Nasdaq Listed Stocks Closing Price Movements Prediction[1]

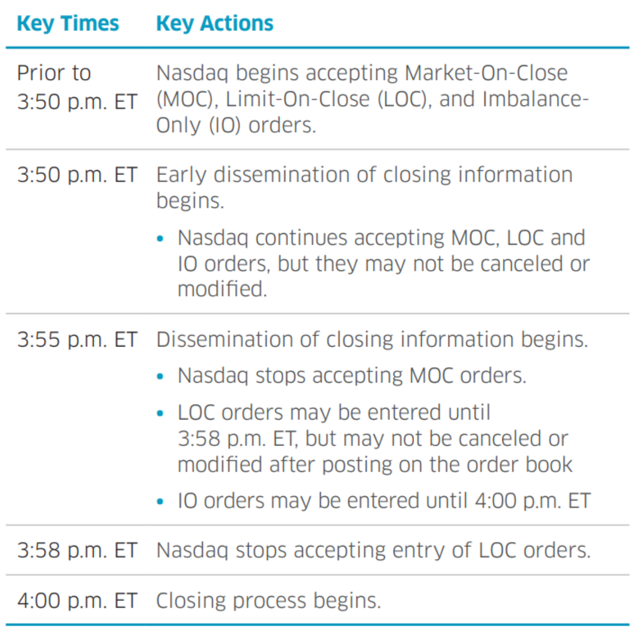
Letian Yu, December 2023
Data Science Institution, Brown University

1 Introduction

1.1 Problem Statement

The American stock exchange, *Nasdaq Stock Market*, represents one of the United States' premier stock trading venues, distinguished by its high trading volume. Each trading day on the Nasdaq Stock Exchange culminates with a unique process known as the "*Nasdaq Closing Cross Auction*". This event plays a pivotal role in establishing the *official closing prices* for stocks that are listed on the Nasdaq, which will be used as used as a reference point by investors, analysts, and financial media.

During the last ten minutes of the daily trading session, market makers will merge data from the conventional order book with that from the auction book. The integration of information from these two distinct sources is crucial in ensuring optimal pricing for all participants in the market. Insights gleaned from the auction data are subsequently instrumental in adjusting prices, evaluating the market dynamics, and identifying potential trading opportunities.



Key Times	Key Actions
Prior to 3:50 p.m. ET	Nasdaq begins accepting Market-On-Close (MOC), Limit-On-Close (LOC), and Imbalance-Only (IO) orders.
3:50 p.m. ET	Early dissemination of closing information begins. • Nasdaq continues accepting MOC, LOC and IO orders, but they may not be canceled or modified.
3:55 p.m. ET	Dissemination of closing information begins. • Nasdaq stops accepting MOC orders. • LOC orders may be entered until 3:58 p.m. ET, but may not be canceled or modified after posting on the order book • IO orders may be entered until 4:00 p.m. ET
3:58 p.m. ET	Nasdaq stops accepting entry of LOC orders.
4:00 p.m. ET	Closing process begins.

Figure 1. Cutoff Times for Nasdaq On-Close Orders

In this project, we will utilize regression machine learning models to predict the closing price movements (indicated by a specifically defined synthetic index) for hundreds of Nasdaq listed stocks. The prediction model can contribute to combining signals from the auction book and the (non-auction) order book.

[1] **GitHub Repository:** https://github.com/LetianY/data1030-optiver-trading-at-close/

1.2 Dataset and Previous Work

The dataset we use is from a public competition provided by Optiver on Kaggle [2]. It contains historic data for the daily ten-minute closing auction on the Nasdaq stock exchange. There are in total 5,237,980 rows and 17 columns in the original dataset, with the target variable being a synthetic index indicating the closing price movement.

$$target = \left(\frac{StockWap_{\{t+60\}}}{StockWAP_t} - \frac{IndexWap_{\{t+60\}}}{IndexWAP_t}\right) * 10000.$$

Besides, there are 5 identifier columns (*stock_id, date_id, seconds_in_bucket, time_id* and *row_id*) and 11 market features (*imbalance_size, imbalance_buy_sell_flag, reference_price, matched_size, far_price, near_price, bid_price, bid_size, ask_price, ask_size, wap*) from the auction book and the order book. The detailed description of the features can be found in Appendix. Below, the diagram shows us the time series structure of our non-iid dataset. For each stock, it might miss records in certain days.

