# **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".

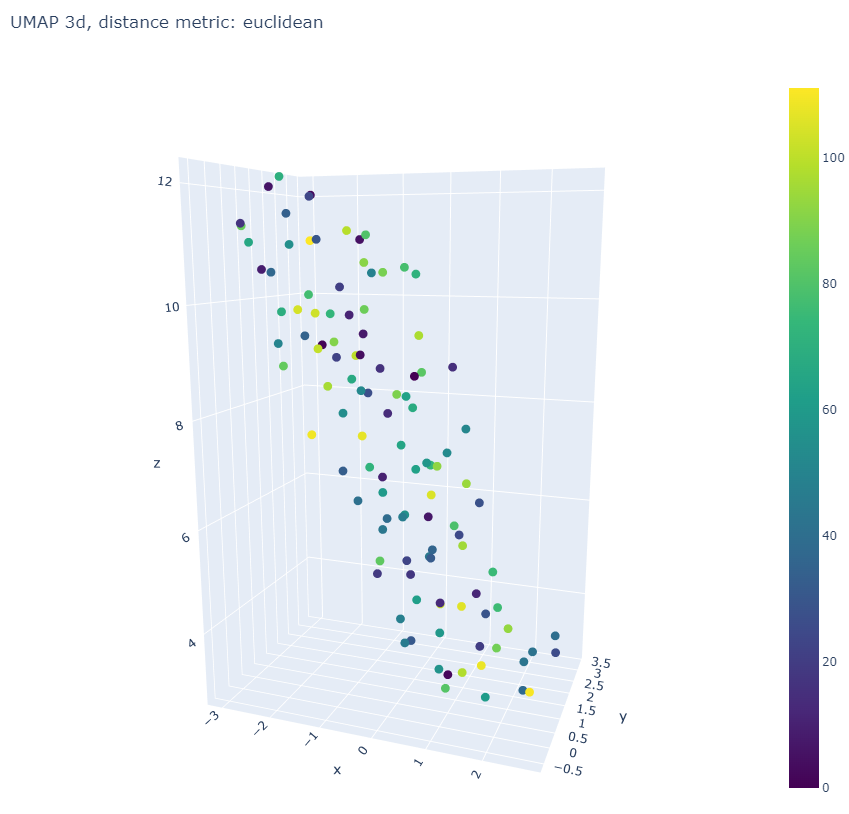
How about the standard deviation? Multiplication/Division by a constant changes it accordingly but we actually want the volatility to be consistent over time slots and stocks, do we?”

-- “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.

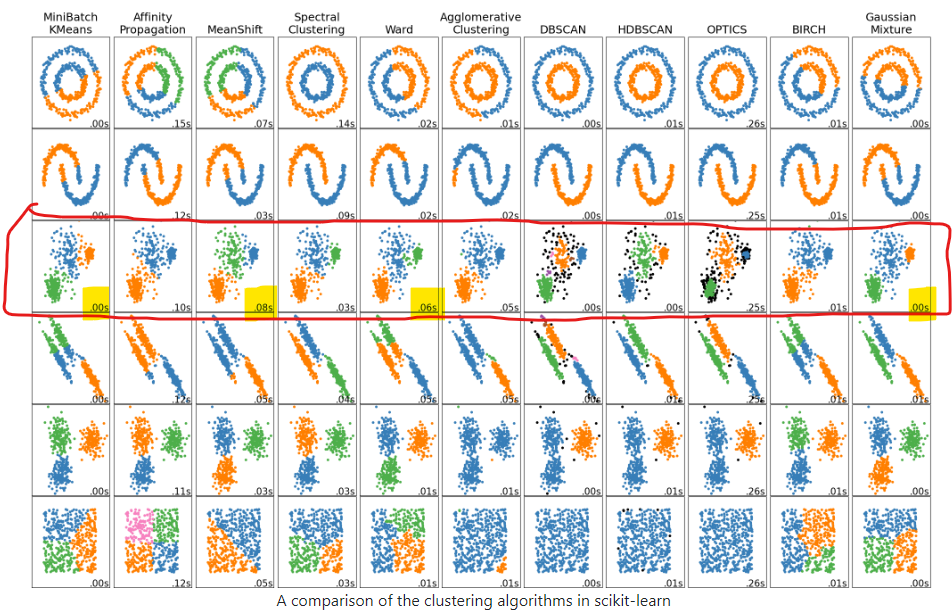
If you look at the logs: dividing the prices by a constant is equivalent to subtracting a constant from all logs of prices. The log return (as difference of two subsequent logs of prices) doesn't change under this transformation.”

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.

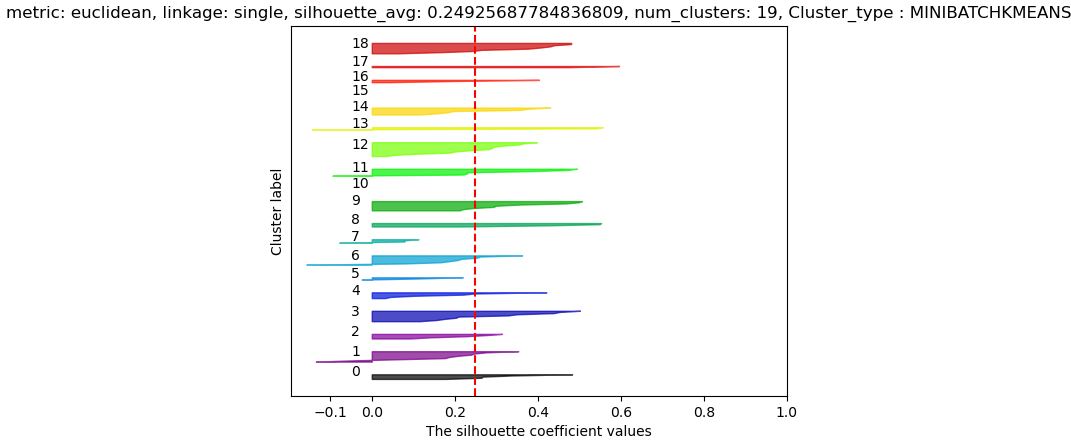
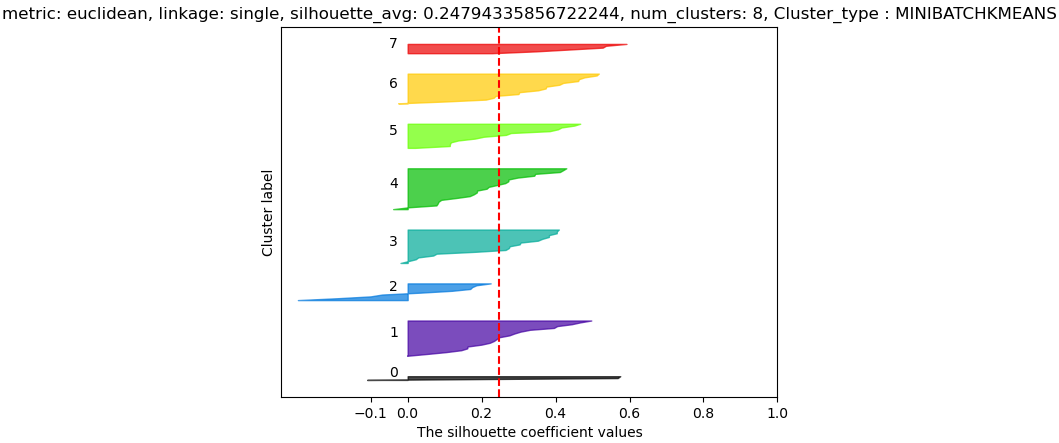
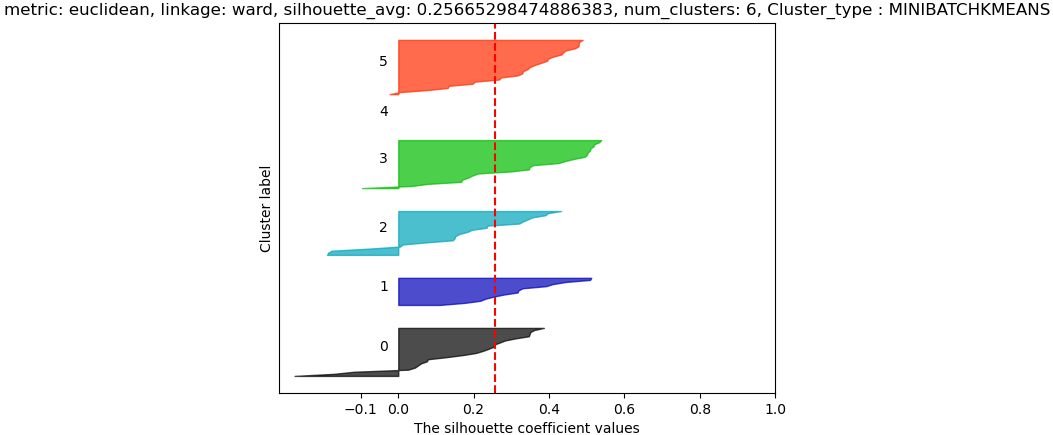
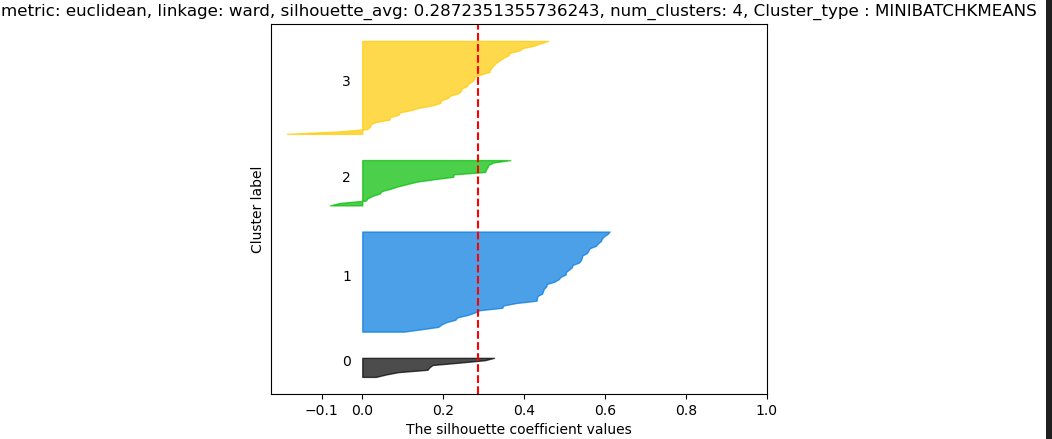
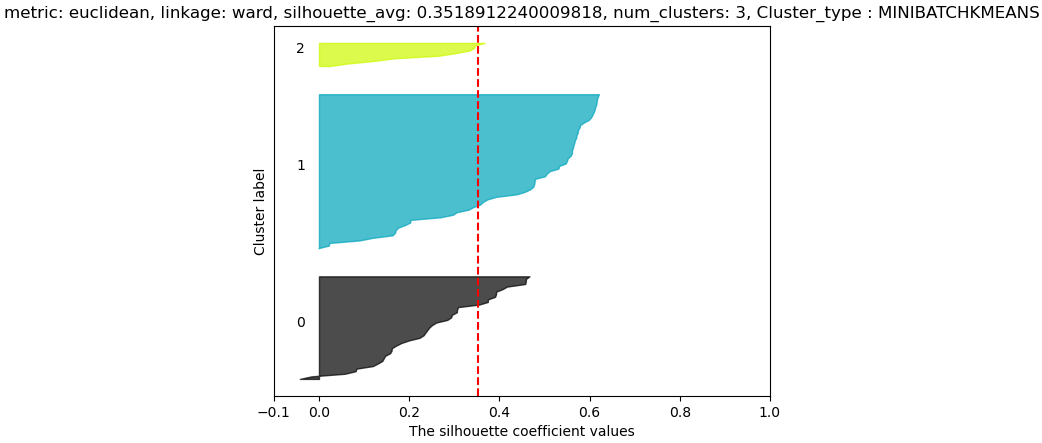
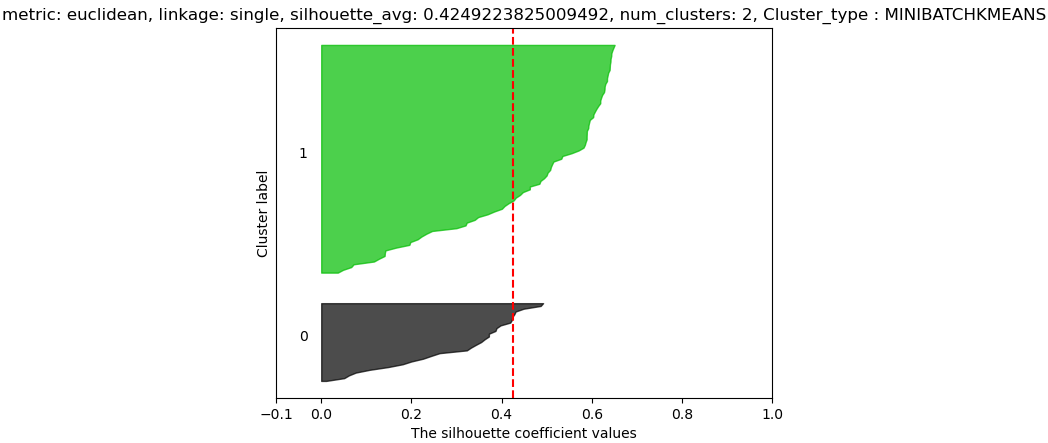
Possible answer: <https://scikit-learn.org/stable/modules/clustering.html>



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



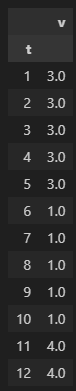
df = pd.DataFrame({'t':[2,6,11], 'v':[3,1,4]}).set\_index('t')

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

SIngular values/eigen values close to zero represent the directions with very low variance. Here we don’t try to convert the singular values to eigen values. We simply make the singular value close zero even smaller (contract) by taking a square root of it.

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.

26) optuna training stops after 1 fold of timeseries split training using LGBM. Linux freezes and crashes.

* When folds were increased to 30, it can run up to 11 folds, it freezes at 12th fold with starting 1046 training indices and 1046 - 1132 val indices. After this it get stuck/freezes.
* PRobably data training on very large data is taking time
* data size can fit in memory because htop shows < 32g
* When the **dataset is huge** and we only have **limited compute capability** we can use **bagging and pasting techniques of ensemble learning** to train different mutually exclusive subsets of the training data using different models of the same algorithm e,g, LGBM). E.g. each lgbm has different hyperparameters. Downside is this method does not allow to capture ALL the interdependencies (bagging can alleviate this a little due to replacement but take note of OOB). These interdependencies can be encoded in the features in possible.

27) bin size 272 cannot be used with GPU error.

* THis bin size refers to the number of indices in the training set during timeseriesplit.

Fold: 3

train\_index [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179

180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197

198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215

216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233

234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251

252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269

270 271 272]

valid\_index [273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290

291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308

309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326

327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344

345 346 347 348 349 350 351 352 353 354 355 356 357 358]

"/home/optimusprime/miniconda3/envs/optiver\_linux/lib/python3.10/site-packages/lightgbm/basic.py", line 3437, in \_\_init\_\_ \_safe\_call(\_LIB.LGBM\_BoosterCreate(File "/home/optimusprime/miniconda3/envs/optiver\_linux/lib/python3.10/site-packages/lightgbm/basic.py", line 263, in \_safe\_call raise LightGBMError(\_LIB.LGBM\_GetLastError().decode('utf-8'))

lightgbm.basic.LightGBMError: bin size 272 cannot run on GPU

28) how to encode time order information (ordinal categorical feature) in train-test tabular dataset for lgbm? How does model know that t=0 sample occurred before t=1?

* Include sequence\_id to represent number from 1 - 3830
* Make this categorical variable explicit

29) we want to include periodicity information about the time series?

