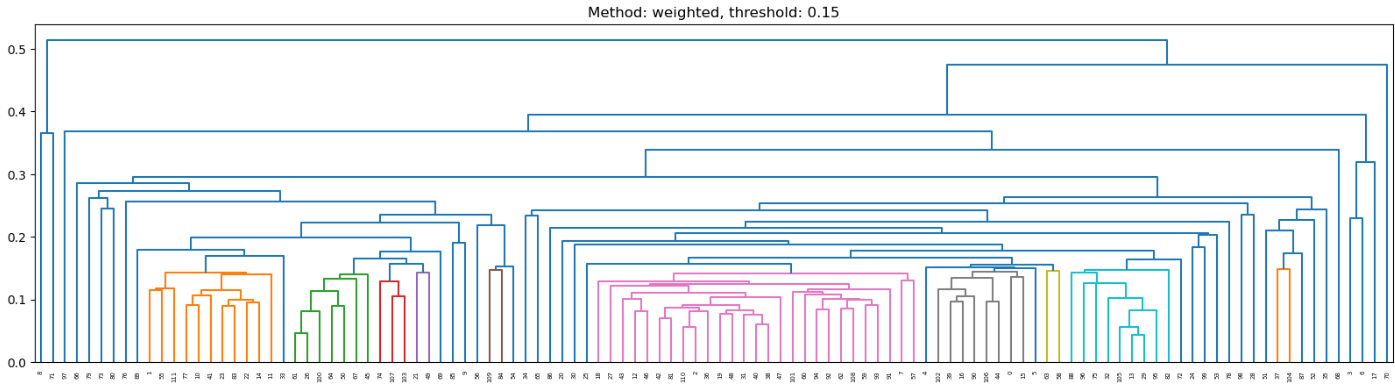


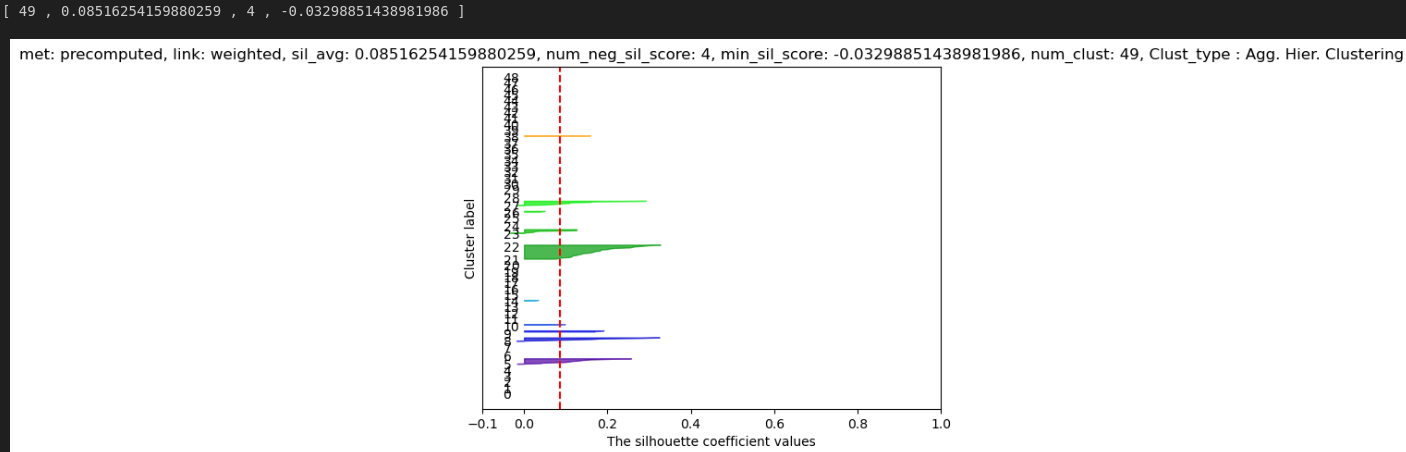


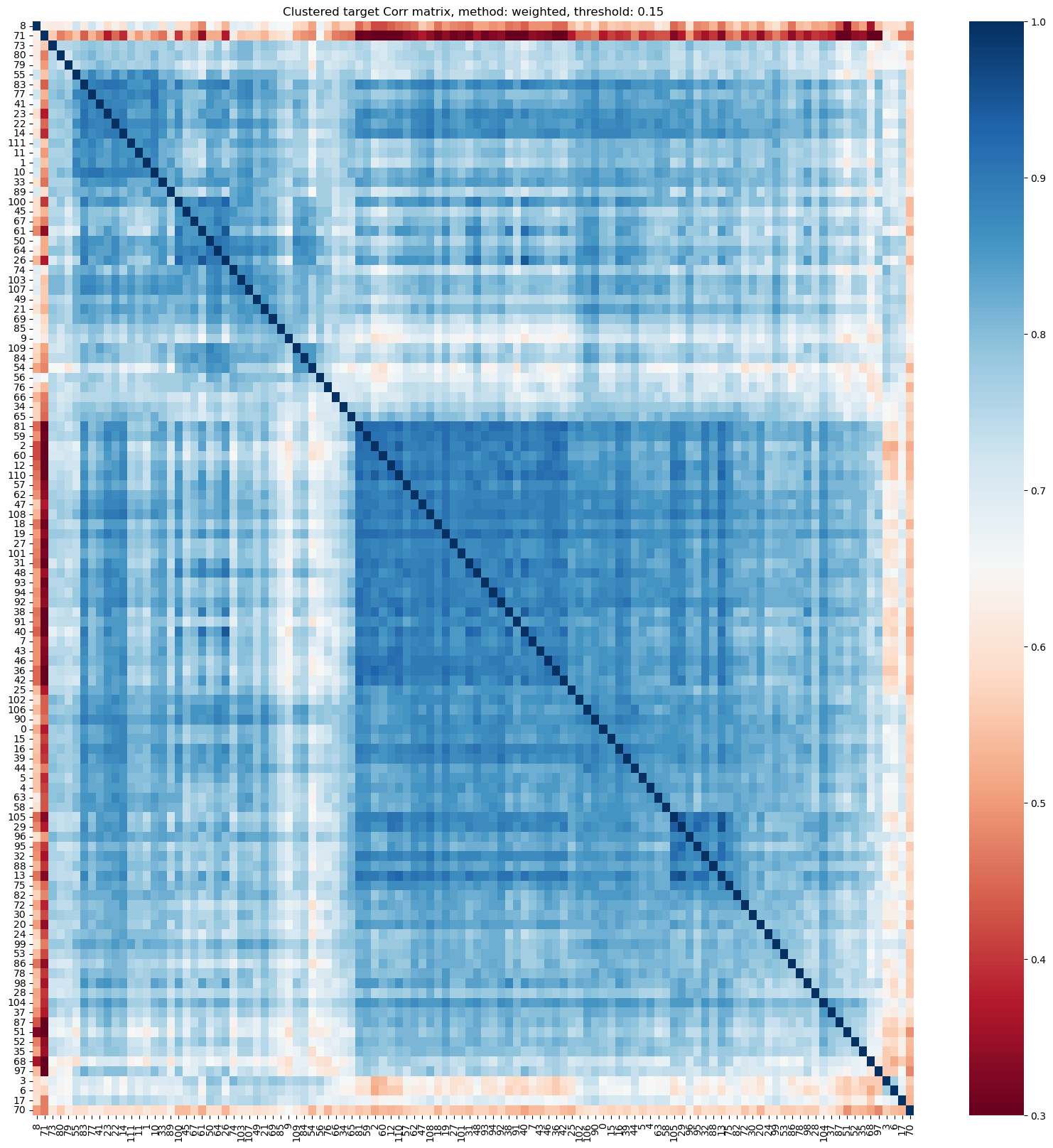


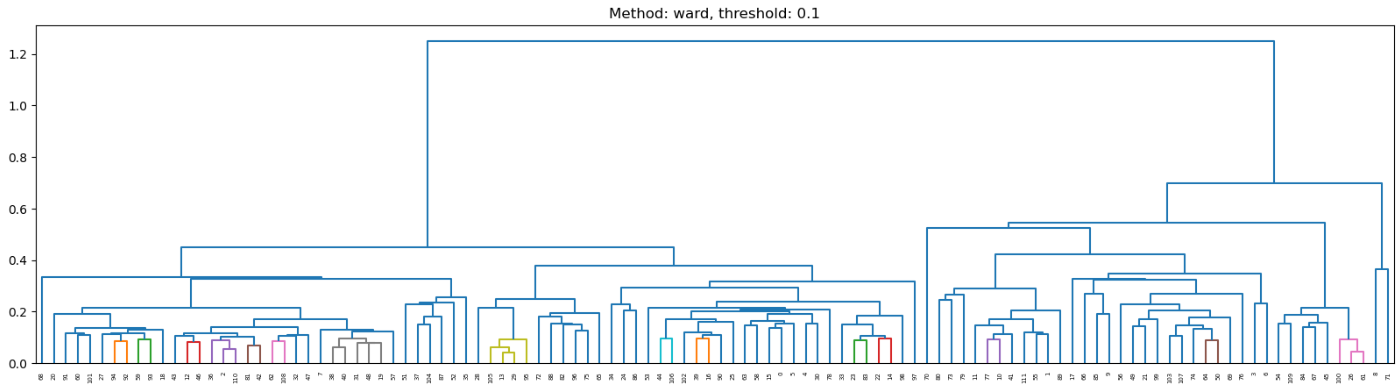


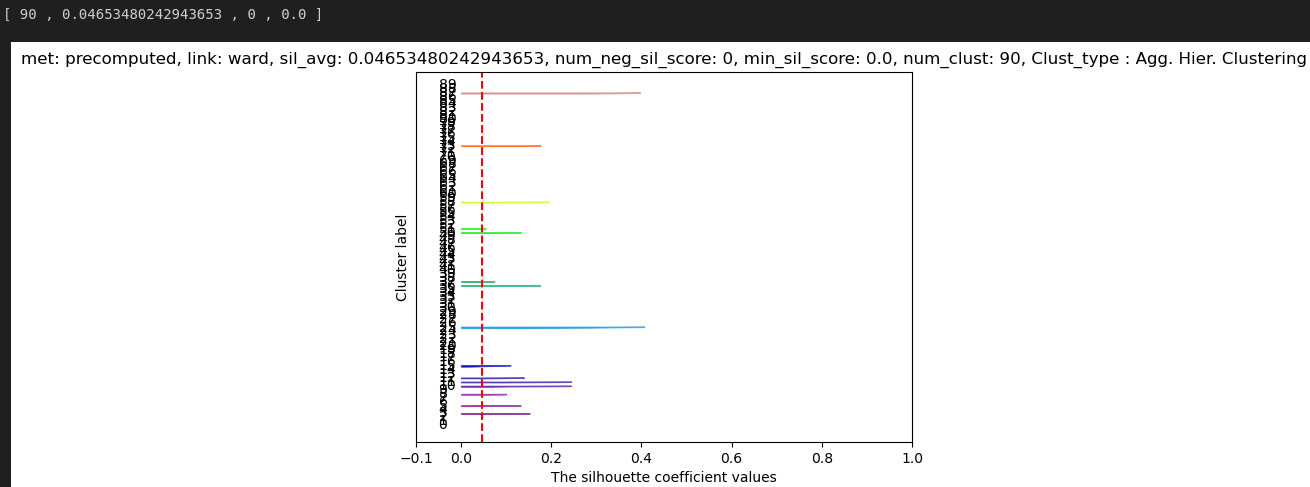


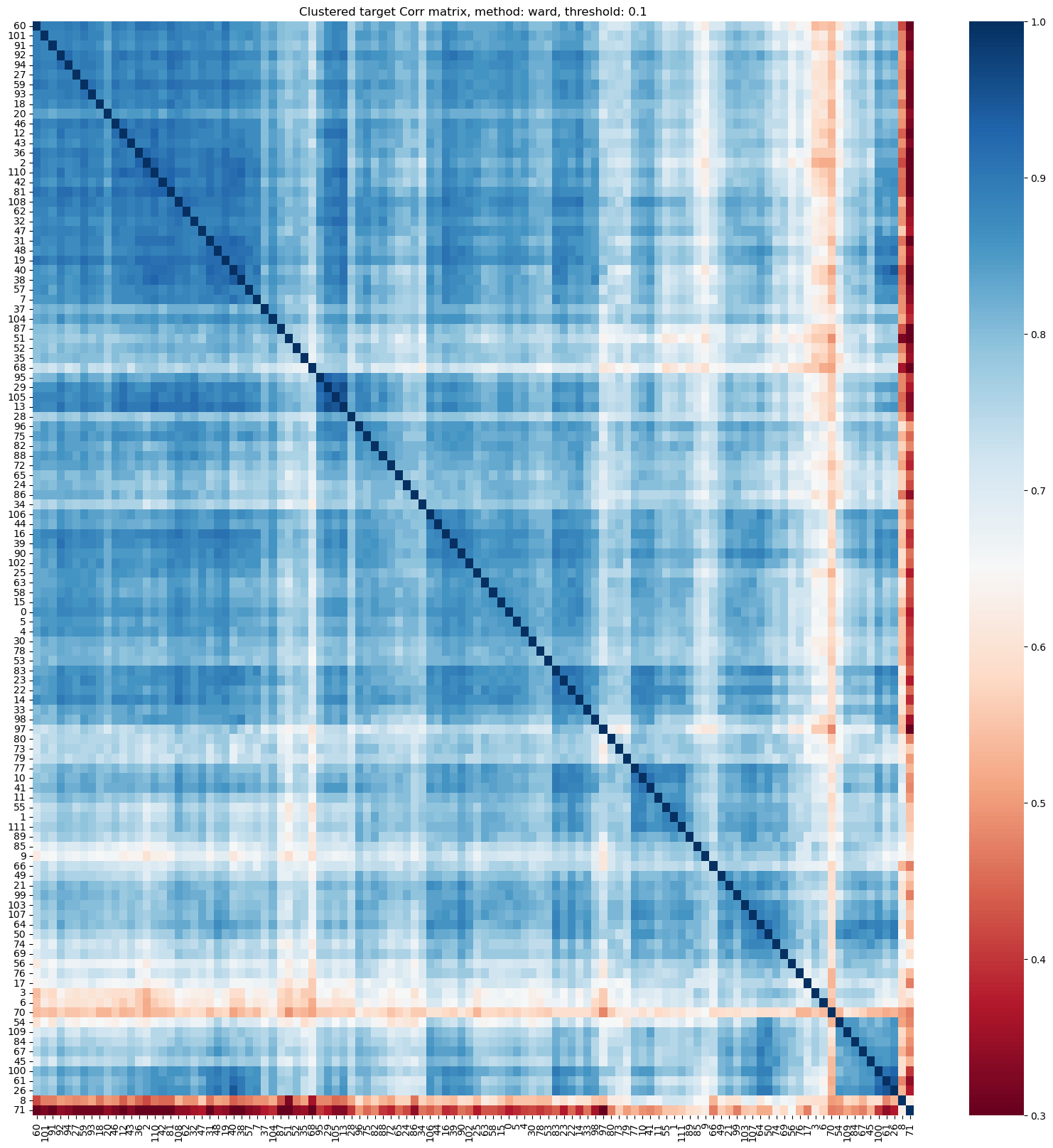












Look at the diagonal line. The square shaped clusters along the diagonal represent the groups/clusters of correlated stocks.

SKew and kurtosis of prices and log transformed prices







For all graphs, log transformation decreases positive skew and increases negative skew. THis can be observed above. For positive skew, log transformation decreases kurtosis (with one exception for stock 16). For negative skew, log transformation increases kurtosis.

Most stock’s Bid price is left skewed and ask price is right skewed.

Log helps to reduce right skewed stocks but also increase skew of left skewed stocks.

Negative skewed stocks fatness (kurtosis) decreased as points became shifted further to the left by log transformation and positive skewed stocks fatness (kurtosis) increased. This not entirely true. For smaller positive skewed stocks, kurtosis does decrease with log transformation. Check liquidity\_features\_inux.ipynb for bigger figure.

**LIquidity Feature Descriptions**

***Background information***

* Number of buckets = 30