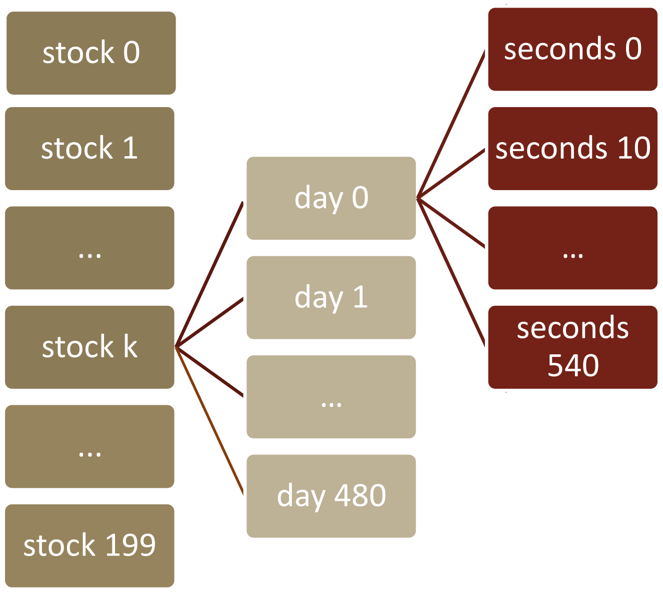


Figure 2. Time Series Structure of Dataset Records

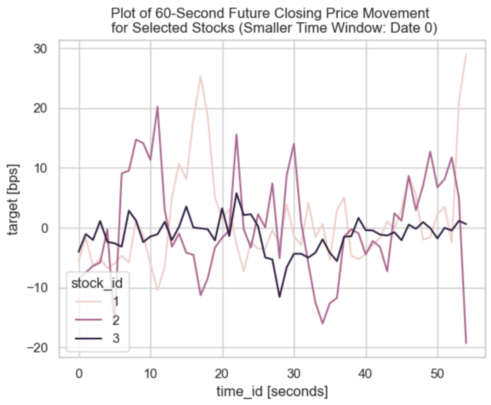
As this is an ongoing challenge on Kaggle, there has been a few submissions using the LightGBM model [3], which is a faster variant of Gradient Boosting Decision Tree, to predict the result. The result is evaluated by the Mean Absolute Error (MAE). Previous top submissions can achieve MAE scores of around 5.3 on separately provided test sets after feature engineering. The focus of our project, however, will be more education-oriented, instead of simply optimizing the scores and running time. As a result, we may cover less feature engineering techniques with domain knowledge, and will only select data with *date_id* less or equal to 120 given limited computational resources. The data size can still reach 1 million after filtering.

2 Exploratory Data Analysis (EDA)

In the EDA part, we take a look at each feature and their interrelationships. Key figures are selected to give us a deeper understanding of the dataset.

2.1 Target Variable

Figure 3 and Figure 4 are the time series plot of the target closing price movement variable for stocks 1-3 on certain time windows. We see that different stocks have different volatilities. Though there can be extreme values in the price movement index, in the long term, mean reversion towards zero is perceived.



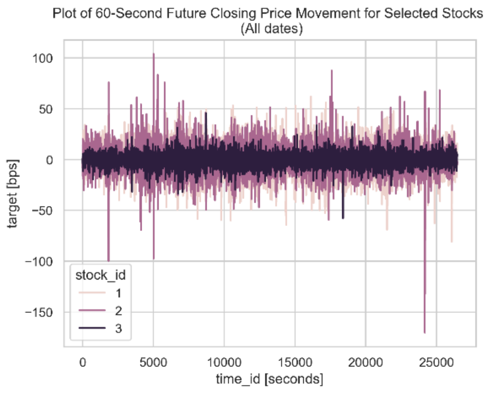


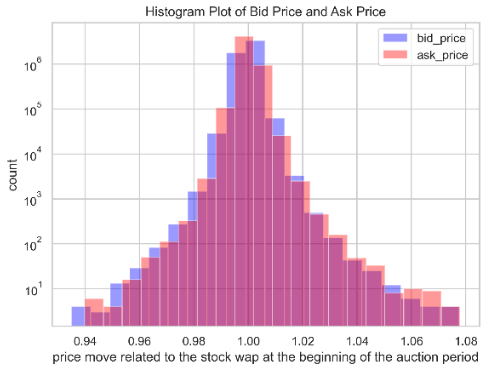
Figure 3 & 4. **Plots of Target Closing Price Movement for Stocks 1-3 on Day 0 / All Dates**