* 9 time\_ids represent 1 day.
* 5 trading days in a week, 5\*9 = 45 time ids
* 21 trading days in a month, 21\*9 = 189 time\_ids
* Make this categorical variable explicit

30) What is the reason for dividing the target by first 10 mins realized volatility?

The reason for dividing the 2nd 10 min rvol by 1st 10 min rvol is because there are 4 cases/scenarios possible.

The first two cases are least likely because of volatility clustering. The last two cases are more likely and these ratios tend to lead to the similar number. For example 1/1 = 1, 5/5 = 1. This helps to reduce the amount of variance in target (i.e. 1 to 5) that has to be learned by the model. In both examples the model has to learn the number 1. The actual target can be extracted by simple transformation. I.e. 1\*1=5 for first example and 1\*5 = 5 for 2nd example.

Source: <https://www.kaggle.com/competitions/optiver-realized-volatility-prediction/discussion/276137>

* Target volatilities span a couple orders of magnitude which might make learning more difficult, volatility ratios are more compact.
* The target ratio is much more correlated between stock ids than the raw target. This is likely important when combining stock ids. Again this makes it easier to predict similar stocks. Don't have to predict two different things but just 1 thing which is similar to both.

**Similarly, how can we use the fact of leverage effect, i.e. volatilities are smaller when wap1 is in uptrend and volatilities are higher when wap1 is in downtrend as feature engineering so that the model only has to learn lesser information?**

31) why is correlation between prediction ratio and target ratio much smaller than prediction and target like below?

corr(p/v1v, y\_val/v1v) 0.37931470492426755

log(corr( )) 0.4821924459770426

corr(p, y\_val) 0.9004878600907132

log(corr( )) 0.9287013679686396

It is possibly due to very low variance in prediction ratio and target ratio signals compared to prediction and target signals so calculated correlation is bigger.

32) why is all\_stock\_train\_pred\_df having NAN values even after assigning non null values???

ANS:

In the following code, .values was missing so there was some index mismatch.

“The reason of this problem occurring is the index mismatch. When you assign from an array using .values, the indexing of the dataframe on the left is used, because of avoiding (index mismatch) conflict.”

def compute\_all\_stock\_train\_pred\_df(self, unique\_stock\_ids, train\_pred):

unique\_train\_time\_ids = self.time\_id\_order[:self.train\_time\_id\_ind]

all\_stock\_train\_pred\_df = pd.DataFrame(index=unique\_train\_time\_ids)

for s in unique\_stock\_ids:

st\_index = self.train\_stock\_id == s

t\_index = self.train\_time\_id[st\_index]

all\_stock\_train\_pred\_df.loc[t\_index, s] = train\_pred[st\_index].values

#all\_stock\_train\_pred\_df = all\_stock\_train\_pred\_df.ffill().bfill()

return all\_stock\_train\_pred\_df

33) How to incorporate stock specific feature in the dataset?

E.g. stock id 31 has sig. Autoco. Lag at 9 and stock id 61 has at lag 35 how to include this in the dataset to tell the model.

Loss of Information: This can potentially lead to an incomplete understanding of the dataset and might affect the performance of your model.

Biased Model: the model might learn to give undue importance to that stock. This could introduce bias in the model's predictions.

Inconsistent dataset causes issues during model training or evaluation.

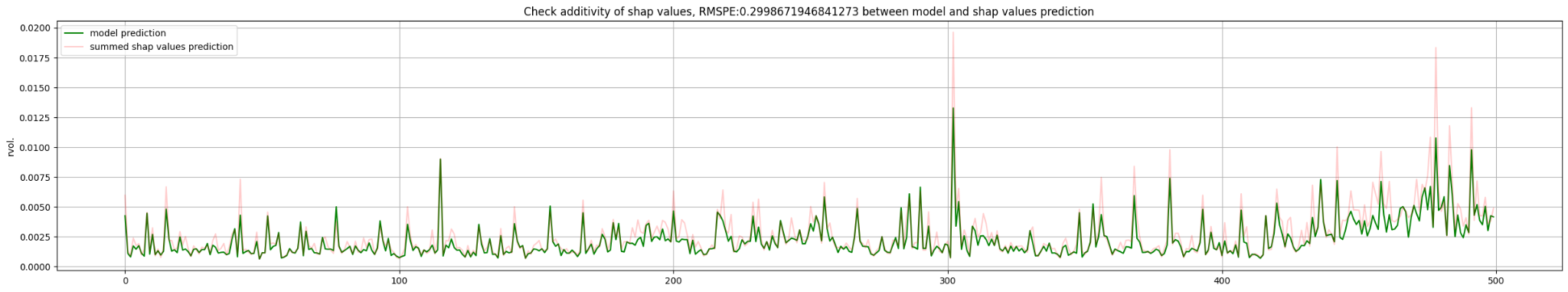
scenarios where having all null values is acceptable or unavoidable. For instance:

Limited Availability: If data for certain stocks is genuinely limited or unavailable for certain features, then having null values might be unavoidable.

Feature Importance: If you determine that certain features are not relevant or important for certain stocks based on domain knowledge or feature importance analysis, then having null values for those stocks might be acceptable.

34) Why is shap value outputs in a different scale compared to model prediction outputs?

Ans: forgot to multiply the **v1tr** weighting to (shap\_base\_value + shap\_values.sum() )

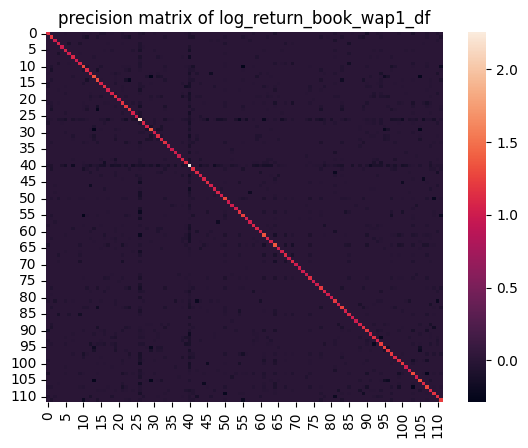


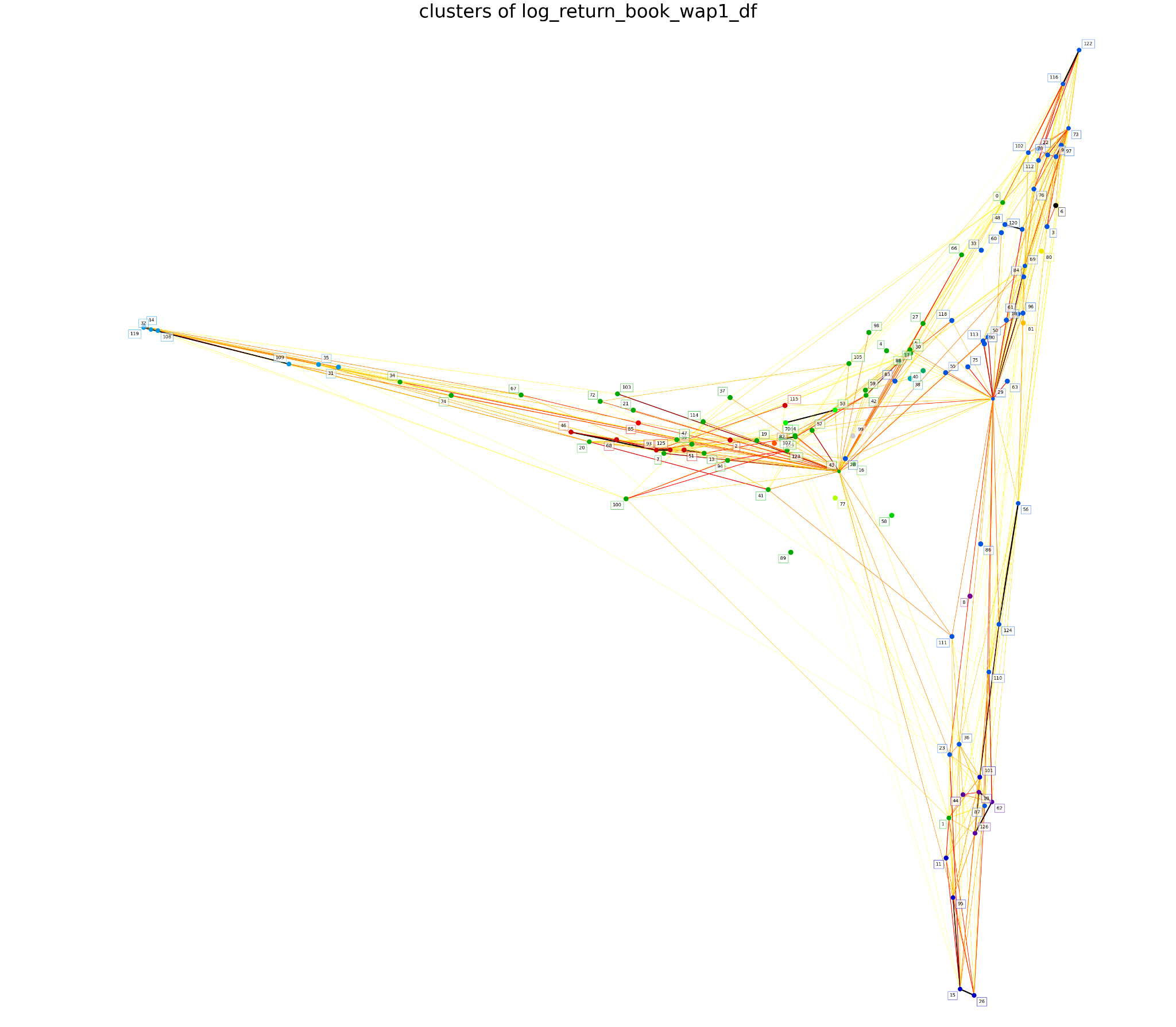
35) Question: MVARCH model of 112 x 112 assets is taking too long to run.

Ans: Need to do returns-based clustering and to find correlation btw. assets using Graphical Lasso, MSD, and Affinity clustering.

Perform MVARCH on these correlated assets to reduce dimension from 112x112 to a few groups of lesser dim.

<https://scikit-learn.org/stable/auto_examples/applications/plot_stock_market.html>





Cluster 1: [6], Number of stocks: 1

Cluster 2: [8], Number of stocks: 1

Cluster 3: [10, 44, 62, 126], Number of stocks: 4

Cluster 4: [11, 15, 26, 95, 101], Number of stocks: 5

Cluster 5: [3, 9, 18, 22, 23, 28, 29, 33, 36, 48, 50, 55, 56, 60, 61, 63, 69, 73, 75, 76, 78, 83, 84, 86, 87, 90, 96, 97, 102, 110, 111, 112, 113, 116, 118, 120, 122, 124], Number of stocks: 38

Cluster 6: [14, 31, 32, 35, 108, 109, 119], Number of stocks: 7

Cluster 7: [38], Number of stocks: 1

Cluster 8: [40], Number of stocks: 1

Cluster 9: [0, 1, 4, 5, 7, 13, 16, 17, 19, 20, 21, 27, 30, 34, 37, 39, 41, 42, 43, 47, 52, 59, 66, 67, 70, 72, 74, 88, 89, 94, 98, 100, 103, 104, 105, 107, 114, 123], Number of stocks: 38

Cluster 10: [58], Number of stocks: 1

Cluster 11: [53, 64], Number of stocks: 2

Cluster 12: [77], Number of stocks: 1

Cluster 13: [80], Number of stocks: 1

Cluster 14: [81], Number of stocks: 1

Cluster 15: [82], Number of stocks: 1

Cluster 16: [85], Number of stocks: 1

Cluster 17: [2, 46, 51, 68, 93, 115, 125], Number of stocks: 7

Cluster 18: [99], Number of stocks: 1

The stocks within a cluster are conditionally correlated given all the other stocks.

Using Mvarch on a cluster will find individual contributions of each stock to a particular stock.

Even Mvarch of 38 stocks is taking too long. Perform Graphical Lasso again on larger clusters to split them into smaller clusters!!

So we need to run Graphical lasso on the 38 stocks again to split it up into further sub-clusters. Or we could just equally split (without graphical lasso) the larger cluster into say 10 subclusters each and run Mvarch on this.

36) How to ensure different conditional correlations (different stocks in each clusters) during different regimes?

Ans: Split 3830 into quarters. 63 days \* 9 time ids per day = 567 time ids for each regime/quarter.

To save time just split into two equal halves.

37) What is the realised quarticity?

Realized Quarticity (RQ) is a measure of the kurtosis or fourth moment of the distribution of intraday returns of a financial asset. It extends the concept of realized volatility, which captures the variance or second moment of returns, to include higher-order moments.

Here's how Realized Quarticity is typically calculated:

Compute the Intraday Returns: Calculate the returns of the financial asset over intraday intervals. These returns are typically computed as the difference between consecutive prices, expressed as a percentage or logarithmic return.

Calculate the Quarticity Estimator: The quarticity estimator is computed as the sum of the fourth powers of intraday returns, **normalized by the number of observations.**

Scale Factor: A scaling factor may be applied to adjust for the length of the intraday intervals and to ensure that RQ is comparable across different time periods or assets.

Realized Quarticity measures the **degree of peakedness** or fatness of the **return distribution**, **capturing** the **presence** of **extreme events** or **outliers**. A high RQ value indicates a **distribution with heavy tails** and a **higher likelihood o**f ex**treme price movements,** while a low RQ value suggests fewer extreme events.

Uses of Realized Quarticity:

Risk Management: RQ provides insights into the risk profile of financial assets by capturing the distributional properties of intraday returns beyond just volatility. It helps risk managers assess the **likelihood of extreme events** and tailor risk management strategies accordingly.

Volatility Forecasting: Realized Quarticity can be used in conjunction with other volatility measures, such as realized variance, to **improve the accuracy of volatility forecasts**. By capturing **higher-order moment**s of the **return distribution**, **RQ provides additional information** about the **shape** and **characteristics** of **volatility dynamics**.

Market Microstructure Analysis: RQ offers valuable insights into the intraday dynamics of financial markets, including the presence of fat tails and the **clustering of extreme events.** This information can be useful for understanding market microstructure phenomena and designing trading strategies that take advantage of market inefficiencies.

38) what is bipower variation and what is its use?

Bipower Variation (BPV) is a measure of volatility or variation in financial asset prices. It is commonly used in high-frequency financial data analysis, particularly in the study of intraday price movements. BPV was introduced as an alternative to traditional measures of volatility, such as squared returns or realized variance, **especially in situations where the data may exhibit jumps or other irregularities.**

Here's how BPV is calculated:

**Compute the Increment**: First, calculate the increments of the price series. In the context of intraday data, this typically involves taking the differences between consecutive price observations.

**Square the Increments**: Square each increment to ensure positivity and to amplify the contribution of larger price changes.

**Multiply Adjacent Increments:** Multiply each increment by the adjacent increment (the one following it in the time series) to amplify the contribution of larger price changes.

**Summation:** Sum up all the products obtained in the previous step.

**Normalization:** Multiply the summation by a normalization factor. The normalization factor is often derived from theoretical considerations and is used to scale the BPV value appropriately. (np.sqrt(2 / np.pi)) \*\* (-2)

The resulting BPV value provides an indication of the variation or volatility in the price series over the specified time period.

Uses of Bipower Variation:

**Robust Volatility Estimation:** BPV is particularly useful in situations where traditional volatility measures may be **biased or unreliable**, such as when dealing with **high-frequency data** that **contains jumps** or other **irregularities**.

**Risk Management:** BPV can be used in risk management models to estimate and monitor the volatility of financial assets, helping traders and investors make more informed decisions.

**Market Microstructure Analysis:** BPV provides insights into the intraday dynamics of financial markets, **including the intensity of trading activity** and the impact of market microstructure on price movements.

39) Even after splitting the cluster into smaller subclusters MVARCH is taking too long to complete one step prediction for all 112 stocks. How to speed this up?