* Bucket width = 20 seconds
* Spread = log ask price1 - log bid price1
* Spread2 = log ask price2 - log bid price2
* Book size = book volume1 + book volume 2
* Book Volume = ask size \* ask price + bid size\* bid price
* Trade volume = trade size \* trade price
* Suffix with \_**buks** stands for simply summed over within the bucket.
* Suffix with \_**wavg** stands for time weighted average within the bucket. Weighted by amount of time spent till the next time point. Instead of simply summing like above, we give more weight to more persistent points.

Time weighted = x1\*(t2 - t1) + x2(t3 - t2) + … / ( (t2 - t1) + (t3 - t2) )

* **wap1\_log\_price**: is wap1 computed using the LOG of bid and ask price in the WAP formula.
* **wap2\_log\_price\_ret\_changes\_n\_wap1\_log\_price\_ret\_constant**: # this indicates the changes in level 2 wap when level 1 wap does NOT change # This happens when all orders in level 1 are filled and new orders are placed in level 2 # indication of liquidity as prices in level 2 are moving towards level 1 # Aggressive Market Orders, Imbalance in Market Depth, Execution of Large Orders, Liquidity Changes:
* Suffix with \_**vol** stands for volatility: WHich is some form of aggregation (e.g. sum of absolutes or square root of sum of squares) of returns. In volatility calculations we assume that the quantity has zero mean because taking log of the quantity centers it around the mean usually. So simply squaring and then summing results in the variance formula. Although it does not divide by number of terms.
* **\_abs\_vol\_** stands for simple sum of returns within a bucket.
* **\_sqr\_vol\_** stands for square root sum of squared returns within a bucket.
* NOte: The definition of log returns is different here compared to the competition log return.

Competition formula:

log( as1\*bp1 + bs1\*ap1 /as1+bs1)\_t2 - log( as1\*bp1 + bs1\*ap1 /as1+bs1)\_t1 = log(s\_t2/s\_t1) = return

Customized formula:

Wap1\_log\_price\_ret = ( ( as1\*log(bp1) + bs1\*log(ap1) ) /as1+bs1 )\_t2 - ( ( as1\*log(bp1) + bs1\*log(ap1) ) /as1+bs1 )\_t1

* Find (a few different) equilibrium prices between bid and ask price where trades are likely to happen. The different prices are the result of levels 0,1, and 2. # Find equilibrium price at which trades are likely to happen # This price minimizes the missing total volume from buy and sell side.

book\_data[**'wap\_eqi\_price0'**] = find\_equilibrium\_price( book\_data, lvl=0)

book\_data[**'wap\_eqi\_price1'**] = find\_equilibrium\_price( book\_data, lvl=1)

book\_data[**'wap\_eqi\_price2'**] = find\_equilibrium\_price( book\_data, lvl=2)