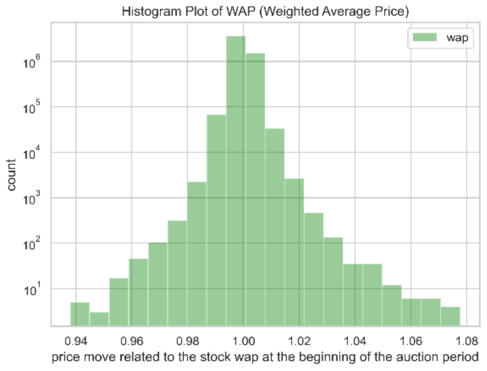
2.2 Correlation Matrix

In Figure 5, we can see that there exists a strong relationship between price features (*bid price, ask price, reference price* and weighted average price *wap*). These prices are all converted price relevant to the stock *wap* at the beginning of the auction period and are actually closely related by definition. Therefore, despite the strong correlation, we still keep all these features.

2.3 Bid, Ask and Weighted Average Price

To further confirm our perception, we plotted the distribution of *bid_price, ask_price* and *wap* as below. As can be seen from figures 5 and 6, the distribution of the converted bid price, ask price, and their weighted average price (wap) are very similar.





Figures 5 &6. **Histogram Plots of Bid Price, Ask Price and WAP**

In figure 7, we see that the converted bid prices have linear relationship with the ask price, and it's always smaller than the ask price. This relationship is a natural effect of the order mechanism.

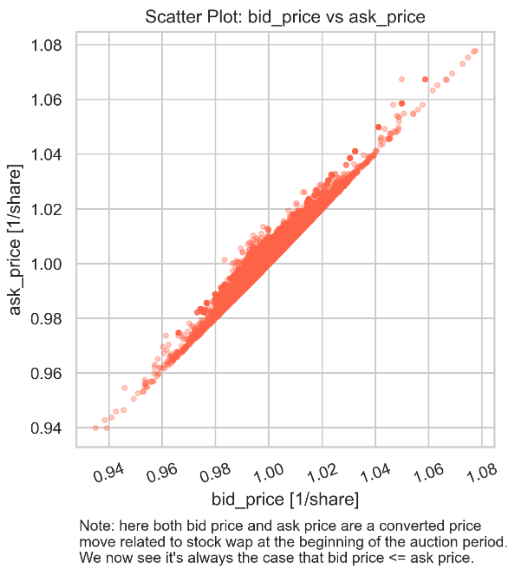


Figure 7. Scatter Plot of Bid Price vs Ask Price

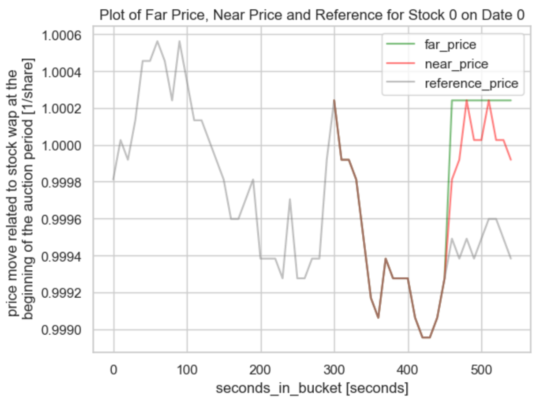


Figure 8. Plots of Far, Near and Reference Prices for Stock 0 on Day 0

2.4 Far Price, Near Price, and Reference

From figure 8, it is obvious that the far price and near price are only available in the last 5 minutes of each day. This will introduce missing values later to our feature matrix, and these missing values cannot be imputed.

2.5 Auto-correlation of Target Variable

Lastly, we look at the autocorrelation plot for the target variable. Though the autocorrelation effect is not significant, we will later construct lagged features to incorporate the time series structure.