Ans: We could perform simulation over the next few periods after fitting instead of fitting 189 points every time to predict 1 point. In this case 189 points will be fitted to predict say 5 points in the future through simulation.

40) try setting the the device setting in model\_Factory() to “cuda” from “CPU”

Ans: after some code changes with errors like

*TypeError: can't convert cuda:0 device type tensor to numpy. Use Tensor.cpu() to copy the tensor to host memory first.  
  
RuntimeError: Expected all tensors to be on the same device, but found at least two devices, cuda:0 and cpu!*

**It is still slow**  CPU is faster than GPU!!

Solution is the reduce computation complexity from “triangular ” to “diagonal” to “scalar”

40) what is model.distribution.std() ? why does it multiply to uv\_scale\_predicted?

Ans: It could be the standard deviation of samples sampled from the t distribution. This std. Is multiplied to the scale which is learnt from volatility evolution.

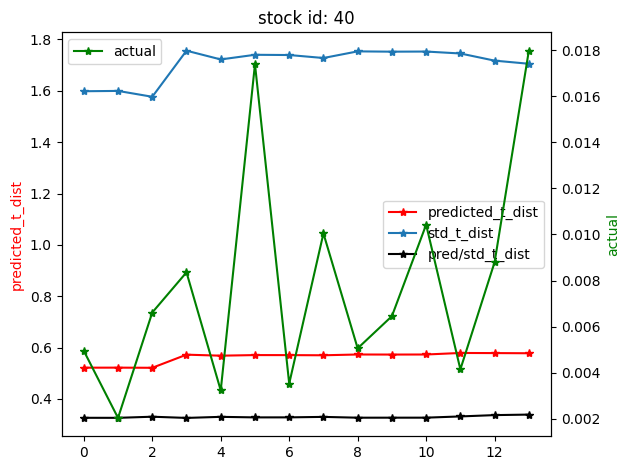
41) Why is predicted realised volatility above the actual realised volatility (over prediction) for all stocks?

The actual and predicted volatilities are at a different scale!

Is it because we assumed zero mean? This could be the problem because mean has to be subtracted before variance can be modelled. Without subtracting the mean return the returns will be squared and this will overestimate the variance. So we can model the mean returns using ARMA or set it to constant.

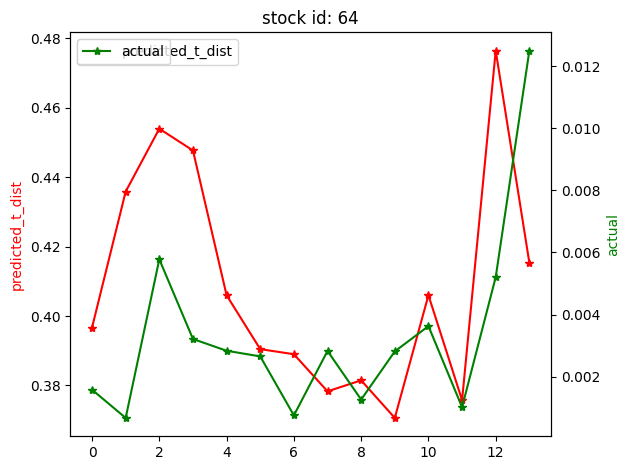
Setting to constant does NOT help reduce the scale of predictions significantly enough. Next strategy is to just feed prediction or the prediction/model.distribution.std() straight to xgboost as a feature. We can decide whether to use prediction or the prediction/model.distribution.std() by checking the spearman’s rank correlation or dynamic time warping with actual volatility signal.

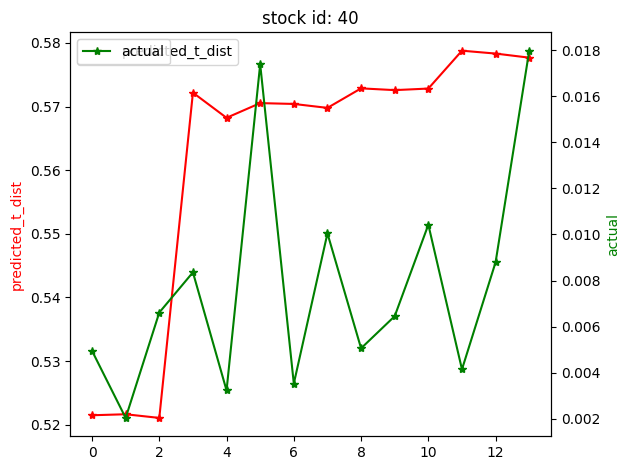
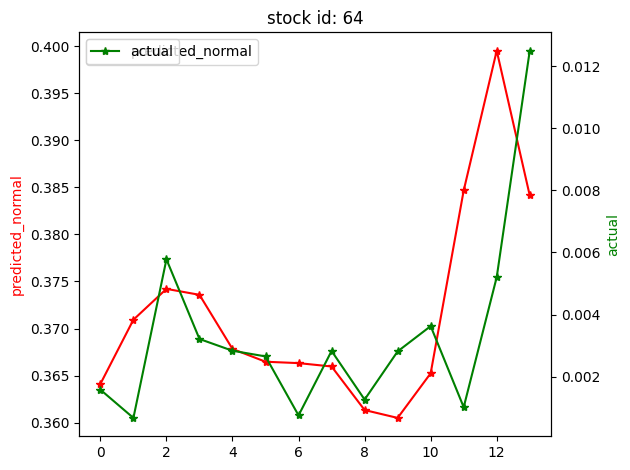
~~Another reason for higher predicted volatilities is we model the cumulative returns of past 189 time periods ( in the training data) This will have very high variation (larger overall standard deviation (label: std\_t\_dist) in the student-t returns distribution) The scale factor (label: pred/std\_t\_dist) may not be able to scale it down enough as the overall standard deviation in the student-t returns distribution may be very big. It is big and this is observed in the image below.~~

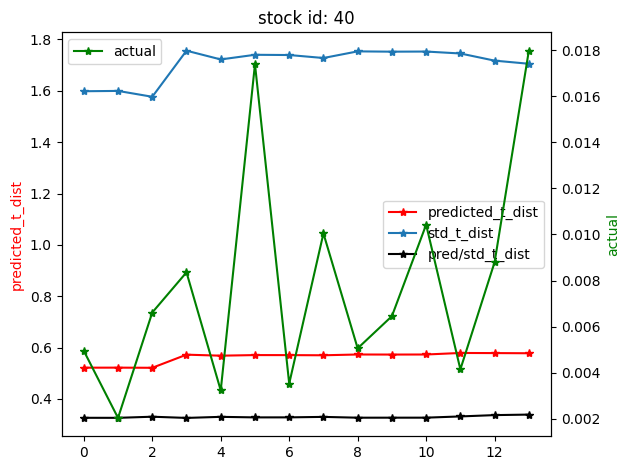


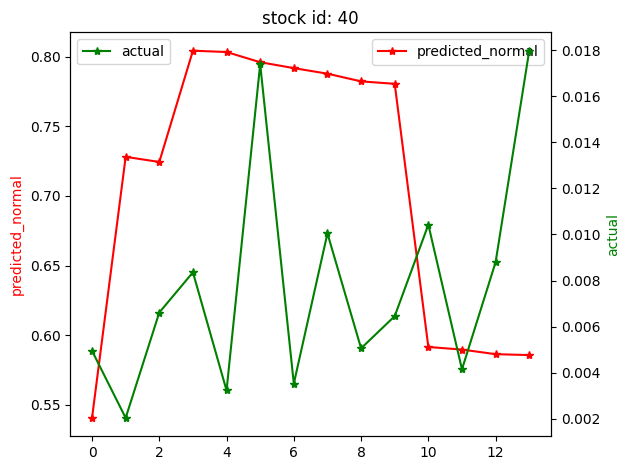
Std\_t\_dist changes with time as training data is rolling forward as well. ~~One remedy could be to use a smaller training set (i.e. closer to prediction) so that this will lower overall variation in the distribution (only includes returns variations close to the prediction step) and the scale factor might be able to bring down the prediction to the scale (level) of the actual volatilities.~~ **This does NOT help. Actually, the predictions increased even more.**

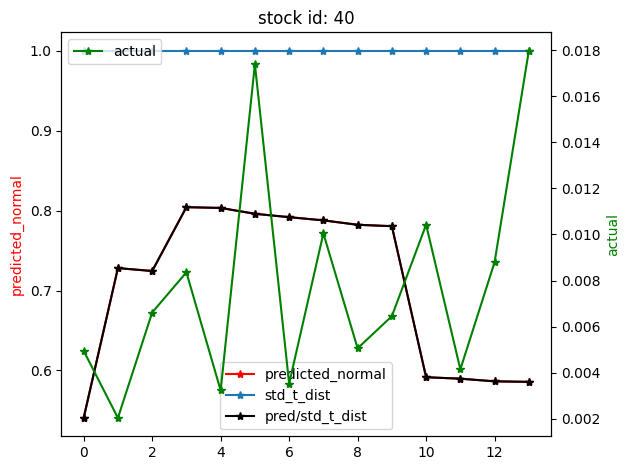
Decreasing the training set to 9\*5 = 45 time ids actually causes some volatilities predictions to be infinite!! As the training set in increased the volatility seems to decrease.











The actual and predictions scale is definitely different!!

42) Which distribution (normal or student-t) to use?

Ans: Some stocks perform well with normal distribution others with student t distribution. We could use ~~jarque bera test~~  skewness and kurtosis to check for normality of the returns distribution to choose whether to use student t or normal distribution. (how about multivariate case with multiple stocks.?? )

OR

We need to check overall performance of each distri. On all stocks and see which distribution performs better on all stocks.

43) Why do I get the error

---------------------------------------------------------------------------

ValueError Traceback (most recent call last)

<ipython-input-21-ec567626bc3c> in <cell line: 222>()

223 fig,ax = plt.subplots(2,1,figsize=(30,6))

224 data = book\_wa1\_HAR\_vol\_estimates\_full\_df[book\_wa1\_HAR\_vol\_estimates\_full\_df['st\_id']==s].iloc[start\_time\_id\_index:end\_time\_id\_index]

--> 225 HAR(data, extra\_plots=True)

226

227

5 frames

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in \_check\_n\_features(self, X, reset)

387

388 if n\_features != self.n\_features\_in\_:

--> 389 raise ValueError(

390 f"X has {n\_features} features, but {self.\_\_class\_\_.\_\_name\_\_} "

391 f"is expecting {self.n\_features\_in\_} features as input."

ValueError: X has 4 features, but LinearRegression is expecting 5 features as input.

Ans: This is because in the code below, XA is fixed for all models, the chatgpt code generation is incomplete. XA and y should change with model.

# this step is essentially performed at the last out of sample prediction

# to measure model goodness of fit. It is esentially same as the least squares done abvoe.

# Regression at t=1 for Standard Errors before performing any out-of-sample forecasts

if t == out\_sample - 1:

modelA = LinearRegression().fit(XA, y)

model = LinearRegression().fit(X, y)

modelQ = LinearRegression().fit(XQ, y)

modelF = LinearRegression().fit(XF, y)

modelC = LinearRegression().fit(XC, y)

modelS = LinearRegression().fit(XS, y)

modelJ = LinearRegression().fit(XJ, y)

models\_at\_t\_1 = {"modelA": modelA, "model": model, "modelQ": modelQ, "modelF": modelF, "modelC": modelC,

"modelS": modelS, "modelJ": modelJ}

# Below we retrieve R^2 & Adjusted R^2, prior to out-of-sample forecasts

r\_squareds = {"R-squared": [], "Adj.R-squared": []}

for val in models\_at\_t\_1.values():

r\_squareds["R-squared"].append(val.score(XA, y))

r\_squareds["Adj.R-squared"].append(1 - (1 - val.score(XA, y)) \* ((in\_sample - lag - 1) / (in\_sample - lag - 3)))

44) MISTAKE in original R code from to generate the HAR features from <https://github.com/jacob-hein/HAR-models-forecasting-realized-volatility-in-US-stocks/tree/main>

Ans: The prediction is done on the last data point which is also used in training so training should not use the last data point that is reserved for prediction.

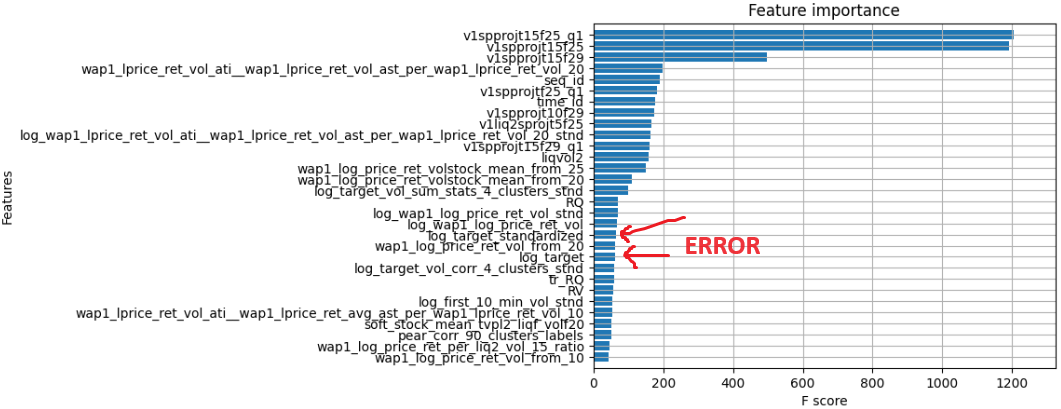
45) How far should be the lookback period in the multivariate GARCH and HAR models?

Ans: As we are predicting for short time periods e.g. 10 mins. We should limit our look back to recent past e.g. 1 day or 5 days at max. This is because high frequency volatility are more affected by closest events than far away events.

5 days gives more data points to fit the model.

46) Why are results better than the best kaggle result?

Because the target was leaked in the features set.



47) How to speed up code from 20 mins to 2 mins?