* **Wap\_eqi\_price0\_ret**: returns (first difference) of **'wap\_eqi\_price0'**
* **Wap1\_log\_price\_ret\_pos\_log\_liq\_ret**: the **wap1\_log\_price\_ret** when **log\_liquidity1\_ret** is positive
* **Wap1\_log\_price\_ret\_neg\_log\_liq\_ret** : the **wap1\_log\_price\_ret** when **log\_liquidity1\_ret** is negative
* **Log\_liquidity1\_ret**: the returns (first difference ) **log\_liquidity1**
* **Log\_liquidity1** : log of **liquidity1**
* **liquidity1**: It is proportional to volume = size\*price (not log price). And inversely proportional to distance from equilibrium price. HIgher volume = higher liquidity. Farther from equilibrium price lower the liquidity.

book\_data[**'liquidity1'**] = (

book\_data['bid\_volume1']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_bid\_price1']) )

+ book\_data['bid\_volume2']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_bid\_price2']) )

- book\_data['ask\_volume1']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_ask\_price1']) )

- book\_data['ask\_volume2']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_ask\_price2']) )

)

* **Wap1\_log\_price\_amp\_max\_wavg** :
* **Wap1\_log\_price\_amp\_min\_wavg :**

# filter out the extremely high and low prices of wap1\_log\_price by multiplying with a large number like 4000 and amplifying with postiive and negative exponential of wap1\_log\_price. Taking log and -log undoes the positive and negative exponentiation.

# 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

* **Ask\_liq1\_diff**: # difference between ask's level 1 and level 2 liquidity

book\_data['ask\_liq1\_diff'] = (

book\_data['ask\_volume1']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_ask\_price1']) )\*\*1

- book\_data['ask\_volume2']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_ask\_price2']) )\*\*1

)

* **Bid\_liq1\_diff**: # difference between bid's level 1 and level 2 liquidity

book\_data['bid\_liq1\_diff'] = (

book\_data['bid\_volume1']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_bid\_price1']) )\*\*1

- book\_data['bid\_volume2']/( 1000\*(book\_data['wap\_eqi\_price1'] - book\_data['log\_bid\_price2']) )\*\*1

)

***Total 46 liquidity*** ***features***

1. **'wap1\_log\_price\_ret\_buks'** : bucketized wap1\_log\_price\_ret, where wap1\_log\_price\_ret is returns (first difference ) of wap1 computed using the LOG of bid and ask price in the WAP formula. Tells whether there was a net increase/decrease in WAP within a bucket. # Amount of wap1 price movements in a time bucket of 30 seconds,
2. **'Wap1\_log\_price\_ret\_abs\_vol\_buks'**: # Amount of sum of absolute wap1 price movements in a time bucket of 30 seconds, i.e. absolute wap1 returns volatility in bucket. Measure amount of price activity regardless of prices going up or down. It’s like volatility.
3. **'Wap2\_log\_price\_ret\_abs\_vol\_buks'**: same as above for using level 2 for wap2.
4. **'Wap1\_log\_price\_ret\_sqr\_vol\_buks'**: same as above but it’s squared, gives larger weight to larger price changes. Measure amount of (large) price activity regardless of prices going up or down.
5. **'Wap2\_log\_price\_ret\_sqr\_vol\_buks'**: same as above for using level 2 for wap2.
6. **'Wap1\_log\_price\_ret\_vol\_buks'**: volatility of returns within a bucket. Square root of Sum of squared returns.
7. **'wap2\_log\_price\_ret\_vol\_buks'**, same as above using level 2 price, wap2.
8. **‘wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol\_buks'**: see above. bucketized Volatility of wap2 returns when wap1 returns are 0.
9. **'Wap\_eqi\_price0\_ret\_abs\_vol\_buks'**: # equilibrium price returns absolute volatility in bucket
10. **'Wap\_eqi\_price0\_ret\_sqr\_vol\_buks'**: same as above but volatility calculated using square root of squared returns
11. **'Wap\_eqi\_price1\_ret\_sqr\_vol\_buks'**: same as above but using eqi\_price\_1
12. **'Wap1\_log\_price\_ret\_pos\_log\_liq\_ret\_sqr\_vol\_buks'**: the **wap1\_log\_price\_ret** when **log\_liquidity\_ret** is positive i.e. increasing liquidity.
13. **'Wap1\_log\_price\_ret\_neg\_log\_liq\_ret\_sqr\_vol\_buks'**: the same as above but when **log\_liquidity\_ret** is negative i.e. decreasing liquidity.
14. **‘log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_buks'**: product of **‘log\_liq2\_ret\_** and **\_wap\_eqi\_price1\_ret\_vol\_buks'** squared. It measures large movements in log\_liquidity and wap\_eqi\_price1 regardless of the directions.
15. **'exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_buks**': # wap equilibrium price 1 returns volatitlity in bucket amplified (> 1) by positive/increasing liquidity returns (through exponent) # and diminished ( < 1) by negative/decreasing liquidity returns (through exponent). It checks whether wap\_eqi\_price1 is high or low during positive log liq1 ret. It ignores when it is negative. The final term is squared.
16. **'exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_buks\_2**': Just a copy of the above.
17. **'Wap1\_log\_price\_ret\_per\_spread\_sqr\_vol\_buks'**: # variance/ squared volatitliy of wap1 price returns per unit of spread # large value indicates volatilty. When the numerator is big then liquidity is less. When denominator is big then liquidity is also less. A large number indicates high volatility.
18. **'Wap1\_log\_price\_ret\_per\_liq2\_vol\_buks'**: # variance/ squared volatitliy of wap1 price returns per unit of liquidity # small value indicates less volatilty?
19. **'Log\_liquidity1\_ret\_sqr\_vol\_buks'**: # measure of variance/ squared volatility of log liquidity1 returns. Measure of variance in log liquidity in each bucket. If liquidity fluctuates then volatility is also coming.
20. **'Log\_spread\_ret\_sqr\_vol\_buks'**: # measure of variance/ squared volatility of log spread returns in each bucket.
21. 'Book\_delta\_count\_buks': # counting number of data points available in each time bucket.
22. **’'Wap1\_log\_price\_wavg'**: # time weighted average of wap1\_log\_price in each time bucket.
23. **'Wap2\_log\_price\_wavg'**: same as above for level 2
24. **'Wap\_eqi\_price0\_wavg'**: # time weighted average of wap\_eqi\_price0 equilibrium price in each time bucket
25. **'Wap\_eqi\_price1\_wavg'**: same as above for price1
26. **’Wavg\_wap1\_log\_price\_amp\_diff'**: # amplification of the difference between wap1\_log\_price\_amp\_max\_wavg and wap1\_log\_price\_amp\_min\_wavg. It is like a measure of spread between the max and min wap1 prices in a time id.
27. **'Wavg\_wap\_eqi\_price0\_amp\_diff'**: same as above for price 0
28. **'Liquidity1\_wavg'**: time weighted average of liq1
29. **'Liquidity2\_wavg'**: same as above for liq2.
30. **'Root\_liquidity2\_wavg'**: time weighted average of square root of liq2
31. **'Spread\_wavg'**: time weighted average of spread
32. **'Inv\_spread\_wavg'**: time weighted average of inverse spread
33. **'Log\_spread\_wavg'**: time weighted average of log spread
34. **'Log\_spread2\_wavg'**: same as above but using log bid price 2 and log ask price2
35. **'Book\_size1\_wavg'**: # time weighted average of book size1 in each time bucket
36. **'Book\_size\_wavg'**: # time weighted average of book size1 in each time bucket
37. **'Trade\_volume\_buks'**: bucketized trade volume, Trade volume = trade size \* trade price
38. **'Sqrt\_trade\_volume\_buks'**: same as above but bucketize the square root of trade volume. Effect of trade volume on market (future price) impact.
39. **'Cube\_root\_trade\_volume\_buks'**, same as above
40. **'trade\_volume\_p2/3\_buks'**: same as above
41. **'Quart\_root\_trade\_volume\_buks'**: same as above
42. **'Trade\_count\_buks'**: # count the number of trades in each time bucket
43. **'Trade\_volume\_per\_liquidity1\_wavg\_buks'**: trade\_volume\_buks/ liquidity1\_wavg
44. **'Trade\_volume\_per\_liquidity2\_wavg\_buks'**: trade\_volume\_buks/ liquidity2\_wavg
45. **'Ask\_liq1\_diff\_wavg'**: # time weighted average of difference betweeen ask's level 1 and level 2 liquidity. ask\_liq1\_diff - bid\_liq1\_diff
46. **'Bid\_liq1\_diff\_wavg'**: # time weighted average of difference betweeen bid's level 1 and level 2 liquidity