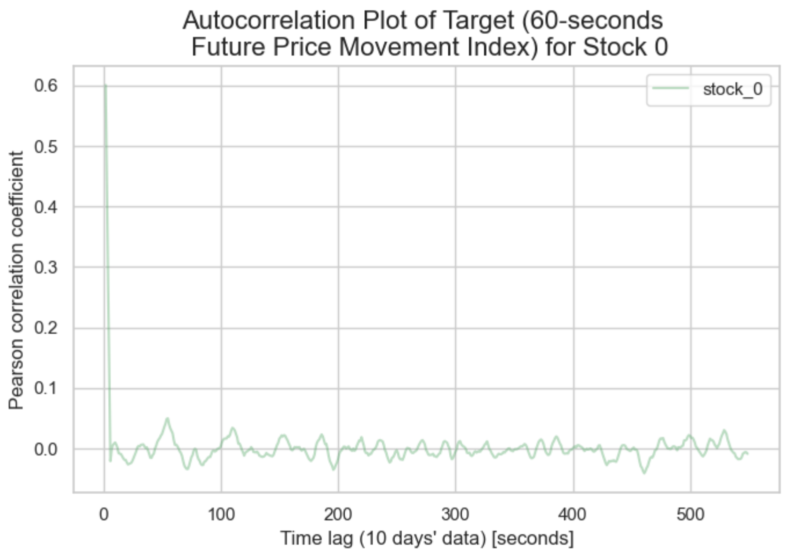


Figure 9. 1-Day Autocorrelation Plot for the Target Price Movement Index for Stock 0

3 Methods

Now we move to the model pipeline in this section.

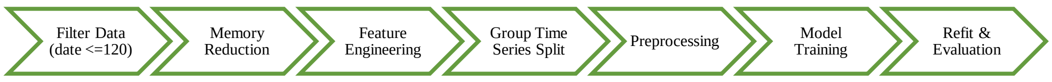


Figure 10. Machine Learning Pipeline Overview

3.1 Memory Reduction

To reduce memory usage, we convert the integer data type (*int64*) for each feature according to their range before further process our data. This can halve the memory usage for the dataset. During the processing and training, we will also delete and collect variables once they are no longer used.

3.2 Feature Engineering

Manual feature engineering is an important part of achieving better the scores while getting meaningful features. We referred to features in [4] during this step.

- *Lagged Targets:*
  55 features (1-day lagged target) are constructed.
- *Pairwise Price Imbalances:*
  This group of features is constructed by using all price-related columns. It can be calculated by *(price_a - price_b) / (price_a + price_b)*.
- *Pairwise Size Imbalances:*
  Similar to price imbalances, this group of features can be constructed by *(size_a - size_b) / (size_a + size_b)*.
- *Statistics:*
  We calculated the mean, standard deviation, skewness and kurtosis for price-related columns.
- *Features with Financial Meaning:*
  Additionally, a few other features are constructed using domain knowledge (e.g., price pressure, market urgency, etc..)

3.3 Data Splitting

To add randomness into our pipeline, we use the following rolling window strategy for 4 folds based on *GroupTimeSeriesSplit* [5] to split the data. Generally, we have set *Test group size = Val group size = 0.2 * total group size.*

Sample Date Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Fold 1	train	train	train	val	val	test	test			
Fold 2		train	train	train	val	val	test	test		
Fold 3			train	train	train	val	val	test	test	
Fold 4				train	train	train	val	val	test	test

Figure 11. Group Time Series Split Strategy

3.4 Preprocessing

We use *OneHotEncoder* for two categorical features (*stock_id, imbalance_size*). Since all the remaining features are numerical, we apply the *StandardScaler* for these continuous features.

3.5 Evaluation Metrics

Following previous work and competition setting, we are using mean absolute error (MAE) as the evaluation metrics for our regression problem.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

Here *n* is the sample size.

3.6 Modelling with Missing Values

As previously said, both *far_price* and *near_price* have nearly 55% missing values due to the market mechanism. We cannot impute the relevant columns. Therefore, in the modelling part, we are using *eXtreme Gradient Boosting (XGBoost)* method [6], which directly handles missing values, and reduced features method [7], which handles missing by pattern submodels, for modeling. In test sets, there are 3 missing patterns, and we tried a few linear and non-linear models. Table 1 shows the summary for all the methods and hyperparameters.

Generally, linear model takes much shorter running time on large datasets, even compared with models trained with GPU acceleration. Reduced features XGBoost method, though taking more memory, has a shorter running time compared with directly using the XGBoost model in our case.