Inefficient code as below:

##### Create trade\_price\_n\_wap1\_deviation feature ######

##### extract trade\_price data #####

subset\_paths = sorted(glob.glob('/home/optimusprime/Desktop/peeterson/optiver/Optiver-Realized-Volatility-Prediction/data/trade\_train'+'.parquet/stock\_id=\*'),key=lambda x: int(x.split('=')[-1].split('.')[0]))

trade\_price\_df = pd.DataFrame(columns=['stock\_id', 'time\_id', 'seconds\_in\_bucket' 'trade\_price' ])

for path in subset\_paths:

st\_id = int(path.split('/stock\_id=')[1])

trade\_train\_st = pd.read\_parquet(path)

trade\_train\_st['stock\_id'] = st\_id

trade\_price\_df = pd.concat([trade\_price\_df, trade\_train\_st[['stock\_id','time\_id', 'seconds\_in\_bucket' , 'price']].rename(columns={'price':'trade\_price'})], axis=0)

subset\_paths = sorted(glob.glob('/home/optimusprime/Desktop/peeterson/optiver/Optiver-Realized-Volatility-Prediction/data/book\_train'+'.parquet/stock\_id=\*'), key=lambda x: int(x.split('=')[-1].split('.')[0]))

book\_price\_df = pd.DataFrame(columns=['stock\_id', 'time\_id', 'seconds\_in\_bucket', 'wap1\_price' ])

for path in subset\_paths:

st\_id = int(path.split('/stock\_id=')[1])

book\_train\_st = pd.read\_parquet(path)

book\_train\_st['stock\_id'] = st\_id

book\_train\_st['wap1\_price'] = ( book\_train\_st['bid\_price1'] \* book\_train\_st['ask\_size1'] + book\_train\_st['ask\_price1'] \* book\_train\_st['bid\_size1'] ) / (book\_train\_st['bid\_size1'] + book\_train\_st['ask\_size1'])

book\_price\_df = pd.concat([book\_price\_df, book\_train\_st[['stock\_id','time\_id', 'seconds\_in\_bucket' , 'wap1\_price']]], axis=0)

##### merge trade\_price and book\_price data #####

######## Caclulate trade\_price\_n\_wap1\_deviation ##########

temp\_df = pd.DataFrame(columns=['stock\_id', 'time\_id', 'seconds\_in\_bucket', 'ratio' ])

trade\_price\_n\_wap1\_deviation = pd.DataFrame(columns=['stock\_id', 'time\_id', 'ratio' ])

for st in unique\_stock\_ids:

trade\_price\_st = trade\_price\_df[trade\_price\_df['stock\_id'] == st]

book\_price\_st = book\_price\_df[book\_price\_df['stock\_id'] == st]

trade\_price\_st['time\_id'] = trade\_price\_st['time\_id'].astype(int)

book\_price\_st['time\_id'] = book\_price\_st['time\_id'].astype(int)

book\_price\_st['seconds\_in\_bucket'] = book\_price\_st['seconds\_in\_bucket'].astype(int)

trade\_price\_st['seconds\_in\_bucket'] = trade\_price\_st['seconds\_in\_bucket'].astype(int)

merged = trade\_price\_st.merge(book\_price\_st, on=['stock\_id', 'time\_id', 'seconds\_in\_bucket'], how='inner')

## \*\*0.5 transformation is done to make the ratio more normal

merged['ratio'] = (merged['wap1\_price'] / merged['trade\_price'])\*\*0.5

temp\_df = merged[['stock\_id', 'time\_id', 'ratio']]

temp\_df1 = temp\_df.groupby(['stock\_id', 'time\_id']).apply(lambda x: np.nanstd(x['ratio'])).reset\_index().rename(columns={0:'ratio'})

trade\_price\_n\_wap1\_deviation = pd.concat([trade\_price\_n\_wap1\_deviation, temp\_df1], axis=0)

####### Calculate correlation between trade\_price\_n\_wap1\_deviation and target ########

merged\_df = train.merge(trade\_price\_n\_wap1\_deviation, on=['stock\_id', 'time\_id'], how='left').ffill().bfill()

corr\_df = merged\_df.groupby('stock\_id').apply(lambda x: np.corrcoef(x['ratio'], x['target'])[0,1])

EFficient faster code as below:

import glob

import pandas as pd

import numpy as np

from joblib import Parallel, delayed

import numba as nb

# Define the Numba-accelerated function for WAP1 price calculation

@nb.njit

def compute\_wap1(bid\_price1, ask\_price1, bid\_size1, ask\_size1):

return (bid\_price1 \* ask\_size1 + ask\_price1 \* bid\_size1) / (bid\_size1 + ask\_size1)

# Function to process trade parquet files

def process\_trade\_file(path):

st\_id = int(path.split('/stock\_id=')[1])

trade\_train\_st = pd.read\_parquet(path)

trade\_train\_st['stock\_id'] = st\_id

return trade\_train\_st[['stock\_id', 'time\_id', 'seconds\_in\_bucket', 'price']].rename(columns={'price': 'trade\_price'})

# Function to process book parquet files

def process\_book\_file(path):

st\_id = int(path.split('/stock\_id=')[1])

book\_train\_st = pd.read\_parquet(path)

book\_train\_st['stock\_id'] = st\_id

# Apply Numba-accelerated WAP1 calculation

book\_train\_st['wap1\_price'] = compute\_wap1(

book\_train\_st['bid\_price1'].values,

book\_train\_st['ask\_price1'].values,

book\_train\_st['bid\_size1'].values,

book\_train\_st['ask\_size1'].values

)

return book\_train\_st[['stock\_id', 'time\_id', 'seconds\_in\_bucket', 'wap1\_price']]

# Get the list of trade and book parquet file paths

trade\_paths = sorted(glob.glob('/home/optimusprime/Desktop/peeterson/optiver/Optiver-Realized-Volatility-Prediction/data/trade\_train'+'.parquet/stock\_id=\*'), key=lambda x: int(x.split('=')[-1].split('.')[0]))

book\_paths = sorted(glob.glob('/home/optimusprime/Desktop/peeterson/optiver/Optiver-Realized-Volatility-Prediction/data/book\_train'+'.parquet/stock\_id=\*'), key=lambda x: int(x.split('=')[-1].split('.')[0]))

# Use joblib's Parallel and delayed to process trade files in parallel

trade\_dfs = Parallel(n\_jobs=-1)(delayed(process\_trade\_file)(path) for path in trade\_paths)

# Use joblib's Parallel and delayed to process book files in parallel

book\_dfs = Parallel(n\_jobs=-1)(delayed(process\_book\_file)(path) for path in book\_paths)

# Concatenate the DataFrames into the final DataFrames

trade\_price\_df = pd.concat(trade\_dfs, axis=0)

book\_price\_df = pd.concat(book\_dfs, axis=0)

##### merge trade\_price and book\_price data and calculate trade\_price\_n\_wap1\_deviation #####

trade\_price\_n\_wap1\_deviation = pd.DataFrame()

# Group by 'stock\_id' and perform operations in one go

for st, trade\_price\_st in trade\_price\_df.groupby('stock\_id'):

book\_price\_st = book\_price\_df[book\_price\_df['stock\_id'] == st]

# Ensure consistent types for merging using .loc to avoid SettingWithCopyWarning

trade\_price\_st.loc[:, 'time\_id'] = trade\_price\_st['time\_id'].astype(int)

book\_price\_st.loc[:, 'time\_id'] = book\_price\_st['time\_id'].astype(int)

trade\_price\_st.loc[:, 'seconds\_in\_bucket'] = trade\_price\_st['seconds\_in\_bucket'].astype(int)

book\_price\_st.loc[:, 'seconds\_in\_bucket'] = book\_price\_st['seconds\_in\_bucket'].astype(int)

# Merge trade and book data on 'stock\_id', 'time\_id', 'seconds\_in\_bucket'

merged = trade\_price\_st.merge(book\_price\_st, on=['stock\_id', 'time\_id', 'seconds\_in\_bucket'], how='inner')

# Calculate the ratio with vectorized operations

merged['ratio'] = (merged['wap1\_price'] / merged['trade\_price'])\*\*0.5

# Group by 'stock\_id' and 'time\_id', and calculate standard deviation of 'ratio'

temp\_df1 = merged.groupby(['stock\_id', 'time\_id'])['ratio'].std().reset\_index()

# Append to the final DataFrame

trade\_price\_n\_wap1\_deviation = pd.concat([trade\_price\_n\_wap1\_deviation, temp\_df1], axis=0)

####### Calculate correlation between trade\_price\_n\_wap1\_deviation and target ########

merged\_df = train.merge(trade\_price\_n\_wap1\_deviation, on=['stock\_id', 'time\_id'], how='left').ffill().bfill()

corr\_df = merged\_df.groupby('stock\_id').apply(lambda x: np.corrcoef(x['ratio'], x['target'])[0,1])

48) warning for the following code.

all\_stocks\_first\_10\_min\_vol\_df = pd.DataFrame()

#all\_stocks\_first\_10\_min\_vol\_df['time\_id'] = all\_uniq\_time\_ids['time\_id']

for st\_id in unique\_stock\_ids:

st\_df = pd.DataFrame()

subset = first\_10\_min\_vol\_df[first\_10\_min\_vol\_df['stock\_id'] == st\_id]

st\_df['time\_id'] = subset['time\_id']

st\_df['first\_10\_min\_vol'] = subset['first\_10\_min\_vol']

st\_df = all\_uniq\_time\_ids.merge(st\_df, on='time\_id', how='left').ffill().bfill()

all\_stocks\_first\_10\_min\_vol\_df[st\_id] = st\_df['first\_10\_min\_vol']

all\_stocks\_first\_10\_min\_vol\_df = all\_stocks\_first\_10\_min\_vol\_df.to\_numpy()[:,:,np.newaxis] # change dimension from 380 x 112 to 3830 x 112 x 1

[/tmp/ipykernel\_62256/3880773615.py:12](about:blank): PerformanceWarning: DataFrame is highly fragmented. This is usually the result of calling `frame.insert` many times, which has poor performance. Consider joining all columns at once using pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe = frame.copy()`

all\_stocks\_first\_10\_min\_vol\_df[st\_id] = st\_df['first\_10\_min\_vol']

ANS: The warning you encountered is due to the fact that adding columns one by one to a DataFrame can cause it to become fragmented, leading to poor performance. To avoid this issue, you should collect all the data in a list and then create the DataFrame in one go, which will be more efficient. Here's how you can modify your code to do that:

import numpy as np

import pandas as pd

# Initialize an empty list to hold the data for each stock

all\_stocks\_data = []

# Iterate through each stock ID and collect the data

for st\_id in unique\_stock\_ids:

subset = first\_10\_min\_vol\_df[first\_10\_min\_vol\_df['stock\_id'] == st\_id]

st\_df = pd.DataFrame({'time\_id': subset['time\_id'], 'first\_10\_min\_vol': subset['first\_10\_min\_vol']})

# Merge with all unique time IDs and handle missing data

st\_df = all\_uniq\_time\_ids.merge(st\_df, on='time\_id', how='left').ffill().bfill()

# Append the first\_10\_min\_vol column to the list

all\_stocks\_data.append(st\_df['first\_10\_min\_vol'].to\_numpy())

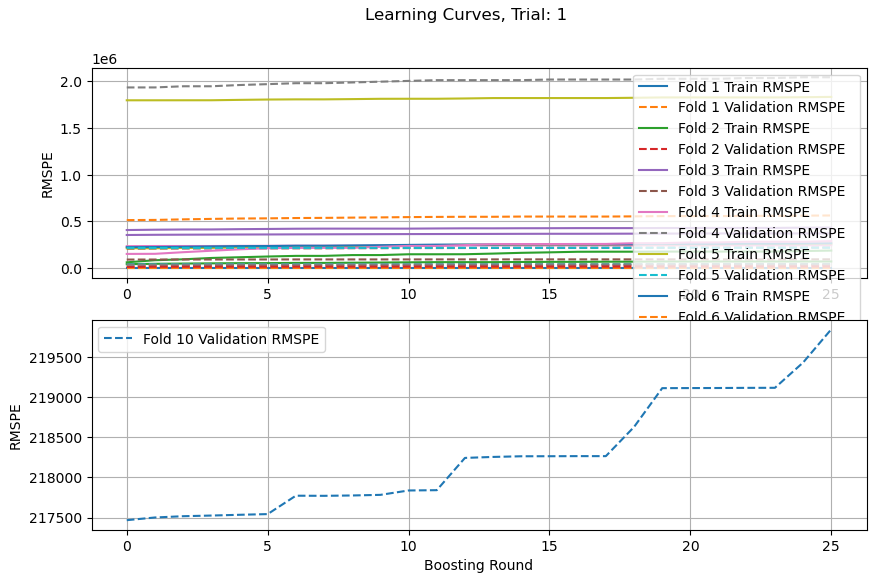
# Convert the list to a NumPy array and then reshape it

all\_stocks\_first\_10\_min\_vol\_array = np.stack(all\_stocks\_data, axis=1)

# Change dimension from 380 x 112 to 3830 x 112 x 1

all\_stocks\_first\_10\_min\_vol\_array = all\_stocks\_first\_10\_min\_vol\_array[:, :, np.newaxis]

49) Using a dataframe sorted by stock id and clustering excluded features results in learning difficulty. TRoubleshooting it!



Ans: some features were sorted but others were not so there were some inconsistencies

[I 2024-08-23 08:55:48,542] A new study created in memory with name: Correct\_residual\_autocorrrelation\_HAR\_feat

Fold: 1

Training....

fold: 1, val rmspe score is 203.36553799214525

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.8200060709976382

log(corr( )) 0.9183503248765418

Fold: 2

Training....

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

fold: 2, val rmspe score is 314.81614810727723

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7275441209881873

log(corr( )) 0.8638870320403546

Fold: 3

Training....

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

fold: 3, val rmspe score is 245.78218652765113

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7499924717390395

log(corr( )) 0.8723478702583103

Fold: 4

Training....

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

fold: 4, val rmspe score is 214.24469431884245

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7879025102952958

log(corr( )) 0.8678673512532091

Fold: 5

Training....

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

fold: 5, val rmspe score is 199.08945361816936

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7426435991710113

log(corr( )) 0.8608028754768562

Fold: 6

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

Training....

fold: 6, val rmspe score is 185.70334709169293

corr(p/v1v, y\_val/v1v) nan

log(corr( )) -4.863593035253185e-16

corr(p, y\_val) 0.8198235128104577

log(corr( )) 0.8869274688571928

Fold: 7

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

Training....

fold: 7, val rmspe score is 187.2172061545259

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.8142522875169197

log(corr( )) 0.8935063429565483

Fold: 8

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

Training....

fold: 8, val rmspe score is 195.9261276474658

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.8425376578364752

log(corr( )) 0.9120976526770079

Fold: 9

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

Training....

fold: 9, val rmspe score is 192.20249571487852

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7973113169937757

log(corr( )) 0.8702149761981036

Fold: 10

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]

Training....

fold: 10, val rmspe score is 210.2526097421195

corr(p/v1v, y\_val/v1v) nan

log(corr( )) nan

corr(p, y\_val) 0.7736030147676815

log(corr( )) 0.8838140210359474

mean rmspe val score over 10 splits is 214.8599806914768

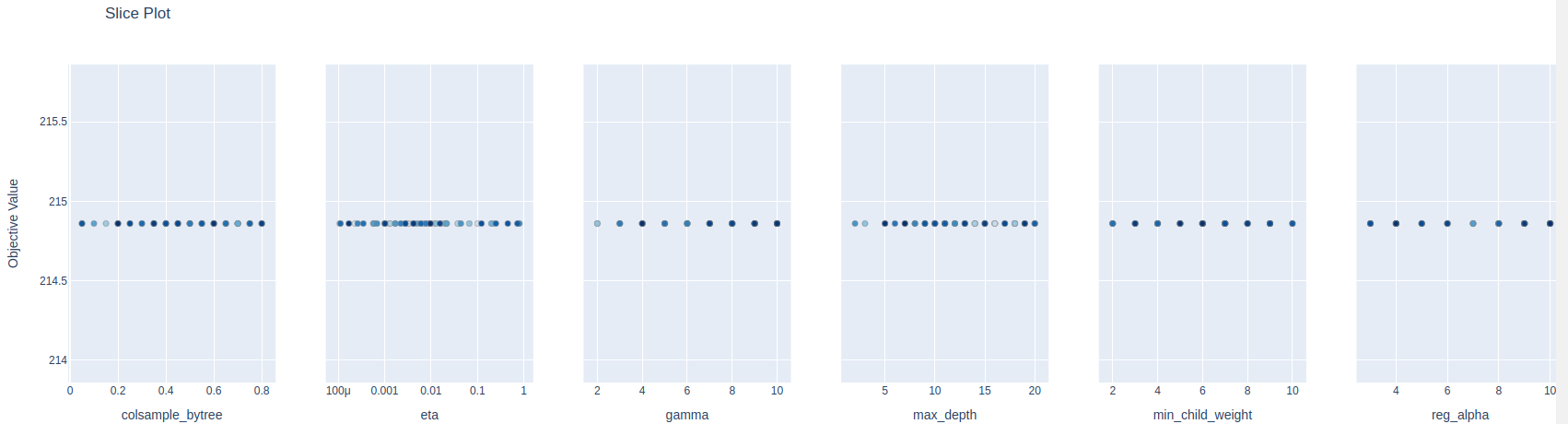
/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2897: RuntimeWarning: invalid value encountered in divide

c /= stddev[:, None]

/usr/local/lib/python3.10/dist-packages/numpy/lib/function\_base.py:2898: RuntimeWarning: invalid value encountered in divide

c /= stddev[None, :]





RuntimeError Traceback (most recent call last)

<ipython-input-23-086d06c1ebd5> in <cell line: 40>()

38 fig.show()

39

---> 40 fig = optuna.visualization.plot\_param\_importances(study)

41 fig.show()

42

5 frames

/usr/local/lib/python3.10/dist-packages/optuna/importance/\_fanova/\_fanova.py in fit(self, X, y, search\_spaces, column\_to\_encoded\_columns)

75 # If all trees have 0 variance, we cannot assess any importances.