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/>

**CLUSTERING ANALYSIS**

Identify features that have similarity across stocks.

Sqrt\_trade\_volume\_buks

Trade\_volume\_buks

Wap1\_log\_price\_ret\_vol\_buks

wap1\_log\_price\_ret\_buks

Log\_spread\_wavg

Liquidity1\_wavg

liquidity2\_wavg

* # average out along the buckets axis
* Take Log of these features.
* log\_trade\_volume\_per\_liquidity1 = np.log(np.mean( sqrt\_trade\_volume\_buks, 2)/np.mean( liquidity1\_wavg, 2))

**Perform\_clustering( )** function

CReate correlation coefficient matrix

* # feature shape of (time\_id=3830,stock\_id= 112)
* Input is a particular feature for all stocks
* # replace nan with mean
* # feature standardization along time\_id axis
* # correlation coefficient between stocks, C.shape = 112x112
* # fisher z transformation, it normalizes the distirution of C, make it more symmetric and stabilize the variance as correlation is bounded by [-1,1]
* # After arctanh the values are unbounded
* # set diagonal self-correlation elements to zero
* # mean centering
* # mean of correlation of a stock to every other stock. e.g. stock 1 to stock 2,3,4,5,6,etc..
* # e.g. For CORR\_1,2 subtract product of mean of correlation of a stock 1 to every other stock and mean of correlation of stock 2 to every other stock
* # set diagonal self-correlation elements to mean of all correlations
* # perform pca over time\_id axis

Perform PCA over the features to cluster on the PCs

* # perform pca over time\_id axis
* # 112 components, each component has 112 loadings (represnting stock ids)
* # singular values (eigenvalues) represent the variance contribution of each component or the true rank of the matrix
* # when singular values are close to zero, the corresponding components can be ignored so we multiply the components with singular values
* # Transpose, i.e. dimension of scaled\_pcs is now stock ids x principal components.
* # each row represents a stock and its entry is a contribution from each of the principal components
* # Visualize the first 3 principal components of the scaled\_pcs to select the clustering algorithm
* # compare the visualization of first 3 principal components of the scaled\_pcs with the dim. reduced UMAP of scaled\_pcs matrix
* # 1) visualize the scaled\_pcs vector distribution in low dim. space in order to choose approapriate clustering algorithm
* # lower dim. distribution of summary stats. vector is representative of the original high dim. distribution
* ### static UMAP
* # do not normalize the PCs because each component explains different amount of variance.
* # IF we normalize and bring it to same scale, we will lose the variance information between components. # replace o\_tree clusters with agglomerative hierarchical clustering
* # threshold\_depth sets the number of clusters
* # find mean principal component for each cluster
* # check if GMM is suitable for clustering
* # perform GMM clsutering
* # select all pairwise correlations between all the stocks in a cluster
* # mean of z transformed pairwise correlation coefficients of all stocks belonging to a cluster,
* # compute clustering score for a particular feature with a fixed number of clusters. Changing cluster number / threshold depth can improve or decrease clustering score.
* # select all pairwise correlations between all the stocks in a cluster
* # mean of z transformed pairwise correlation coefficients of all stocks belonging to a cluster
* # score is clustering density, sum of mean correlation per cluster over all clusters, penalized by number of clusters. A large negative or positive value is preferred. Closer to zero is bad.
* # this can also help to find the correct number of clusters
* # group stocks by cluster from smallest to largest labels
* # display the correlation values of stocks in each cluster next to each other

Find\_high\_correlation\_stocks( ) function

* # select the percentile in order to set the threshold for correlation coefficient
* # use percentile when its difficult to set the threshold at a particular value
* Find a global measure that finds a stock or a few stocks that has MOST (not necessarily high) correlation with ALL the other stocks
* # correlation coefficient between stocks, C.shape = 112x112
* # select the median correlation coefficient of each stock
* # mask of stocks with median correlation coefficient above cutoff percentile
* # select the indexes of stocks in the feature matrix > cutoff percentile (NOT stock ids)

**FEATURE ENGINEERING**

Using the **46 liquidity** **features** above, more features are created through taking ratios of quantity between earlier and later buckets, creating volatility per unit of wap1\_logprice\_ret volatility, as ….. including clustering results.