Model Name	**Final Parameters Tunned**	**4-Fold Running Time**	**Device**
XGBoost	reg_alpha: [0, 0.01, 0.1, 1], reg_lambda: [0.1, 0.5, 1, 2], max_depth: [1, 3, 7, 11, 13]	15444.11 seconds (> 4 hours)	GPU P100
Reduced Feature Method: Lasso	alpha: np.logspace(-2, 1, 15)	3073.89 seconds (< 1 hour)	CPU
Reduced Feature Method: Ridge	alpha: np.logspace(-2, 1, 15)	896.27 seconds (about 15min)	CPU
Reduced Feature Method: Random Forest	max_features: [0.5, 0.75, 1.0, None], max_depth: [1, 5, 7, 11, 13, None]	>12 hours per fold, time limit exceeded, abandoned	CPU
Reduced Feature Method: Support Vector Machine	gamma: [0.001, 0.1, 10, 1000], C: [0.01, 1, 10]	time limit exceeded, abandoned	CPU
Reduced Feature Method: XGBoost	reg_alpha: [0, 0.01, 0.1, 1], reg_lambda: [0.1, 0.5, 1, 2], max_depth: [1, 3, 7, 11, 13]	5821.88 seconds (1.61 hours)	GPU P100

Table 1. Machine Learning Pipeline Overview

Note that due to limited computation power, we were not able to get a result for Support Vector Machine and Random Forest. These two methods will therefore be excluded.

4 Results

4.1 Score Summary

Model Name	**Test Scores**
Baseline: prediction = mean of target index in test set by stocks	mean MAE: 6.9267
Baseline: prediction = zeros(mean reversion)	mean MAE: 6.9330
Baseline: prediction = 0 iff imbalance flag = 0, 0.1 iff imbalance flag = 1, -0.1 iff imbalance flag = -1	mean MAE: 6.9331
XGBoost	[6.7388, 6.7423, 6.7413, 6.7407] mean MAE: 6.7407, std: 0.0013
Reduced Feature Method: Lasso	[6.7355, 6.7354, 6.7354, 6.7353] mean MAE: 6.7354, std: 0.00005
Reduced Feature Method: Ridge	[6.7569 6.7568 6.7599 6.7573] mean MAE: 6.7577, std: 0.00126
Reduced Feature Method: XGBoost	[6.7403, 6.7393, 6.7388, 6.7371] mean MAE: 6.7383, std: 0.0011

Table 2. Summary Table for Baseline and Model Scores

The model scores are shown in the above table 2. To compare with the baseline, we calculate the baseline score in 3 ways and select the best mean MAE as our final baseline score. The first method is to choose the test mean by each stock as the baseline score. The second method, inspired by the mean reversion phenomenon in EDA, simply choose zero as the prediction. The third method map the prediction based on the *buy_sell_imbalance_flag*.

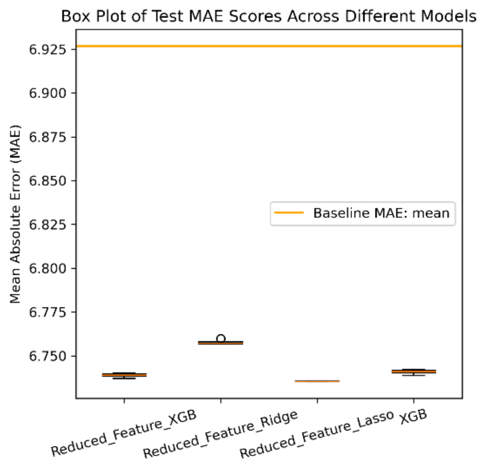
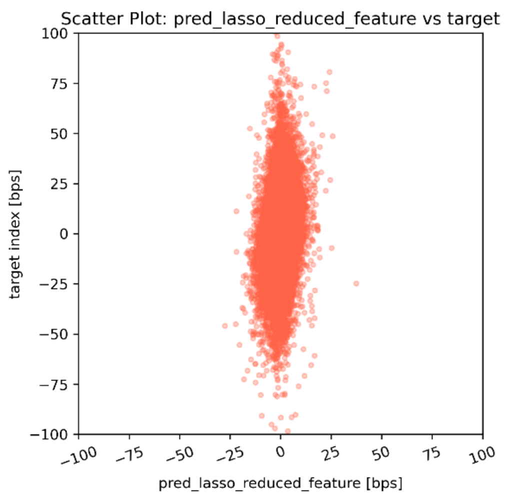