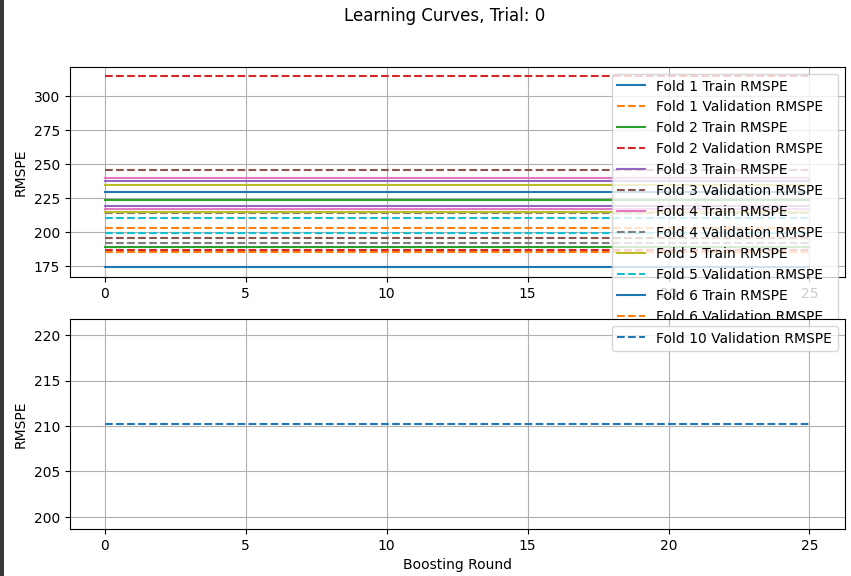
76 # This could occur if for instance `X.shape[0] == 1`.

---> 77 raise RuntimeError("Encountered zero total variance in all trees.")

78

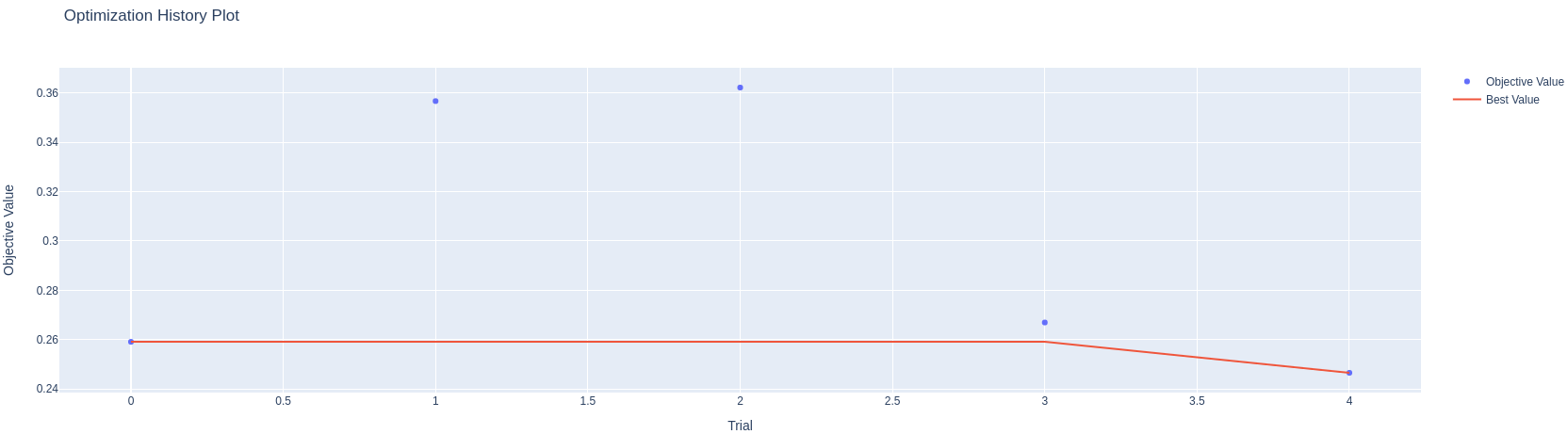
79 def get\_importance(self, feature: int) -> Tuple[float, float]:

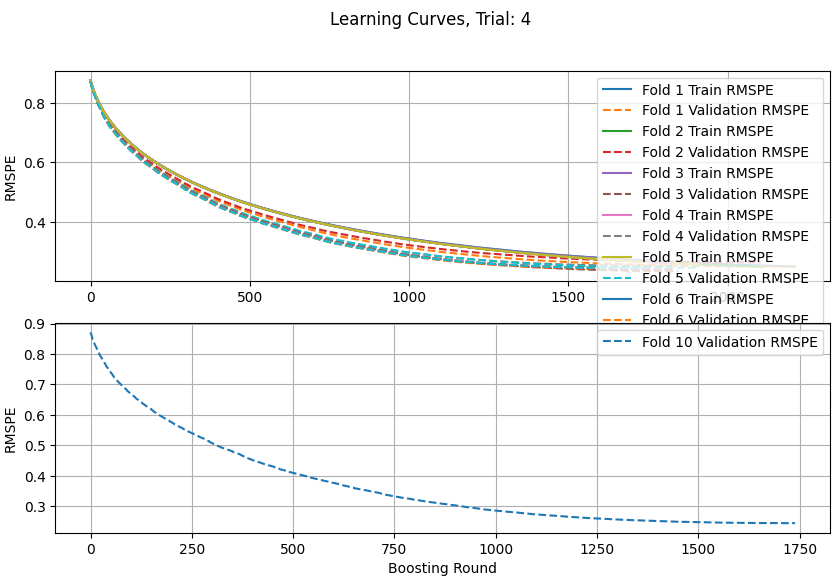
RuntimeError: Encountered zero total variance in all trees.



Regardless of parameter changes all the training are the same!! The problem could be with data transformation, handling of extreme and nan values in our data.

1. Check if jager features alone are causing any problems? train\_feat\_df\_jager.pkl
   1. Nope, ON Jager’s data alone the error is decreasing \





Best number of iteration/boosting rounds: 1714

Trial no.: 4

Value: 0.24656564491283312

Params:

max\_depth: 6

eta: 0.0025409340303260705

subsample: 0.7

colsample\_bytree: 0.1

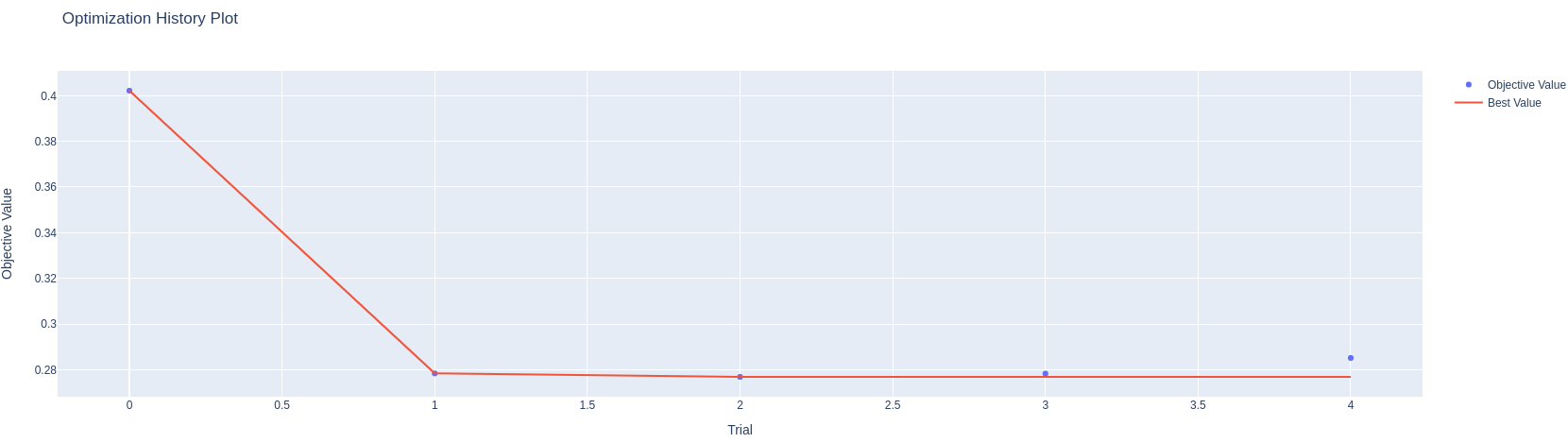
gamma: 7

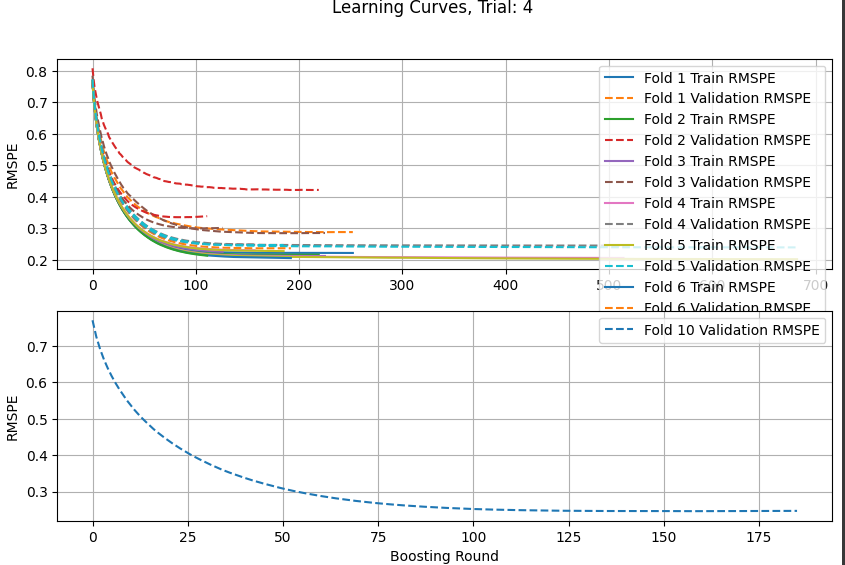
reg\_alpha: 10

reg\_lambda: 5

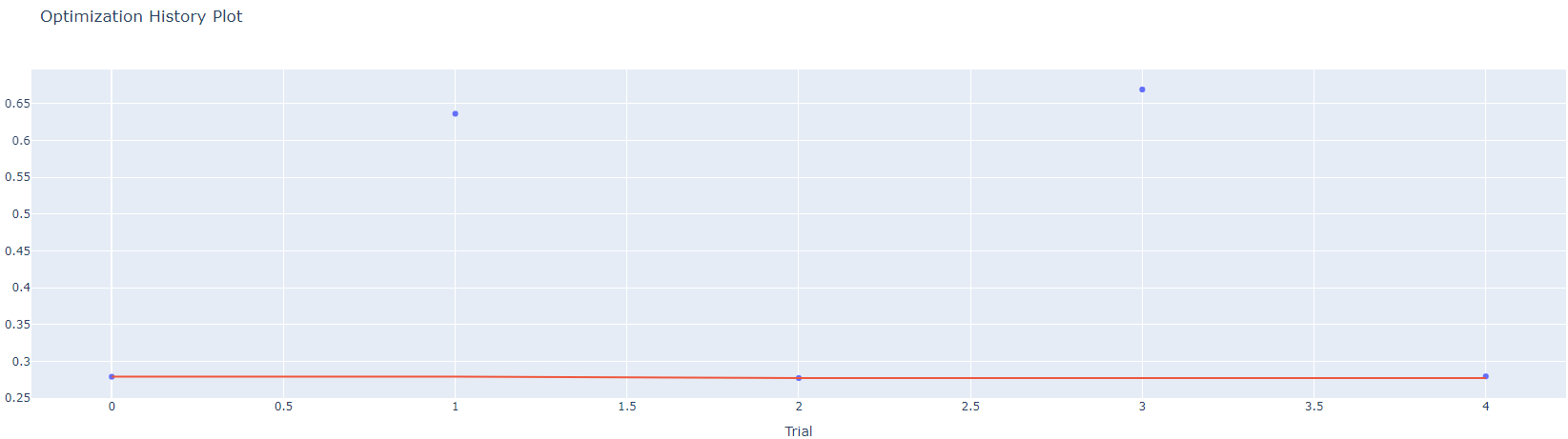
min\_child\_weight: 4

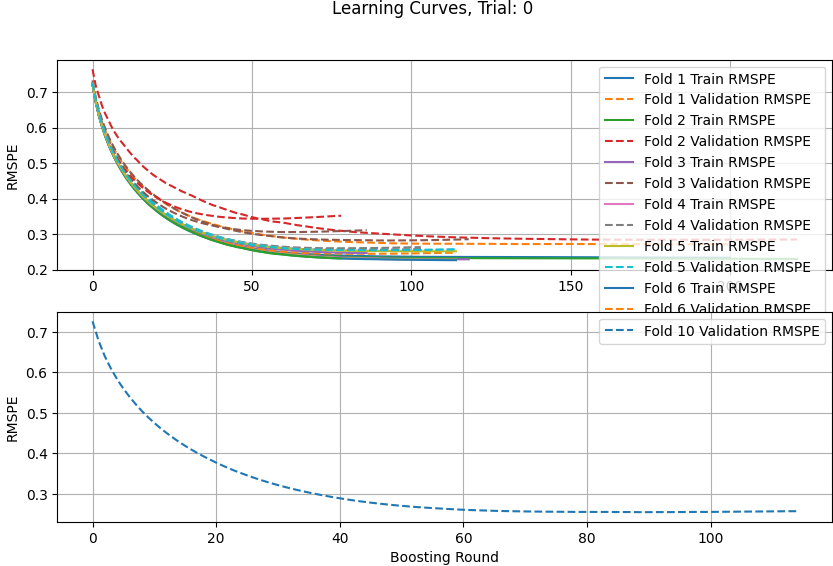
1. NExt add ONLY our features before any transformation and check where this problem occurs. train\_feat\_df\_only\_our\_feat.pkl
   1. NO problem, -np.inf and np.inf are set to -1e8 and 1e8 instead of max values. Log\_target and log\_target\_standardized are removed.