Total 153 liquidity features stored in final\_features dictionary

All features below are averaged along the buckets axis before computing the feature.

1. **‘‘Wap1\_log\_price\_ret\_vol’:**  # average out along the buckets axis
2. ‘**log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol**’: large movements in log\_liquidity and wap\_eqi\_price1 regardless of the directions per unit of wap1\_log\_price\_ret\_vol. Taking log of this gives positive and negative numbers if quantity is >1 or < 1.
3. **‘exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol**’: measure of increasing liquidity and volatility in equilibrium price per unit of wap1\_log\_price\_ret\_vol. Taking log of this gives positive and negative numbers if quantity is >1 or < 1.
4. ‘**exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_2**’: copy of above
5. **‘Wap1\_log\_price\_ret\_per\_liq2\_vol’**: # average out along the buckets axis
6. **‘Wap1\_log\_price\_ret\_per\_spread\_sqr\_vol’**: # average out along the buckets axis
7. ‘**log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio**’: # ratio of mean of last 15 buckets to mean of first 15 buckets within a time id. ratio > 1 means increasing trend. ratio < 1 means decreasing trend.
8. **‘Wap1\_log\_price\_ret\_per\_liq2\_vol\_15\_ratio’:** similar to above.
9. **‘Wap1\_log\_price\_ret\_per\_spread\_sqr\_vol\_15\_ratio’**: # ratio of mean of last 15 buckets to mean of first 15 buckets within a time id. ratio > 1 means increasing trend. ratio < 1 means decreasing trend
10. ‘**exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio**’: similar to above
11. **‘exp\_log\_liq1\_ret\_**\***\_wap\_eqi\_price1\_ret\_vol\_2\_15\_ratio** ’: similar to above
12. **‘Wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol\_15\_ratio’:** similar to above
13. **‘Wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol\_15\_ratio\_median\_stock’**: # ratio of mean of last 15 buckets to mean of first 15 buckets within a time id. ratio > 1 means increasing trend. ratio < 1 means decreasing trend # median across all stocks (dimension 1) for that time id.
14. **‘log\_liq2\_ret\_**\***\_wap\_eqi\_price1\_ret\_vol\_15\_ratio\_median\_stock’**: similar to above.
15. **'Wap1\_log\_price\_ret\_per\_spread\_sqr\_vol\_15\_ratio\_median\_stock'’**: similar to above.
16. **‘Wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol’**: squared volatility in wap2 log price returns when wap1 log price returns are constant per unit of wap1\_log\_price\_ret\_vol. Taking log of this gives positive and negative numbers if quantity is >1 or < 1.
17. **‘Wap1\_log\_price\_ret\_neg\_log\_liq\_ret\_sqr\_vol’**: the wap1\_log\_price\_ret when log\_liquidity\_ret is negative i.e. decreasing liquidity per unit of wap1\_log\_price\_ret\_vol.
18. **‘Wap1\_log\_price\_ret\_pos\_log\_liq\_ret\_sqr\_vol’**: similar to above.
19. **‘Wap1\_log\_price\_ret\_pos**-**neg\_log\_liq\_ret\_sqr\_vol**: # difference between volatitlity in wap1\_log\_price\_ret when liquidity1 is positive and negative. i.e. increases minus decreases

get\_cohesion\_features(train\_buckets, final\_features, buk=0) function

* # uses wap1\_log\_price\_ret\_buks with buk from bucket 0 to 30 of a time id, all
* When buk is set to 10 and 20 buckets start from 10 and 20. NOT all buckets used. uses wap1\_log\_price\_ret\_buks from bucket 10 to 30 of a time id, last two thirds, uses wap1\_log\_price\_ret\_buks from bucket 20 to 30 of a time id, last one third
* This function only works with one feature **‘wap1\_log\_price\_ret\_buks’**

1. ''**wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_vol\_ast\_per\_wap1\_lprice\_ret\_vol'+ buk**': It’s made up of three parts
   1. **Stocks\_overall\_wap1\_log\_price\_ret\_vol**: # variance along time\_id axis, mean along bucket axis and then square root. # basically standard deviation of wap1\_log\_price\_ret\_buks in each stock. This is like (overall) volatility over entire time period for each stock. # shape of (1,112,1). IT is overall volatility from all time ids for each stock.
   2. **‘Wap1\_log\_price\_ret\_normalized’:**  # normalize the variance of in each time id of wap1\_log\_price\_ret\_buks by overall volatility (**Stocks\_overall\_wap1\_log\_price\_ret\_vol**) from all time ids, assume that mean of wap1\_log\_price\_ret\_buks is zero, # shape of (3830,112,30)
   3. Variance across all all stocks in each time id (3830,1,1) is broadcasted divided by wap1\_log\_price\_ret\_vol (3830,112,1) which has volatility in all time ids and all stocks. By dividing we are calculating how much of the overall volatility across all stocks is contributed by each stock.
   4. multiply by stocks\_overall\_wap1\_log\_price\_ret\_vol (shape = (1,112,1) ) to get the original variance of wap1\_log\_price\_ret\_buks to get final shape of (3830,112,1)
   5. Volatility across stocks at each time id divided by volatility at each stocks and time id (i.e. wap1\_log\_price\_ret\_vol) = factor of overall volatility across stocks contributed by each stock in each time id. This is multiplied by (stocks\_overall\_wap1\_log\_price\_ret\_vol) overall volatility across time for each stock. It is just a scalar giving importance to amount of variance across all time. # at = across time, as = across stock
2. '**wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_avg\_ast\_per\_wap1\_lprice\_ret\_vol'+ buk**’: same as above but instead of variance its average.
3. ‘**'wap1\_log\_price\_ret\_normalized\*mean\_centered\_per\_wap1\_lprice\_ret\_vol' + buk** ’ : wap1\_log\_price\_ret\_normalized is mean centred by removing the mean across stocks at each time id. It is then squared and averaged along the buckets and then square root taken. Mean across stocks represents the overall markets wap1\_log\_price\_rets. It is divided by wap1\_log\_price\_ret\_vol. THen multiplied by stocks\_overall\_wap1\_log\_price\_ret\_vol. # deviation from market is np.mean(wap1\_log\_price\_ret\_normalized, 1, keepdims=True) minus wap1\_log\_price\_ret\_normalized

get\_misc\_features(train\_buckets, final\_features) function

* Features used in this function are
  + Trade\_volume\_buks
  + Wap1\_log\_price\_ret\_vol\_buks
  + Sqrt\_trade\_volume\_buks
  + Liquidity2\_wavg
  + Log\_spread2\_wavg
  + Log\_spread\_ret\_sqr\_vol\_buks
  + Log\_liquidity1\_ret\_sqr\_vol\_buks
  + Log\_liquidity2\_ret\_sqr\_vol\_buks
  + trade\_count\_buks
  + Book\_delta\_count\_buks
  + trade\_count\_buks