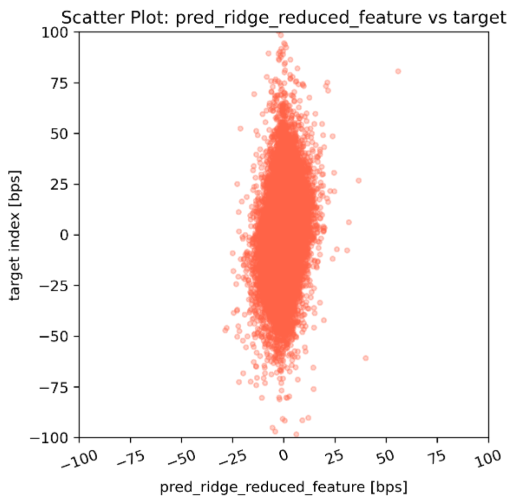
Figure 12. Box Plot of Test MAE Scores for Different Models

According to figure 12, our models only perform a little bit better than the baseline mean, where Lasso get the best MAE score in terms of both mean. However, the performance difference between models is quite minor. We will see this from the scatter plot in next subsection.

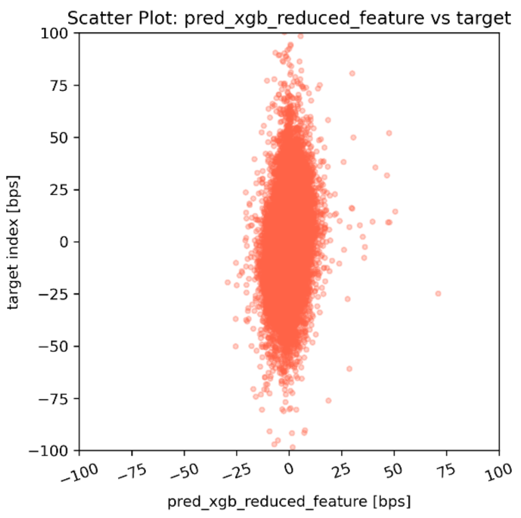
4.2 Scatter Plot for Fold 4 Test Result

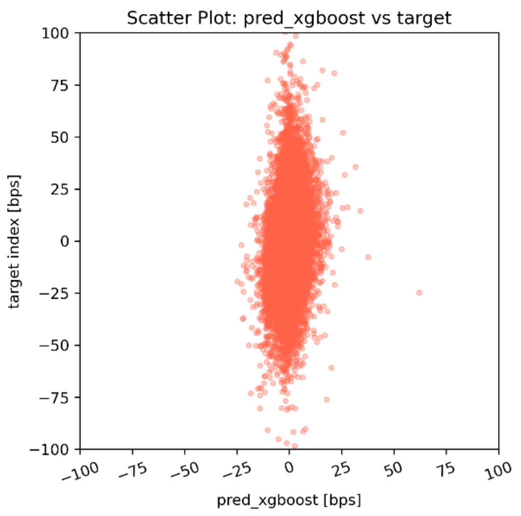
From figures 13-16 below, we see that the models' predictive power is not as desired. We know that if the model fit the data well, it should look like a diagonal line. Specifically, extreme values in the target are not captured.





Figures 13 & 14. **Scatter Plot: Prediction vs True for LASSO and Ridge Reduced Features**





Figures 15 & 16. **Scatter Plot: Prediction vs True for XGB (Reduced Features) and XGB**

4.3 Feature Importance

Since all the predictions generates similar test scores, in this section, we look at the feature importance of XGBoost. We are not calculating the permutation importance considering the computation time on large dataset and the strong correlation between features.

4.3.1 Global Feature Importance: XGBoost Gain and Weight

For XGBoost metrics, weight focuses on frequency of use in the model, not directly on impact on prediction accuracy. Gain emphasizes the quality of splits involving the feature, indicating its contribution to model accuracy.