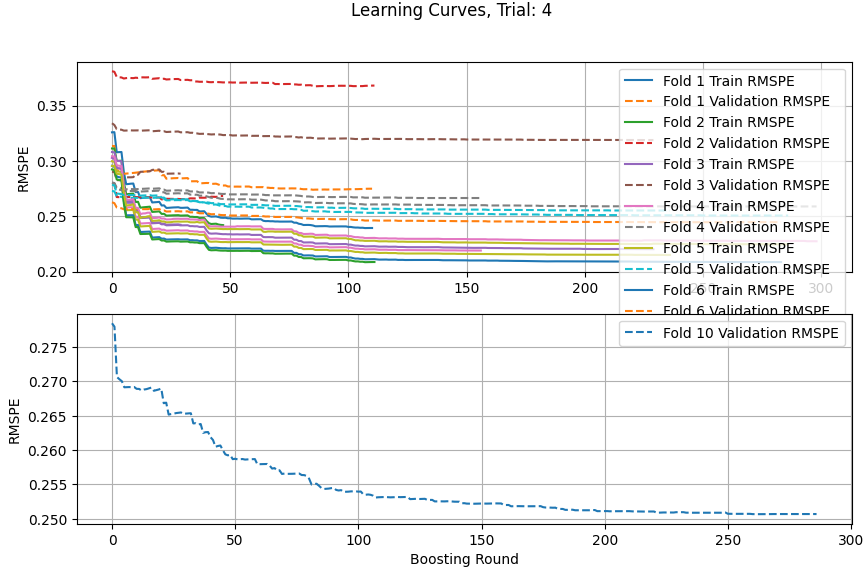




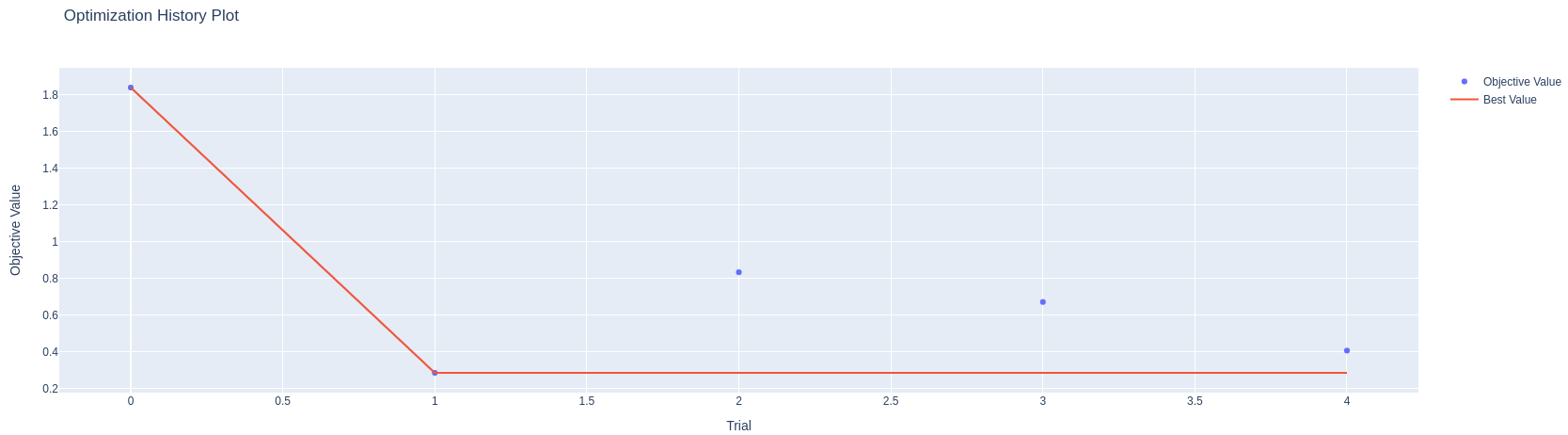
1. ONLY our numerical features before any transformation and removed all features with the names “labels” and “clusters” to remove categorical features. train\_feat\_df\_only\_our\_feat.pkl
   1. No problem. All 5 trials are decreasing

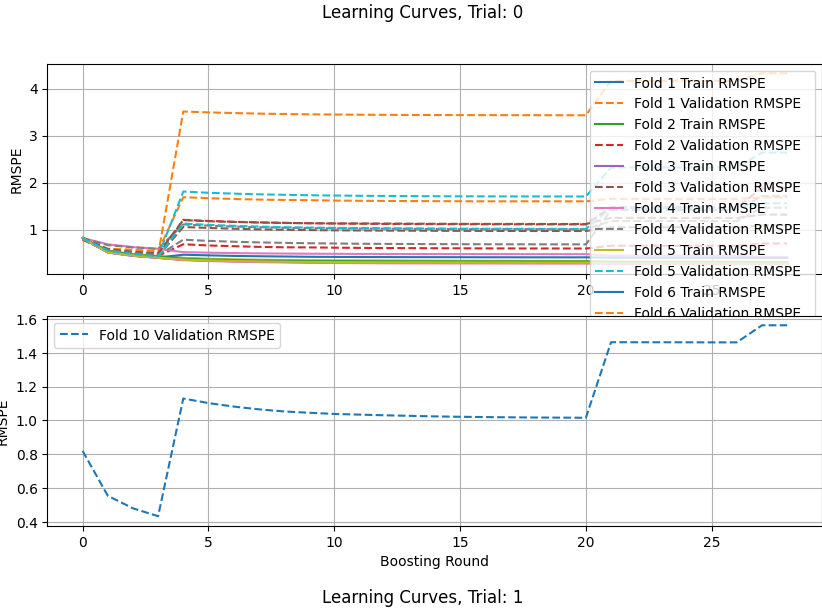


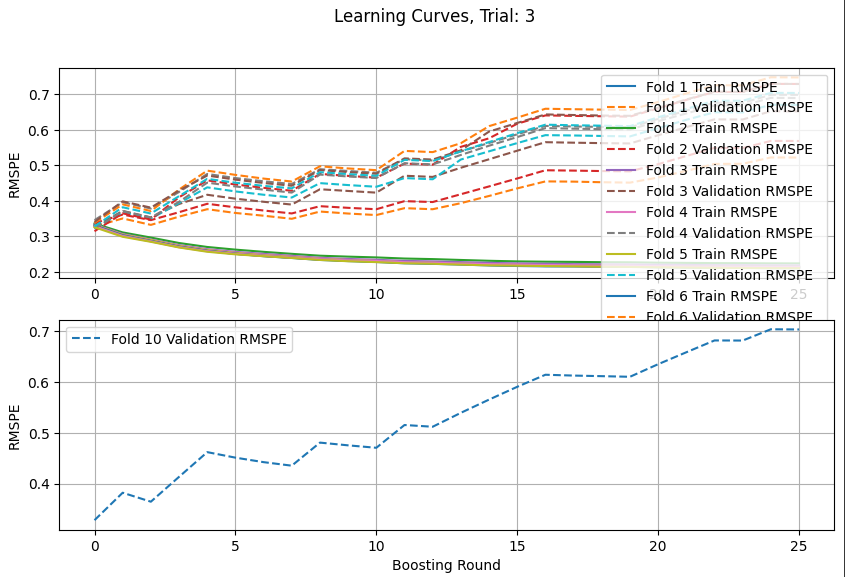




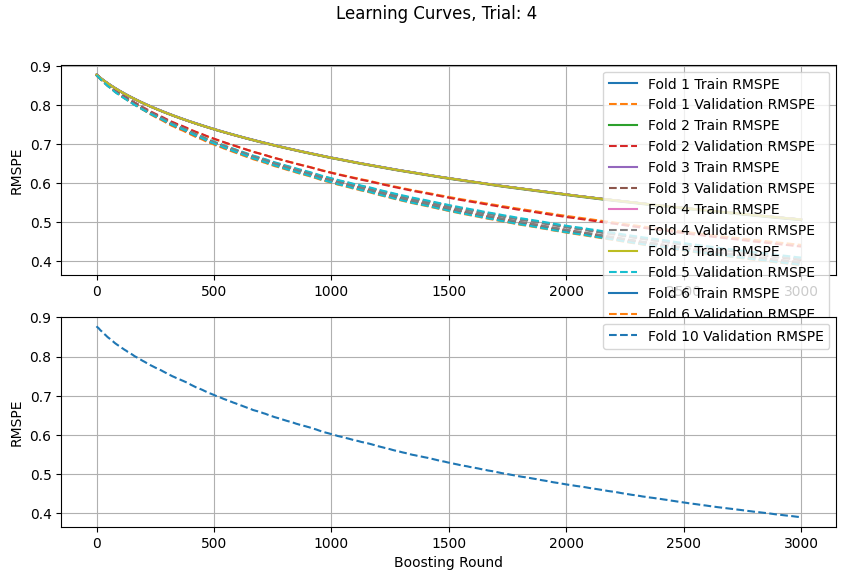
1. NExt add combine our features and jager features before any transformation and check where this problem occurs. train\_feat\_df\_added\_our\_feat.pkl
   1. NO problem, -np.inf and np.inf are set to -1e8 and 1e8 instead of max values. Log\_target and log\_target\_standardized are removed.



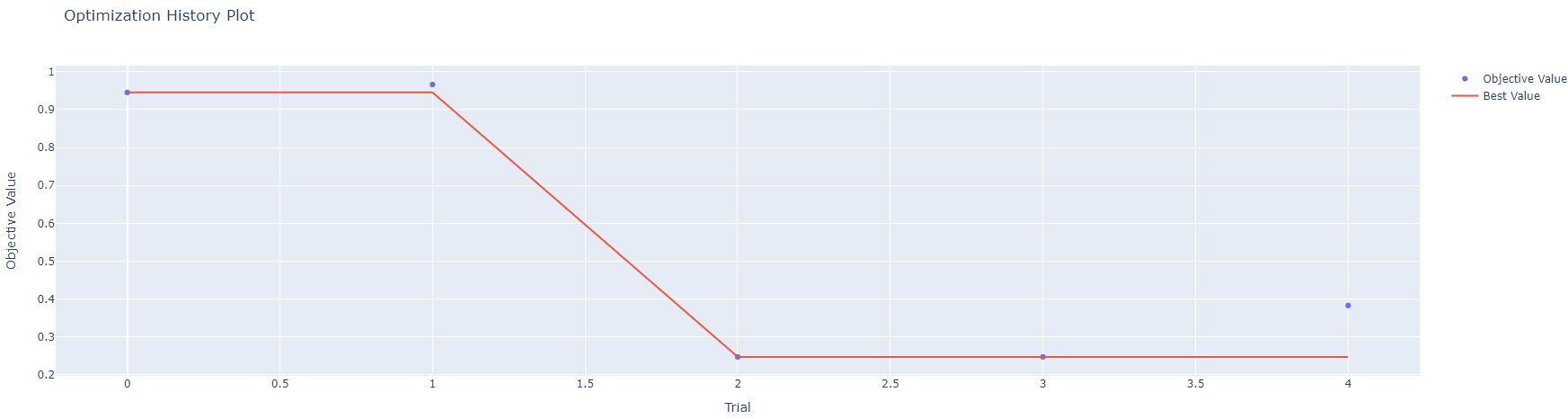
Some curves were observed like this 

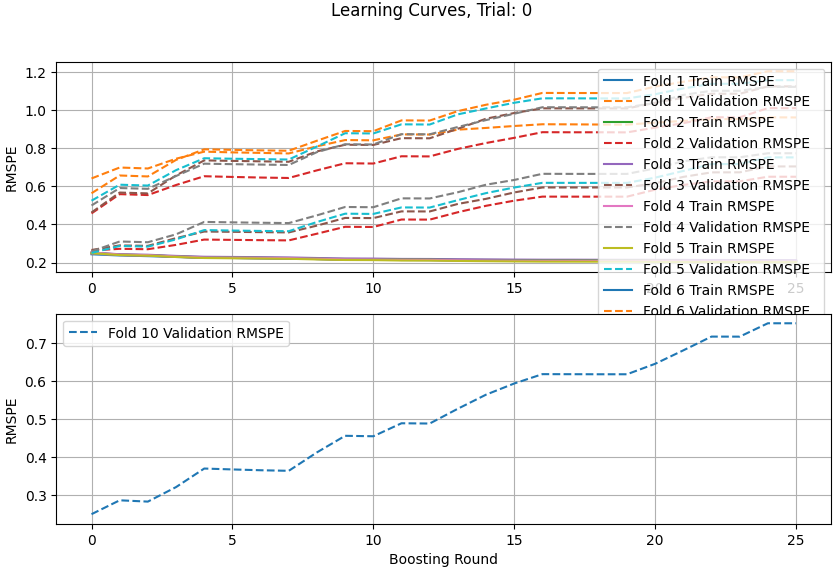


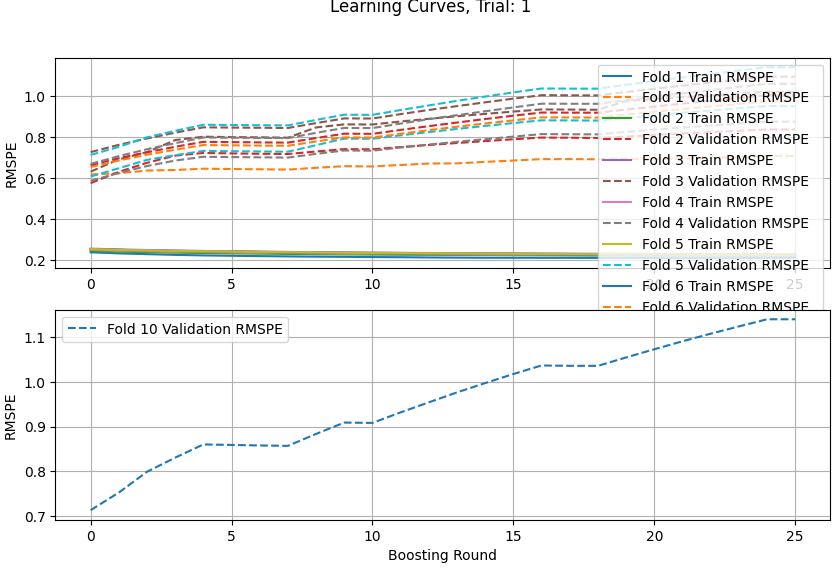
BUt mostly going down as shown below

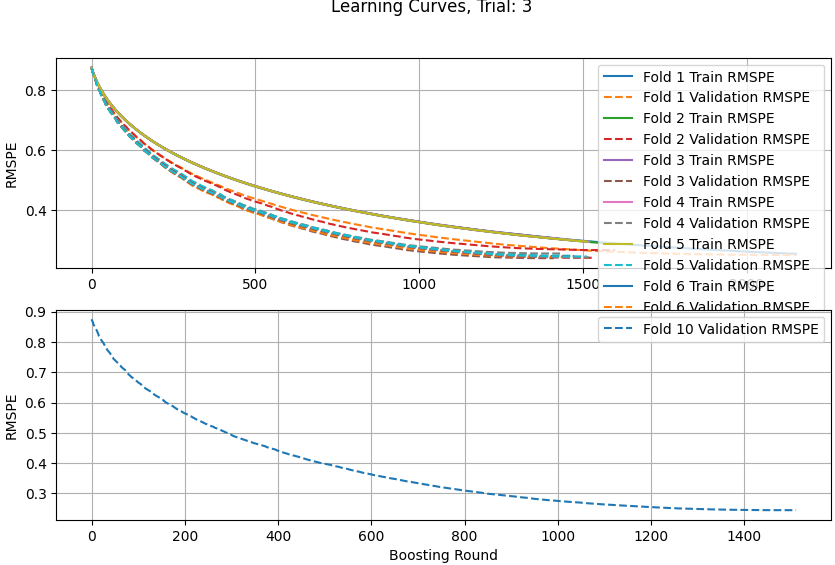


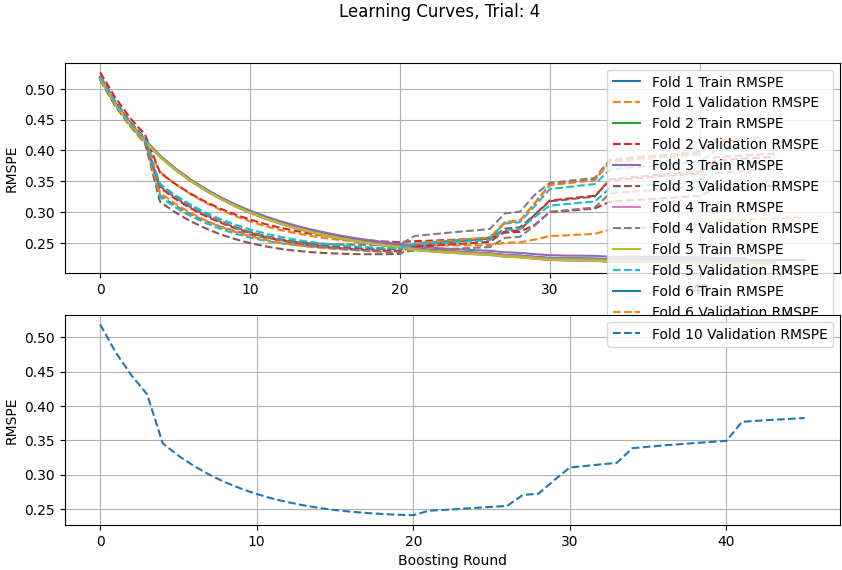
1. NExt add combine our features and jager features before any transformation and check where this problem occurs. and removed all features with the names “labels” and “clusters” to remove categorical features. train\_feat\_df\_added\_our\_feat.pkl
   1. There is some feature in our dataset with data type float that is causing learning error.