1. ‘Soft\_stock\_mean\_tvpl2’: soft\_stock\_mean\_tvpl2\_:0
2. Soft\_stock\_mean\_tvpl2\_:10 : soft\_stock\_mean\_tvpl2\_f10
3. Soft\_stock\_mean\_tvpl2\_:20 : soft\_stock\_mean\_tvpl2\_f20

…

…

…

**All Features:**

Total liquidity features shape (428932, 152)

**Features same with Jager’s code:**

1. stock\_id = stock\_id
2. time\_id = time\_id
3. log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol = cvol1
4. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol = evol1
5. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol = e2vol1
6. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_2 = evol1
7. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_2 = e2vol1
8. wap1\_log\_price\_ret\_per\_liq2\_vol = lvol1
9. wap1\_log\_price\_ret\_per\_spread\_sqr\_vol = svol1
10. log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio = cvol1\_15\_15
11. wap1\_log\_price\_ret\_per\_liq2\_vol\_15\_ratio = lvol1\_15\_15
12. wap1\_log\_price\_ret\_per\_spread\_sqr\_vol\_15\_ratio = svol1\_15\_15
13. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio = evol1\_15\_15
14. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio = e2vol1\_15\_15
15. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_2\_15\_ratio = evol1\_15\_15
16. exp\_log\_liq1\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_2\_15\_ratio = e2vol1\_15\_15
17. log\_liq2\_ret\_\*\_wap\_eqi\_price1\_ret\_vol\_15\_ratio\_median\_stock = cvol1\_15\_15s
18. wap1\_log\_price\_ret\_per\_spread\_sqr\_vol\_15\_ratio\_median\_stock = svol1\_15\_15s
19. wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol = vol2e
20. wap1\_log\_price\_ret\_neg\_log\_liq\_ret\_sqr\_vol = volq0\_lm
21. wap1\_log\_price\_ret\_pos\_log\_liq\_ret\_sqr\_vol = volq0\_lp
22. wap1\_log\_price\_ret\_pos-neg\_log\_liq\_ret\_sqr\_vol = volq0\_dt
23. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_vol\_ast\_per\_wap1\_lprice\_ret\_vol\_:0 = tbin\_var
24. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_avg\_ast\_per\_wap1\_lprice\_ret\_vol\_:0 = market\_var
25. wap1\_log\_price\_ret\_normalized\*mean\_centered\_per\_wap1\_lprice\_ret\_vol\_:0 = deviations\_from\_market
26. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_vol\_ast\_per\_wap1\_lprice\_ret\_vol\_:10 = tbin\_var\_from\_10
27. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_avg\_ast\_per\_wap1\_lprice\_ret\_vol\_:10 = market\_var\_from\_10
28. wap1\_log\_price\_ret\_normalized\*mean\_centered\_per\_wap1\_lprice\_ret\_vol\_:10 = deviations\_from\_market\_from\_10
29. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_vol\_ast\_per\_wap1\_lprice\_ret\_vol\_:20 = tbin\_var\_from\_20
30. wap1\_lprice\_ret\_vol\_ati\_\*\_wap1\_lprice\_ret\_avg\_ast\_per\_wap1\_lprice\_ret\_vol\_:20 = market\_var\_from\_20
31. wap1\_log\_price\_ret\_normalized\*mean\_centered\_per\_wap1\_lprice\_ret\_vol\_:20 = deviations\_from\_market\_from\_20
32. soft\_stock\_mean\_tvpl2\_:0 = soft\_stock\_mean\_tvpl2
33. soft\_stock\_mean\_tvpl2\_:10 = soft\_stock\_mean\_tvpl2\_f10
34. soft\_stock\_mean\_tvpl2\_:20 = soft\_stock\_mean\_tvpl2\_f20
35. soft\_stock\_mean\_tvpl2\_liqf = soft\_stock\_mean\_tvpl2\_liqf
36. soft\_stock\_mean\_tvpl2\_liqf\_volf10 = soft\_stock\_mean\_tvpl2\_liqf\_volf10
37. soft\_stock\_mean\_tvpl2\_liqf\_volf20 = soft\_stock\_mean\_tvpl2\_liqf\_volf20
38. v1proj\_25\_15 = v1proj\_25\_15
39. v1proj\_25\_15\_lr1\_high\_corr\_stocks = v1proj\_25\_15\_lr1\_hc
40. v1proj\_25\_15\_vol1\_high\_corr\_stocks = v1proj\_25\_15\_vol1\_hc
41. v1proj\_25\_lr1\_high\_corr\_stocks = v1proj\_25\_lr1\_hc
42. v1proj\_25\_vol1\_high\_corr\_stocks = v1proj\_25\_vol1\_hc
43. lsvol = lsvol
44. liqvol1 = liqvol1
45. liqvol1\_smean = liqvol1\_smean
46. liqvol2 = liqvol2
47. liqvol1\_15\_15 = liqvol1\_15\_15
48. trade\_count = trade\_count
49. root\_trade\_count = root\_trade\_count
50. root\_trade\_count\_smean = root\_trade\_count\_smean
51. root\_book\_delta\_count = root\_book\_delta\_count
52. root\_trade\_count\_var = root\_trade\_count\_var
53. v1proj\_29\_15 = v1proj\_29\_15
54. v1proj\_20 = v1proj\_20
55. v1proj\_25 = v1proj\_25
56. v1proj\_29 = v1proj\_29
57. v1proj\_29\_q1 = v1proj\_29\_q1
58. v1proj\_29\_q3 = v1proj\_29\_q3
59. v1proj\_25\_q1 = v1proj\_25\_q1
60. v1proj\_25\_q3 = v1proj\_25\_q3
61. v1proj\_29\_15\_q1 = v1proj\_29\_15\_q1
62. v1proj\_29\_15\_q3 = v1proj\_29\_15\_q3
63. v1proj\_25\_15\_q1 = v1proj\_25\_15\_q1
64. v1proj\_25\_15\_q3 = v1proj\_25\_15\_q3
65. v1proj\_20\_std = v1proj\_20\_std
66. v1proj\_29\_q3q1 = v1proj\_29\_q3q1
67. tvpl2\_rmed2v1 = tvpl2\_rmed2v1
68. tvpl2\_rmed2v1lf25 = tvpl2\_rmed2v1lf25
69. tvpl2\_rmed2v1lf29 = tvpl2\_rmed2v1lf29
70. tvpl2 = tvpl2
71. tvpl2\_liqf10 = tvpl2\_liqf10
72. tvpl2\_liqf20 = tvpl2\_liqf20
73. tvpl2\_liqf29 = tvpl2\_liqf29
74. tvpl2\_smean\_vol = tvpl2\_smean\_vol
75. tvpl2\_smean\_vol\_liqf10 = tvpl2\_smean\_vol\_liqf10
76. tvpl2\_smean\_vol\_liqf20 = tvpl2\_smean\_vol\_liqf20
77. tvpl2\_smean\_vol\_liqf29 = tvpl2\_smean\_vol\_liqf29
78. v1liq2projt5 = v1liq2projt5
79. v1liq2projt10 = v1liq2projt10
80. v1liq2projt20 = v1liq2projt20
81. liqt10rf29 = liqt10rf29
82. liqt20rf29 = liqt20rf29
83. v1liq2sprojt10f25 = v1liq2sprojt10f25
84. v1liq2sprojt5f25 = v1liq2sprojt5f25
85. v1spprojt10f29 = v1spprojt10f29
86. v1spprojt15f25 = v1spprojt15f25
87. v1spprojt15f29 = v1spprojt15f29
88. v1spprojt15f29\_q1 = v1spprojt15f29\_q1
89. v1spprojt15f29\_q3 = v1spprojt15f29\_q3
90. v1spprojt15f25\_q1 = v1spprojt15f25\_q1
91. v1spprojt15f25\_q3 = v1spprojt15f25\_q3
92. v1spprojtf29\_q1 = v1spprojtf29\_q1
93. v1spprojtf29\_q3 = v1spprojtf29\_q3
94. v1spprojtf25\_q1 = v1spprojtf25\_q1
95. v1spprojtf25\_q3 = v1spprojtf25\_q3
96. wap1\_log\_price\_ret\_vol\_from\_0 = vol1\_from\_0
97. wap1\_log\_price\_ret\_volstock\_mean\_from\_0 = vol1stock\_mean\_from\_0
98. wap1\_log\_price\_ret\_vol\_from\_10 = vol1\_from\_10
99. wap1\_log\_price\_ret\_volstock\_mean\_from\_10 = vol1stock\_mean\_from\_10
100. wap1\_log\_price\_ret\_vol\_from\_20 = vol1\_from\_20
101. wap1\_log\_price\_ret\_volstock\_mean\_from\_20 = vol1stock\_mean\_from\_20
102. wap1\_log\_price\_ret\_vol\_from\_25 = vol1\_from\_25
103. wap1\_log\_price\_ret\_volstock\_mean\_from\_25 = vol1stock\_mean\_from\_25
104. vol1\_mean = vol1\_mean
105. mean\_half\_delta = mean\_half\_delta
106. mean\_half\_delta\_lsprd = mean\_half\_delta\_lsprd
107. log\_wap1\_log\_price\_ret\_vol = vol1
108. target = target