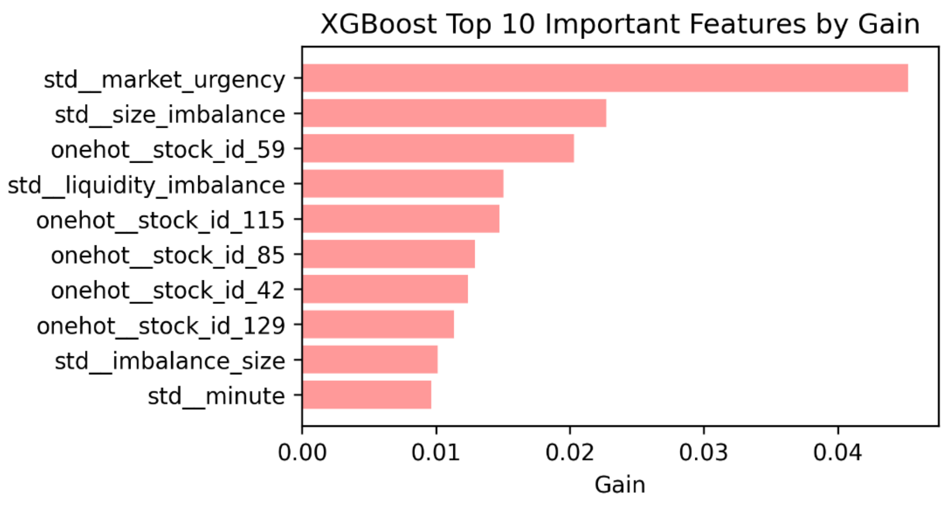


Figure 17. **XGBoost Top 10 Features by Gain**

For Gain, the top 3 most important features are *market_urgency, size_imbalance* and *stock_id_59*.

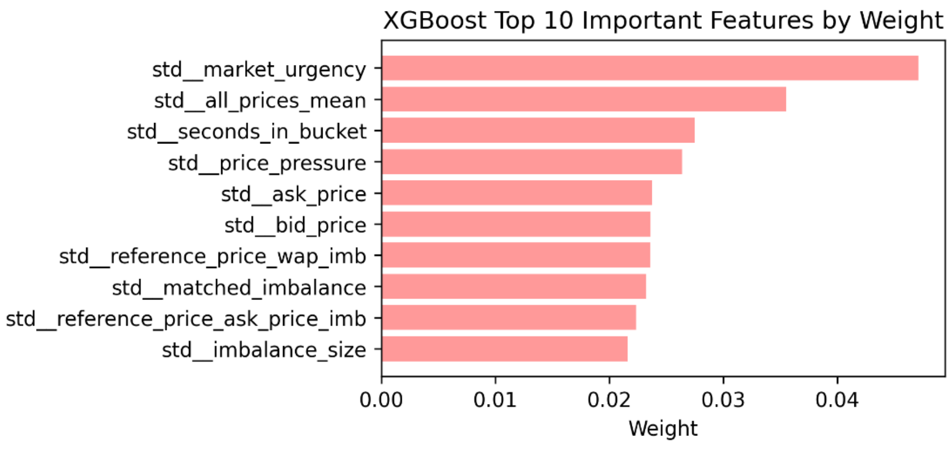


Figure 18. **XGBoost Top 10 Features by Weight**

For Weight, the top 3 most important features are *market_urgency, all_prices_mean* and *seconds_in_bucket.*

4.3.2 Global Feature Importance: SHAP Value [8]

This type of model-agnostic and consistent metrics, incorporating interactions between features, can provide us with a more nuanced, individualized measure of feature importance. From figure 19, the top 3 most important features are *market_urgency, reference_price_wap_imb* and *liquidity_imbalance*.

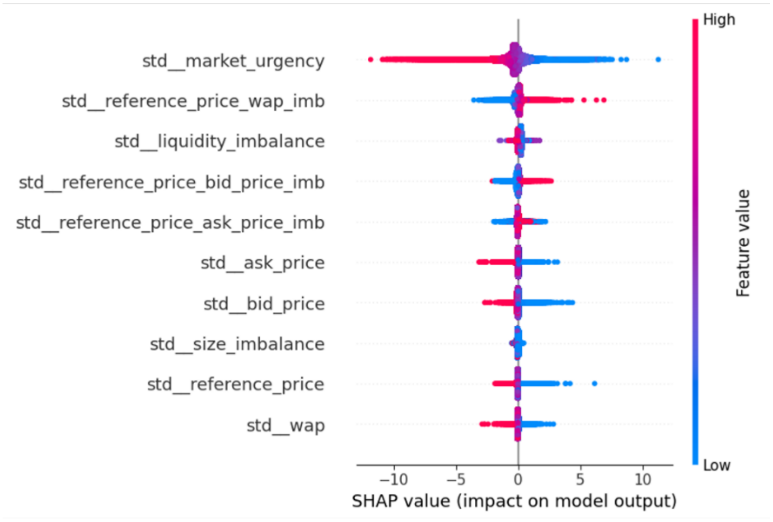
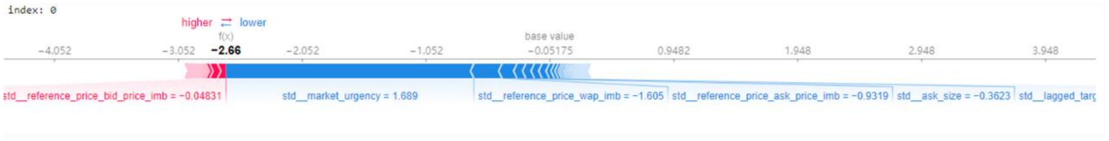


Figure 19. **Global SHAP Value for Top 10 Features of XGBoost Model**

4.3.3 Local Feature Importance: SHAP Value [8]

In SHAP force plots, features pushing the prediction higher are shown in red, and those pushing the prediction lower are in blue. The expected value among test data is -0.05175. We see that market_urgency (with negative contribution) and other constructed features are playing the most important part.



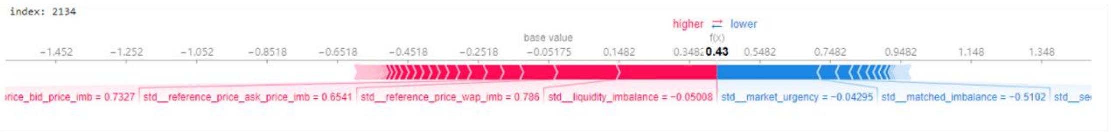


Figure 20 & 21. **Local SHAP Value of XGBoost Model for Records with Index 0 and 2134**

4.3.4 Summary

To conclude, many constructed features (e.g., *market_urgency*) are among all types of the most important features, which showcases the importance of feature engineering for this prediction problem. The least important features given by both SHAP and XGBoost are all one-hot encoded *stock_id*.

5 Outlook

5.1 Predictive Power Enhancement

Currently, the result is not desirable enough. To improve the predictive power in the future, we may construct more meaningful features to capture market dynamics. Moreover, we can include the whole dataset and do fine-tuning for larger parameter grid with more computational resources. It might also help if we try to apply new models such as LSTM, LightGBM to solve the problem.

5.2 Model Interpretability

In terms of interpretability, we are currently unable to find a general way to interpret the result for reduced features method, as the importance scores are only for submodels. More literature review and discovery are needed for this.

(Word Count: 1980)

6 References

[1] *Nasdaq Closing Cross Frequently Asked Questions.* (n.d.). NasdaqTrader. Retrieved December 8, 2023, from https://www.nasdaqtrader.com/content/productsservices/Trading/ClosingCrossfaq.pdf

[2] *Optiver - Trading at the Close.* (n.d.). Kaggle. Retrieved October 1, 2023 from https://www.kaggle.com/competitions/optiver-trading-at-the-close/

[3] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

[4] *Best Public Score.* (n.d.). Kaggle. Retrieved December 8, 2023 from https://www.kaggle.com/code/peizhengwang/best-public-score

[5] *GroupTimeSeriesSplit: A scikit-learn compatible version of the time series validation with groups.* (n.d.). GitHub. Retrieved December 8, 2023 from https://rasbt.github.io/mlxtend/user_guide/evaluate/GroupTimeSeriesSplit/

[6] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

[7] Fletcher Mercaldo, S., & Blume, J. D. (2020). Missing data and prediction: the pattern submodel. *Biostatistics,* 21(2), 236-252.

[8] Lundberg, S. M., Erion, G. G., & Lee, S. I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888.*

7 Appendix: Feature Description Table

Features	Description
stock_id	A unique identifier for the stock. Not all stock IDs exist in every time bucket.
date_id	A unique identifier for the date. Date IDs are sequential & consistent across all stocks.
imbalance_size	The amount unmatched at the current reference price.
imbalance_buy_sell_flag	buy-side imbalance: 1; sell-side imbalance: -1 no imbalance: 0
reference_price	The price at which paired shares are maximized, the imbalance is minimized and the distance from the bid-ask midpoint is minimized, in that order. Can also be thought of as being equal to the near price bounded between the best bid and ask price.
matched_size	The amount that can be matched at the current reference price (in USD).
Far_price	The crossing price that will maximize the number of shares matched based on auction interest only. This calculation excludes continuous market orders.
Near_price	The crossing price that will maximize the number of shares matched based auction and continuous market orders.
Bid and ask price	Price of the most competitive buy/sell level in the non-auction book.
Bid and ask size	The dollar notional amount on the most competitive buy/sell level in the non-auction book.
wap	The weighted average price in the non-auction book.
seconds_in_bucket	The number of seconds elapsed since the beginning of the day's closing auction, always starting from 0.