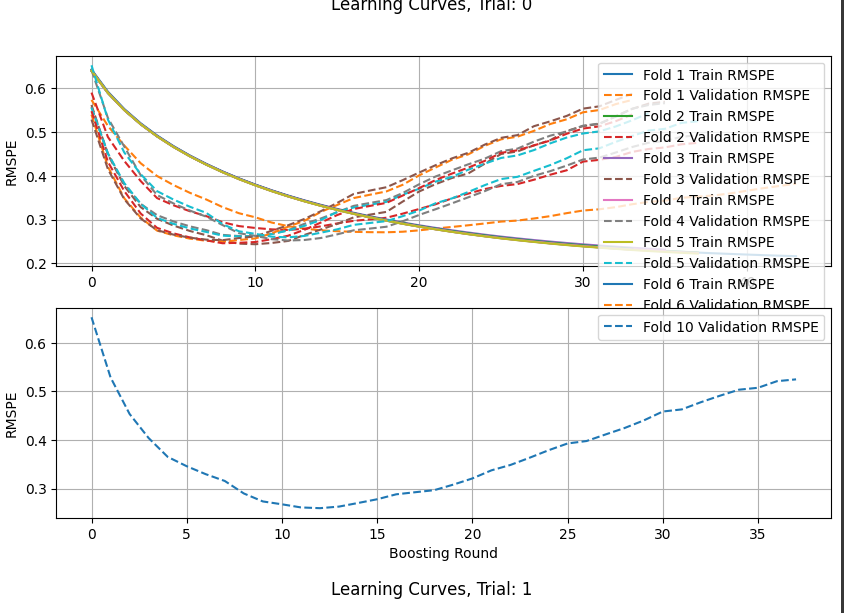


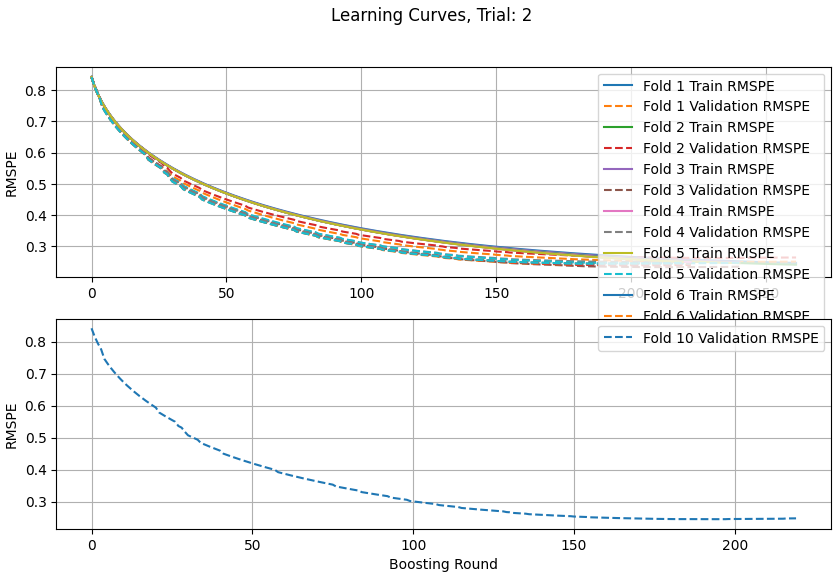


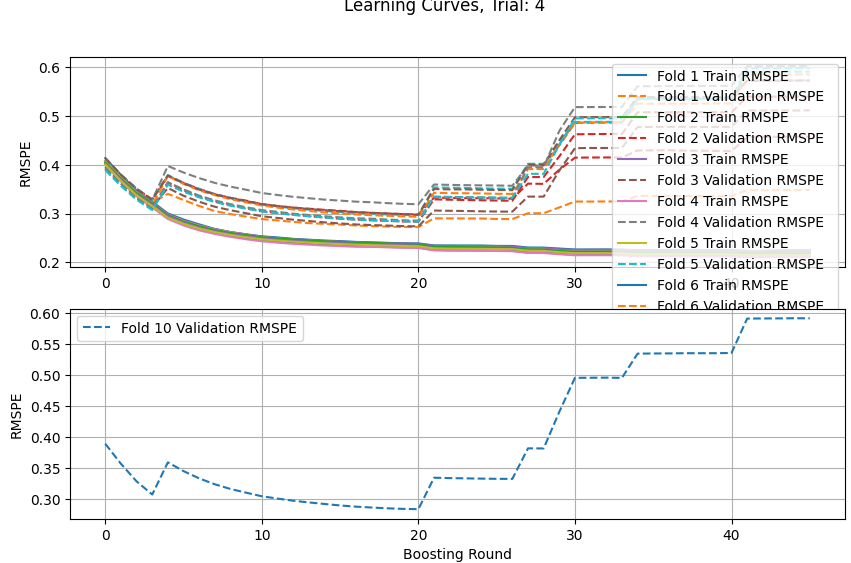


1. Replaced nan with median and inf and -inf with max and min. Used the full dataset. Including the categorical features. Train\_feat\_df\_all\_feat\_null\_inf\_replaced.pkl

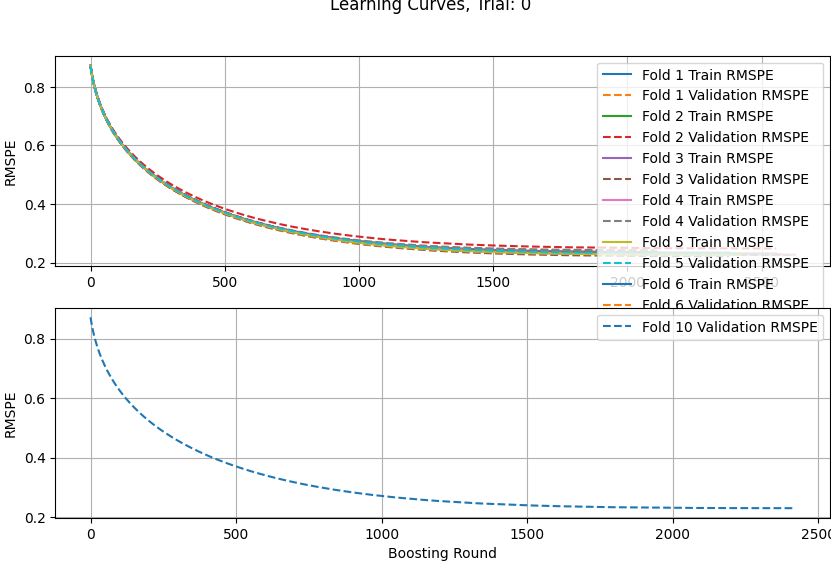




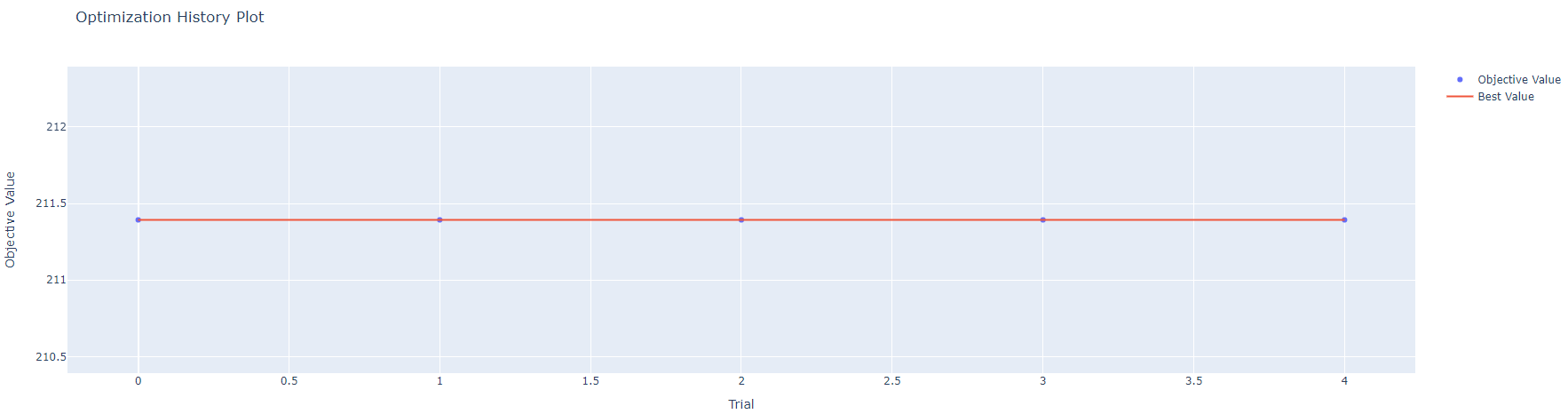


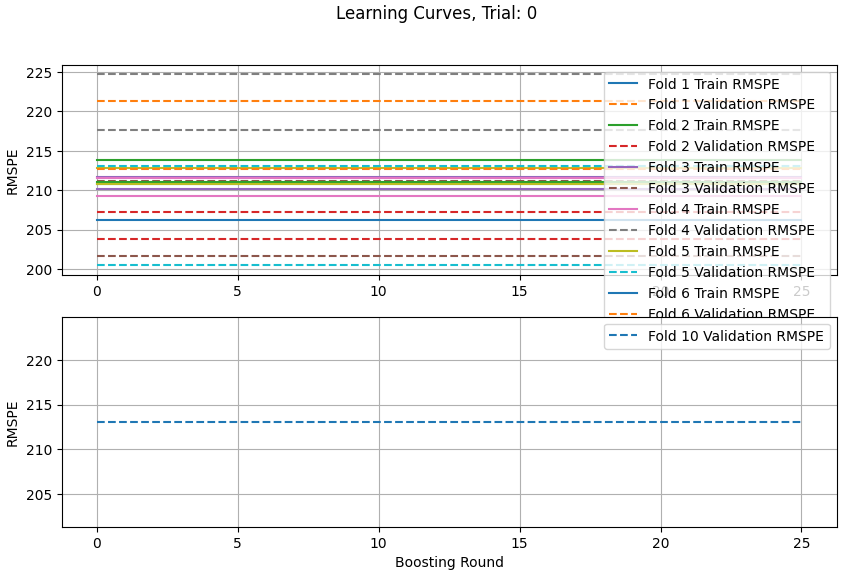


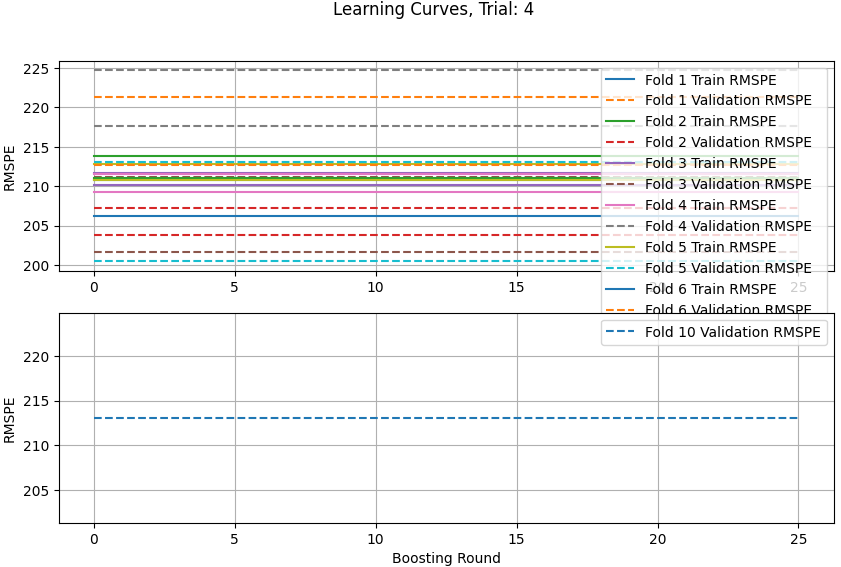
1. Removed features from train\_feat\_df\_added\_our\_feat.pkl with number of nan ,inf and -inf> 1000. Used 1e8 and -1e8 for pos and neg inf. Removed all feat with “labels” and “clusters” in name and other int features. Used this data sets. Train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced.pkl
2. After removing those features all trials are decreasing.



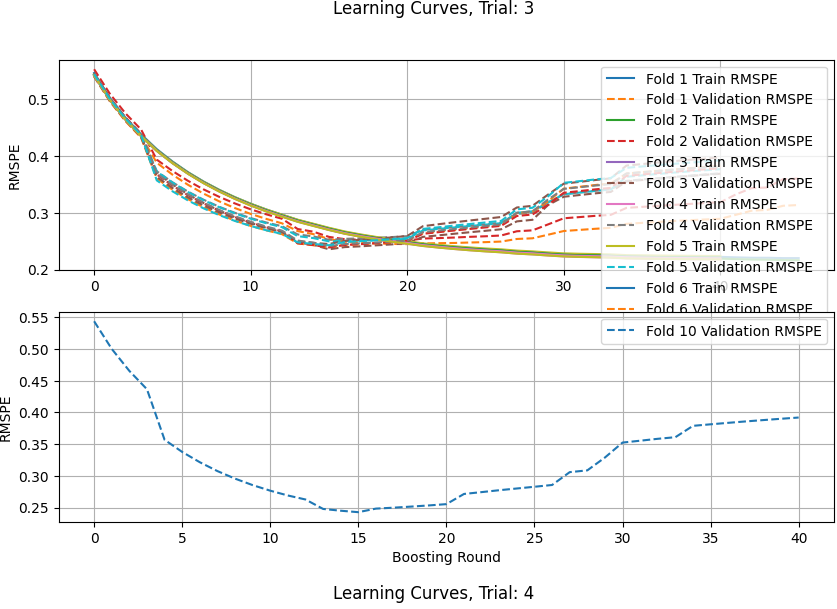
1. Try performing normalisation on Train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced.pkl
2. After time id was reordered error became very big.
3. Without time id reordering the error is ALSO high.

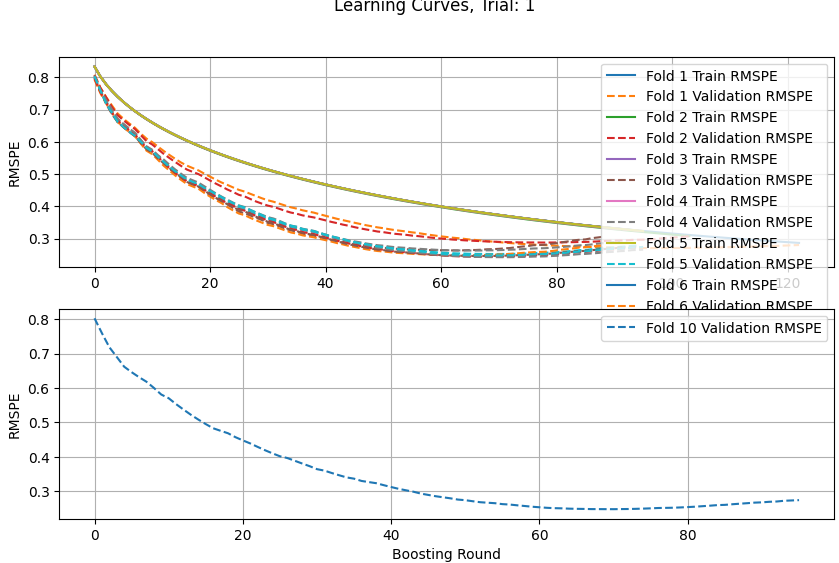




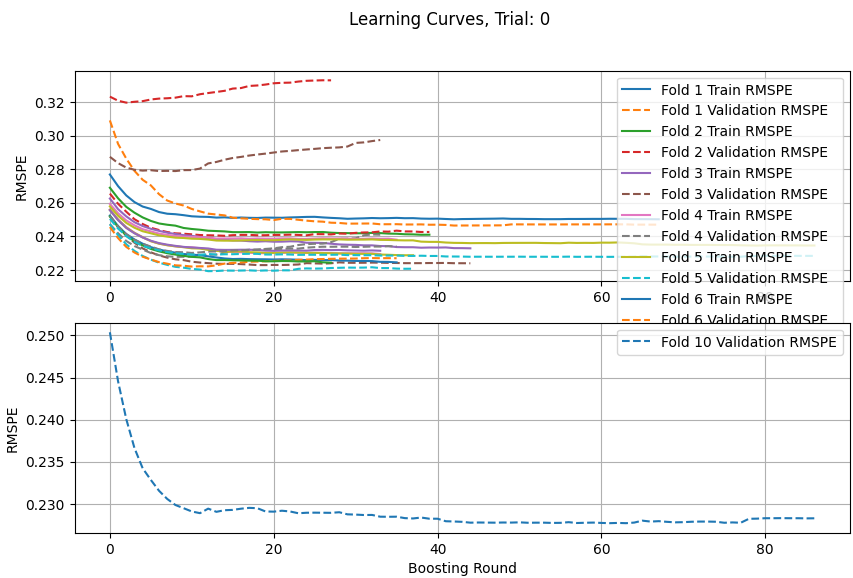


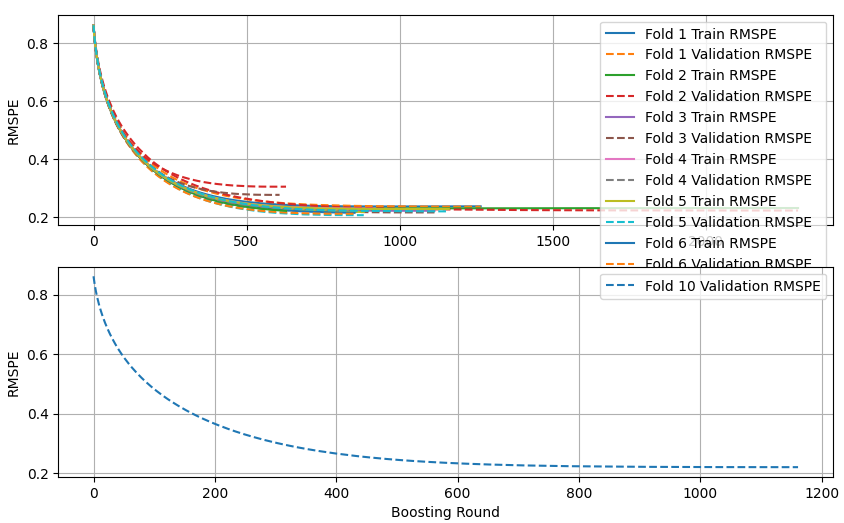
1. Remove features with name “labels” and “clusters” and other int features in the original dataset that has been log transformed but NOT standardized. Train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced\_normalized.pkl Also remove feat with > 1000 nan, inf and -inf.
2. Without time id reordering : error always decreases its ok.

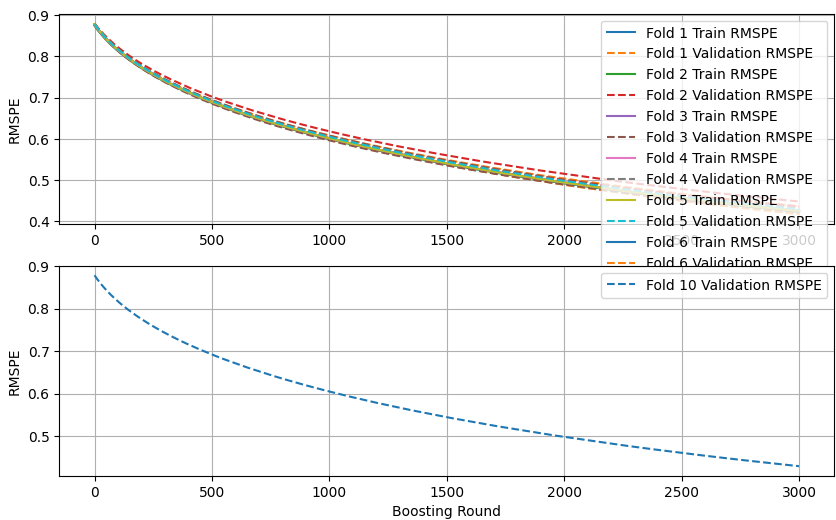




1. with time id reordering: It is Ok. Error is always decreasing

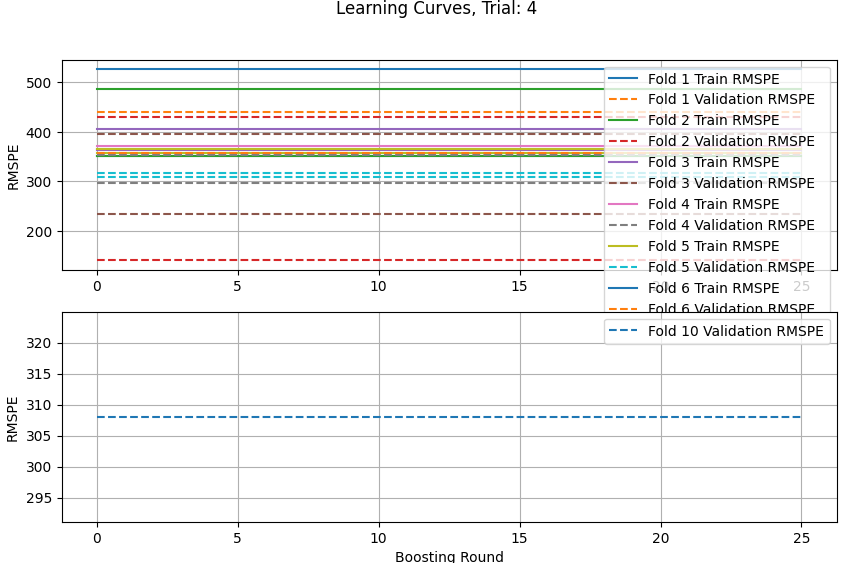
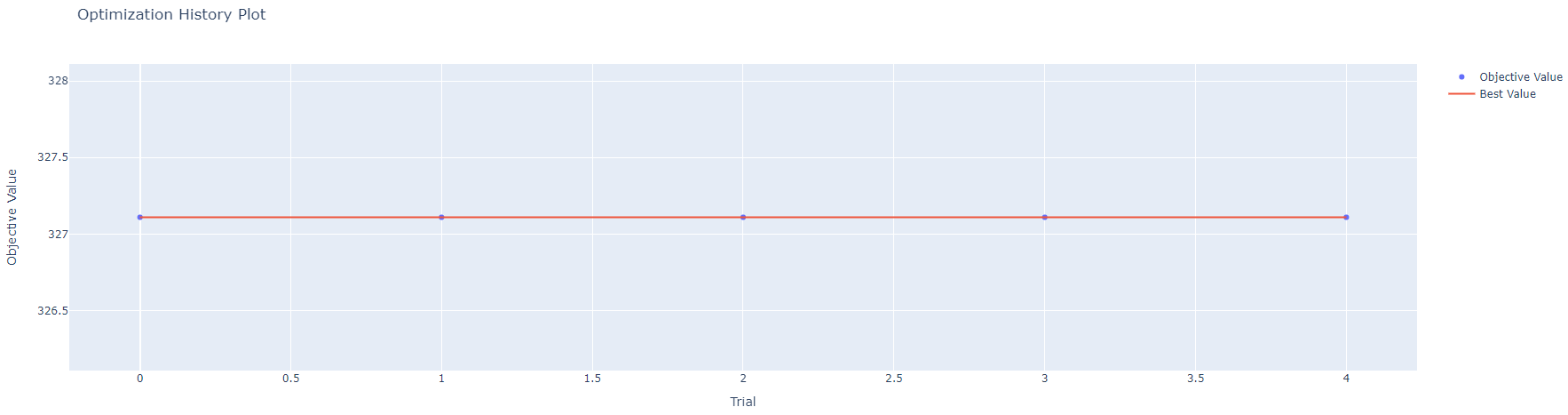






Problem is with standardization or Minmax scaling both of these are leading to errors. Transformation alone is not a problem.

1. Remove features with name “labels” and “clusters” and other int features in the original dataset that has been log transformed but NOT standardized. Train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced\_minmax.pkl Also remove feat with > 1000 nan, inf and -inf.

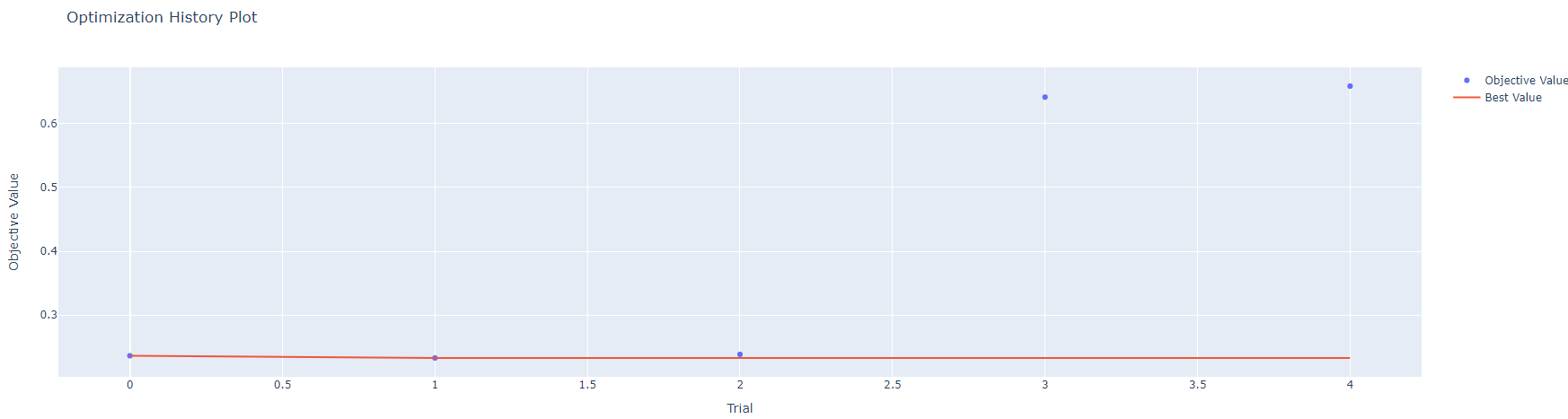


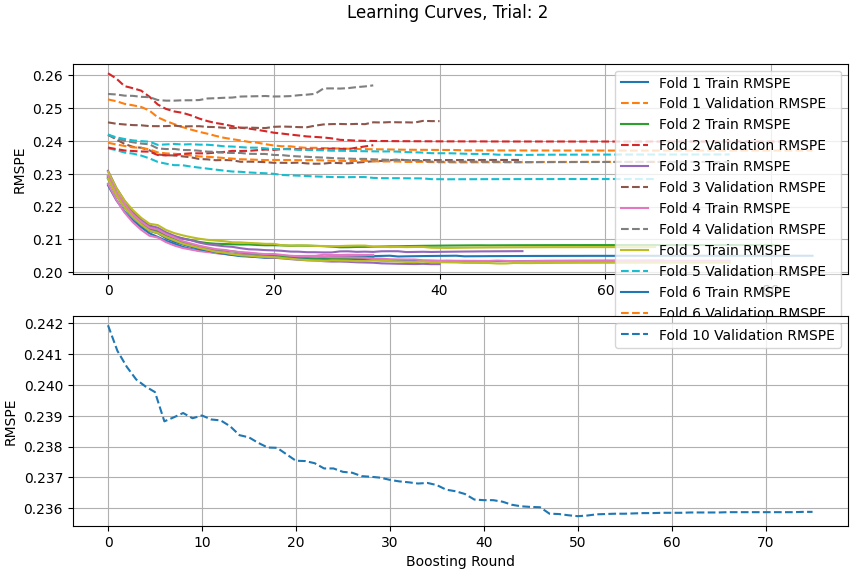
1. Replaced nan with median and inf and -inf with max and min. Used the full dataset.

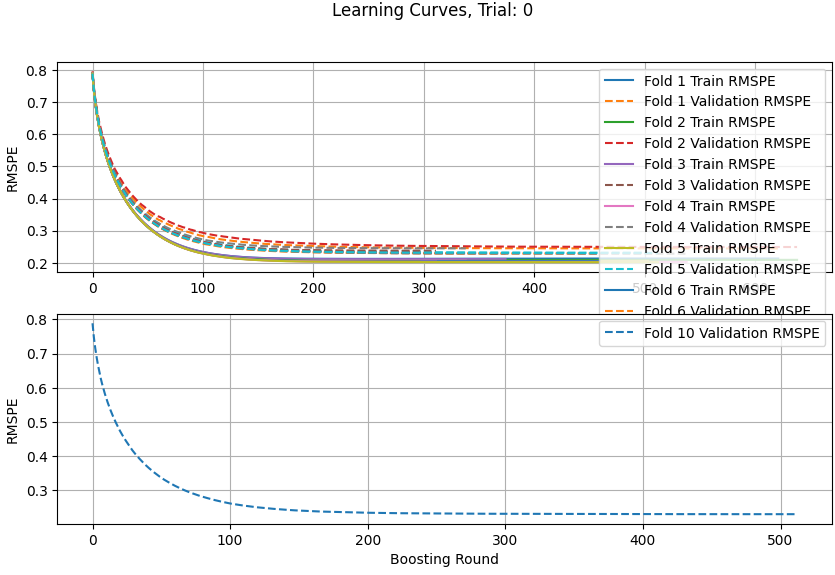
Including the categorical features and REmoved the categorical features

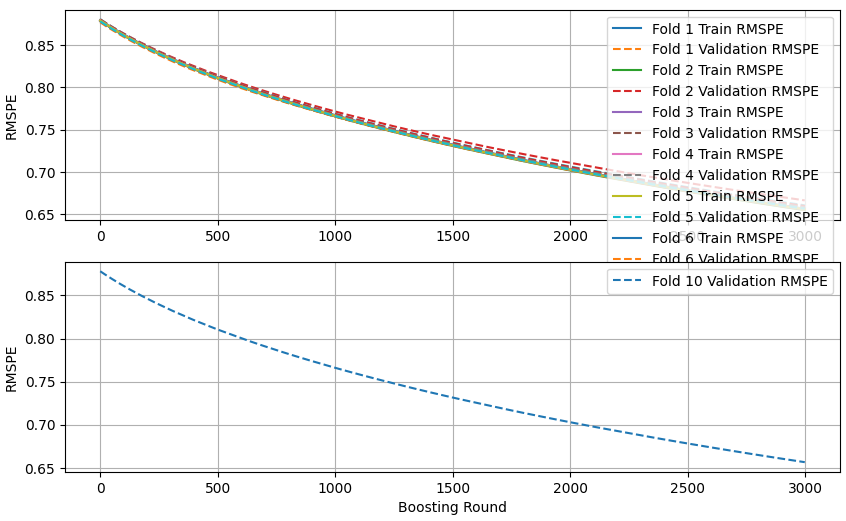
Train\_feat\_df\_all\_feat\_null\_inf\_replaced.pkl

Maybe they have a little influence but most likely our normalisation has more influence on poor training??









We use train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced\_transformed\_reordered.pkl

Or train\_feat\_df\_all\_feat\_large\_null\_neg\_posinf\_dropped\_null\_inf\_replaced\_transformed.pkl

For training.

50) Difference between features generated using full train data and partial test data.

Ans: missing values are handled differently in each stock. Trade data has 5 missing time ids. 112\* 5 missing values.

We should avoid ffill() and bfill() in test data as we dont know the order we could just leave it as np.nan after reindexing using test.csv file.

Merge trade data on stock\_id and time\_id of test.csv instead of just assigning.