**Feature differences with Jager’s code (due to different clustering method):**

1. 'wap1\_log\_price\_ret\_vol',
2. 'wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol\_15\_ratio',
3. 'wap2\_logprice\_ret\_changes\_n\_wap1\_logprice\_ret\_constant\_sqr\_vol\_15\_ratio\_median\_stock',
4. 'liqvol1\_smean\_c3',
5. 'root\_trade\_count\_smean\_c1',
6. 'root\_trade\_count\_smean\_c2',
7. 'root\_trade\_count\_smean\_c3',
8. 'Trade\_count\_15\_15',
9. 'Root\_trade\_count\_15\_15',
10. 'v1proj\_25\_15\_std',
11. 'V1proj\_29\_15\_std',
12. 'V1proj\_25\_std',
13. 'V1proj\_29\_std',
14. 'v1proj\_25\_c1',
15. 'V1proj\_25\_c2',
16. 'V1proj\_25\_c3',
17. 'V1proj\_25\_c4',
18. 'v1proj\_25\_c5',
19. 'Soft\_stock\_mean\_tvpl2\_c1',
20. 'soft\_stock\_mean\_tvpl2\_c2',
21. 'Soft\_stock\_mean\_tvpl2\_c3',
22. 'soft\_stock\_mean\_tvpl2\_10\_c1',
23. 'Soft\_stock\_mean\_tvpl2\_10\_c2',
24. 'soft\_stock\_mean\_tvpl2\_10\_c3',
25. 'Soft\_stock\_mean\_tvpl2\_20\_c1',
26. 'soft\_stock\_mean\_tvpl2\_20\_c2',
27. 'Soft\_stock\_mean\_tvpl2\_20\_c3',
28. 'v1proj\_25\_c1\_std', 'v1proj\_25\_c2\_std',
29. 'V1proj\_25\_c3\_std',
30. 'V1proj\_25\_c4\_std',
31. 'v1proj\_25\_c5\_std',
32. 'v1proj\_25\_vc1',
33. 'V1proj\_25\_vc2',
34. 'V1proj\_25\_vc3',
35. 'v1proj\_25\_vc4',
36. 'v1proj\_25\_vvc1',
37. 'v1proj\_25\_vvc2',
38. 'v1proj\_25\_vvc3',
39. 'v1spprojt15f25\_c1',
40. 'v1spprojt15f25\_c2',
41. 'v1spprojt15f25\_c3',
42. 'v1spprojt15f25\_c4',
43. 'v1spprojt15f25\_vc1',
44. 'v1spprojt15f25\_vc2',
45. 'v1spprojt15f25\_vc3',
46. 'wap1\_log\_price\_ret\_cluster2\_tr',
47. 'wap1\_log\_price\_ret\_cluster3\_tr'

**Our own features:**

‘First\_10\_min\_vol\_df’:

‘Trade\_price\_std’:

‘Trade\_price\_real\_vol’:

‘Trade\_size\_std’:

‘Trade\_size\_mean’:

‘Trade\_order\_count\_std’:

‘Trade\_order\_count\_mean’: