EXHIBIT 3

Selective Midpoint Indicator (SMI): A Gradient Boosted Signal Enabling Stable Order Executions

Hanyu Zhang, Shreejith Billenahalli Lokegowda, Stefanos Bazinas, Sophia Lichoulas

New York Stock Exchange

November 25, 2024

1. Introduction

Equity securities, particularly highly liquid stocks, exhibit frequent price fluctuations even within just a few microseconds. While numerous studies analyze market dynamics at levels of multiple milliseconds, few consider microsecond granularity. Furthermore, prevailing research prioritizes alpha-seeking strategies over market stability and execution quality. To extend the current literature, this paper presents a novel decision tree-based model capable of forecasting market stability for a given symbol at the microsecond level.

Our Selective Midpoint Indicator ("SMI") provides an ultra-fast, lightweight, and transparent model which can be seamlessly integrated into the NYSE Pillar trading system and greatly improves the performance of an order type leveraging input from the SMI to prevent executions when the SMI predicts market instability. The SMI achieves recall rates as high as 90% for liquid symbols while minimizing the duration of the instability windows during a trading day, thus ensuring fill rates are minimally impacted.

2. Current State and Motivation

NYSE Arca currently offers a discretionary order type that market participants can leverage to provide liquidity and interact with incoming orders in a more flexible way than traditional limit orders. A Discretionary Pegged Order (as defined in NYSE Arca Rule 7.31-E(h)(3)) is a non-displayed order type that pegs to the same side of the Protected Best Bid and Offer ("PBBO") - the PBB for buy orders and PBO for sell orders - with discretion to trade up to the midpoint of the PBBO if there is available liquidity on the opposite side. Despite the flexibility that this order

type currently offers, it does not currently provide any price protection during highly volatile market conditions. Such periods of volatility often occur rapidly and last for less than 1ms, thereby offering an inherent advantage to market participants with the lowest latency profiles.

Such order types have become more prevalent in recent years and are already available on some U.S. equity exchanges. Existing offerings all share a major limitation - a timer-based approach. [1] Upon order acknowledgment, orders are protected (either by limiting the order's discretionary range or its working price) for a time interval determined at the time of order acknowledgment. Current offerings differ from each other in (1) the way the time interval is determined (static vs. dynamic) and (2) the way orders are handled during unstable periods. However, all of the existing offerings currently use a pre-determined time interval, meaning that none of the existing indicators react to information and data generated between order acknowledgement and the expiration of the timer. This pitfall can lead to missed executions simply because the timer is not making real-time predictions, and orders are protected for a non-dynamic and pre-determined time interval.

Our new SMI seeks to provide an improved experience for market participants by altogether eliminating the need for a timer. The SMI employs a gradient boosting machine learning model in evaluating real-time data from NYSE Arca's order book to predict future microsecond-level market stability for a given symbol. Selective Midpoint orders ("SeMi Orders"), as proposed in SR-NYSEARCA-2024-112, will leverage the SMI and will be locked during times when the SMI predicts quote instability for a given symbol, significantly improving execution quality for these orders and improving markouts at the cost of minimally reduced fill rates.

3. Methodology

3.1 Understanding Decision Trees

We modeled our stability prediction problem as a decision tree problem. Decision trees are supervised learning algorithms used for classification and regression tasks. They fit our problem well since we are using structured data. The algorithm splits the data into subsets based on the value of input features. The splitting forms a branching decision process and continues until a stopping criterion is met (e.g., a maximum depth or minimum number of samples per leaf).

---

[1] CBOE's Quote Depletion Protection (QDP), Nasdaq's Midpoint Extended Life Order (M-ELO) and Dynamic M-ELO and IEX's D-Peg all employ machine learning approaches to provide some level of price protection for incoming orders.

Essentially, a decision tree creates a sequential series of questions about the input data that returns an output label.

Gradient boosting is an ensemble learning technique. It involves two kinds of models: (1) weak learners, which are usually decision trees, and (2) a strong learner formed by combining multiple weak learners. Weak learners are trained sequentially, and each successive model works to improve the error from the previous model by focusing on accurately predicting cases where the first model does poorly. The final prediction in a classification problem uses majority voting of all weak models.

3.2 Decision Tree vs. Neural Networks

Decision trees are an industry standard when working with tabular data. While they are an obvious model to consider, we will explain why they are ultimately better suited than Deep Learning for our use case.

The main advantage of decision trees is explainability. The output of a decision tree is a hierarchy of specific questions such that users can easily follow the decision-making process. One can assess the importance of a feature or even examine what caused a specific model output. In other use cases, explainability would be a nice-to-have, but in our case it is critical; ensuring the best possible execution for clients is of crucial importance. Markets change in unexpected ways over time. We thus require our model to be flexible and interpretable in cases of shifting market dynamics, regime changes, or regulatory updates. An interpretable model allows us to evaluate the relationships between such market shifts and feature selection and weightings, as well as overall model performance.

The primary requirements of our model are best-in-class performance and memory efficiency. With over 80 features under consideration, we must select a model that scales well with number of features. It would be impossible to integrate a neural network model into the NYSE Pillar trading system while at the same time maintaining a sufficiently low inference time.

4. Market Stability Model

Before developing the model, we must first define “instability.” At a high level, we want our definition to capture relatively large price moves during a relatively short time frame. We consider symbols independently and recognize that market conditions change over time. We parameterize our ground truth labeling process to closely represent market instability for various symbols and changing conditions.

For clarity, we introduce an intermediate label termed "price jump" and then use that term to define "unstable," and "side-aware unstable." In the included examples, we assume there is one PBBO update per microsecond for simplicity.

4.1 Price Jump

A “price jump” is defined as a case where the PBBO mid-price moves by a pre-defined percentage of a symbol’s spread (the spread threshold ***X***) in either direction within a configurable time interval (the time horizon ***G***). If the move is positive (negative) - i.e. the mid-price increases (decreases) - then the price jump is classified as positive (negative) and is assigned a value of 1 (-1). Otherwise, the price jump value is 0.

Otherwise stated, a price jump is labeled 1 or -1 at a given point in time if, looking back to the start of the time horizon, the mid-price was at least the spread threshold in difference from the current mid-price. This approach of identifying discrete price jumps over configurable time intervals offers us the flexibility to adjust either of the parameters (spread threshold and/or time horizon) according to symbol-specific dynamics. Note that a price jump can be labeled in real time since the definition looks backwards in time.

Figure 1

Price jumps over time

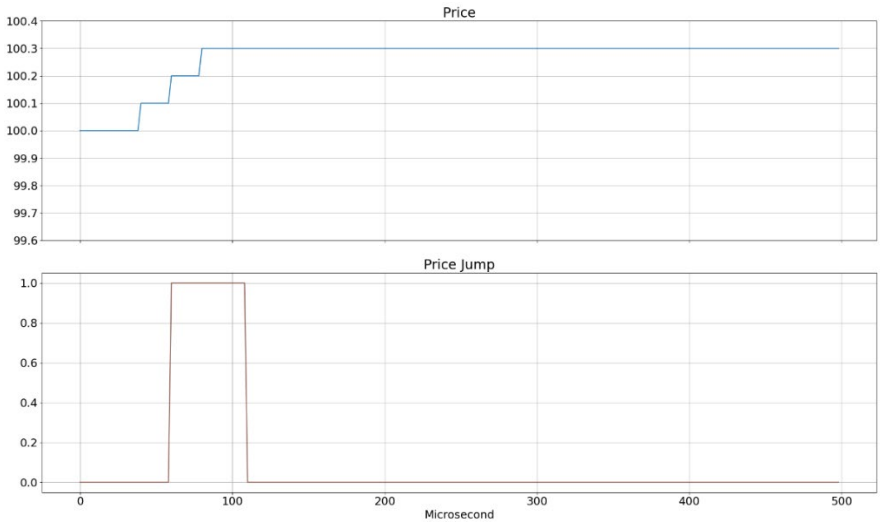


Figure 1 shows an example of how price jumps are calculated, for a sample symbol with a spread threshold of 20%, current spread of $1.00, and a time horizon

of 50us. The PBBO mid-price is $100 from 0us to 40us, and it increases by $0.10 every 20us until 80us for a total of three $0.01 price moves. At 60us, the PBBO mid-price has now increased by at least 20% of the $1.00 spread ($0.20) over the preceding 50us. Thus, all PBBO updates from 60us (the first threshold breach) to 130us (the last threshold breach plus the time horizon) are labeled with a positive price jump of 1.

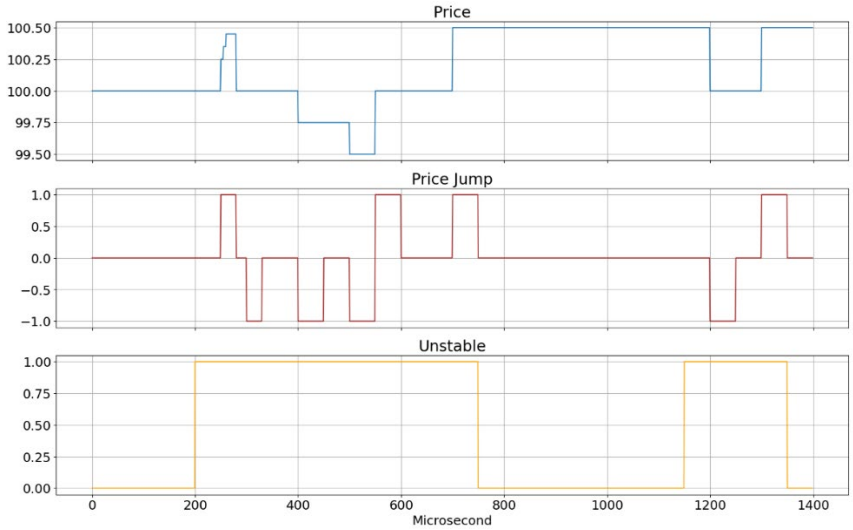
4.2 Unstable Quote

Price jumps label discrete PBBO updates. Since the SMI is a continuous signal, we need to apply our price jump definition to further define continuous periods of market instability. We use the following process to define market stability.

For each PBBO update (index *i*), we check whether the conditions for a price jump are satisfied (independent of direction). If so, we then look for subsequent price jumps within our time horizon ***G***. If one can be found, we iterate this process until no price jump can be identified and classify an unstable time window starting at the *i-1* PBBO update or 50us prior to the *i* PBBO update (whichever is closest to *i*) and ending at the last price jump.

Figure 2

Unstable window (time horizon of 400us)

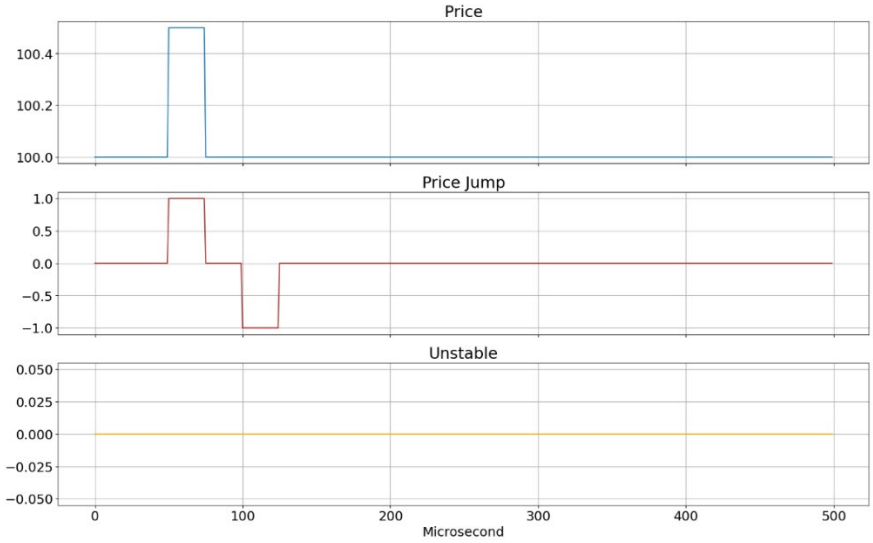


To ensure that the unstable windows we are capturing are persistent, we employ a third parameter ***g***. If multiple price jumps are too close to each other and

fall into one small time window such that the distance between the first and last price jump is less than g (e.g., g = 100us), then we do not mark this window as unstable. This is a pivotal step in the process, as it allows us to isolate and correctly identify persistent price moves and filter out temporary ones that quickly reverse back to the original price. Figure 3 shows how the configuration of parameter g filters out temporary price fluctuations.

Figure 3

Stable window with price jumps



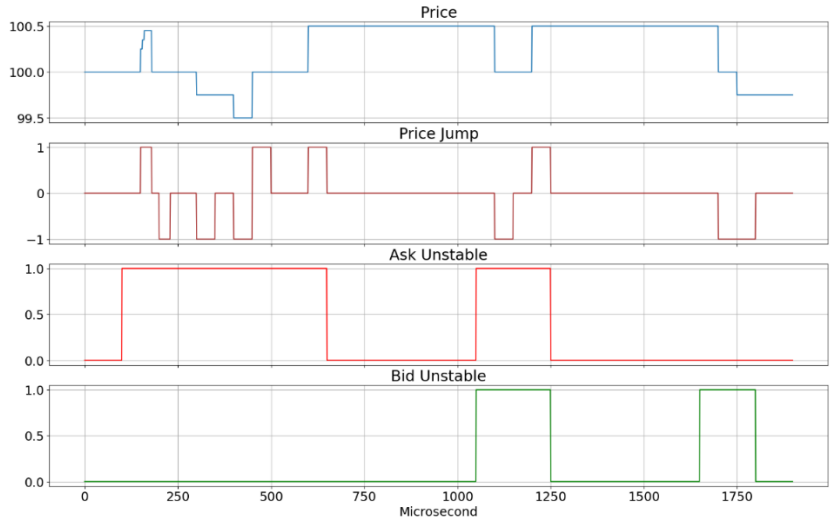
4.3 Side-Aware Unstable Quote

In the development of the SMI, it is important to differentiate between quote instability on each side of the quote, i.e., separately for the bid and ask side. This is of crucial importance when considering its implementation in future order type behavior, since the aim of SeMi Orders is to provide protection against adverse movements in a stock's price.

In this context, we would want to avoid preventing executions of buy (sell) orders based on upwards (downwards) instability of the mid-price of the PBBO, since these would be considered favorable conditions for order execution. In other words, if the mid-price of the PBBO is higher (lower) at the end of an unstable window, we only mark PBBO updates in this window as unstable for the ask (bid) side of the model (Figure 4). Side-aware instability is therefore less restrictive than the instability indicator described in section 4.2 and leads to separate models for each side of the PBBO, a bid model and an ask model.

Figure 4

Side-aware unstable window



4.4 Final Model Parameters

As outlined in sections 4.1-4.3, the three parameters that need to be accurately tuned and used in our model evaluation are the:

1. Spread threshold ***X***, used in identifying and labelling price jumps.
2. Time horizon ***G***, used in identifying and labelling unstable time windows.
3. Minimum time increment ***g***, used to filter for persistent price changes during unstable time windows.

5. Model Development

5.1 Feature Selection

We carefully choose features to avoid any unnecessary model input. The features come from both the NYSE Arca order book and market data, and data points include but are not limited to book depth information, PBBO updates, number of immediate-or-cancel orders/shares and price aggressiveness of such orders. Appendix A contains a complete list of the 83 features under consideration as inputs in the SMI model. This list was developed through an iterative process of evaluating individual feature significance.

5.2 Data Gathering & Preprocessing

As part of the training process, we use data from the NYSE Arca order book for the period 8/29/24-10/22/24 and consider a set of 500 symbols chosen to reflect a representative sample of the US equity markets (full symbol list in Appendix B). Symbols are chosen based on multiple criteria, including but not limited to absolute price level, spread (in dollars), spread (in basis points), and liquidity (daily ADV) to ensure a representative sample across all criteria categories is evaluated.

PBBO updates are labeled according to Section 4.3 as 1s or 0s for each side of the PBBO. Due to the nature of U.S. equity markets, stable datapoints usually largely outnumber unstable datapoints. As previously argued, unstable market conditions tend to last for very short time intervals but tend to represent the majority of unfavorable executions on a given day. To account for this data imbalance and to avoid a biased model output, we employ under-sampling methods when appropriate to reduce the number of observations in the majority (stable) class and randomly shuffle the data before training.

5.3 Training Process

Each model is retrained daily using data from the preceding three days. Evaluation data points are created for each PBBO update, rather than each book depth update. Book depth information has much more granularity than PBBO updates: two adjacent data points in the first several levels of book depth could be the same if the change happens on a deeper level. To reduce the size of the dataset and increase the informational value-add of each data point, we create new data points upon each PBBO update and merge the closest book depth state as of the time of each PBBO event. We calculate all features and labels based on this merged dataset.

6. Results

6.1 Performance Metrics

The objective of our framework is to correctly predict and capture as many unstable PBBO updates as we can, while at the same time limiting the number of times we incorrectly predict a stable PBBO update as unstable. Our emphasis lies more in achieving greater recall rather than solely pursuing an elevated overall F1

score. This strategic shift reflects our willingness to accept instances where the model identifies more data points as unstable than indicated by the ground truth, provided that the number of false unstable instances (and aggregate time associated with those instances) remains proportionally insignificant compared to the total number of PBBO updates (or total seconds) within a given day. As a result, our focus is on:

(1) Maximizing model recall, i.e., making sure we are capturing the maximum number of truly unstable data points.

$$Recall = \frac{True\ Unstable}{True\ Unstable + False\ Stable}$$

(2) Maximizing model precision, i.e., making sure we mislabel as few truly stable data points as possible in our predictions.

$$Precision = \frac{True\ Unstable}{True\ Unstable + False\ Unstable}$$

(3) Minimizing overlocking, i.e., minimizing the amount of time (in seconds) that the model predicts unstable market conditions over the ground truth.

$$Overlocking = \frac{Model\ predicted\ unstable\ time\ (sec)}{Actual\ unstable\ time\ (sec)}$$

We evaluate model performance holistically across metrics. Price-driven performance supersedes fill rates when evaluating price protection mechanisms. To improve price-based markouts, recall is our North Star metric whereas precision and overlocking serve as guardrail metrics. So long as recall is strong and overlocking is low, we are willing to accept a precision baseline of at least 20% during the model evaluation process, which is reasonable given the imbalanced nature of our dataset.

6.2 Parameter Calibration

In order to evaluate the performance of our model, it is important to first calibrate our three parameters (see Section 4.4). In the below sections, we evaluate the stability and performance of a baseline model for different levels of each parameter to arrive at the final parameter values. Given the dynamic nature of U.S. equity markets and the diversity of symbols it is composed of (e.g., price, spread,

liquidity), these parameter values are likely to change over time to ensure proper calibration.

6.2.1 Minimum Time Increment (*g*)

Table 1 below shows the distribution of time between successive PBBO updates for 15 of the most liquid U.S. equity symbols as well as the entirety of the U.S. equity markets for the period July-October 2024. For these 15 symbols, 10% of all PBBO updates happen within at most 64us from one another. This result holds for the aggregate of U.S.-listed equity securities, with 10% of all PBBO updates occurring within 46-55us during the four months. This further illustrates our earlier point about the ever-increasing speed of price updates in today's markets and highlights the need to ensure proper identification of market instability.

Some PBBO updates reflect a real change in the price of a security, while some simply capture a reversion in price when a security was temporarily dislocated. We do not attempt to decide what the "true" price of a security is, but it stands to reason that quick corrections of dislocations would often have relatively short times between PBBO updates.

Table 1

Distribution of time (microseconds) between successive PBBO updates

	Jul-24				Aug-24				Sep-24				Oct-24			
	P10	P25	P40	P50	P10	P25	P40	P50	P10	P25	P40	P50	P10	P25	P40	P50
AAPL	25	79	292	695	29	101	386	910	33	117	433	916	37	144	509	1,095
AMD	27	99	411	960	36	139	508	993	49	211	623	1,241	46	180	540	1,064
BAC	13	43	108	271	13	43	106	235	13	46	117	265	16	55	146	362
F	9	35	75	131	5	32	66	108	9	40	85	149	7	39	86	151
NIO	9	32	69	114	8	30	60	99	9	31	64	107	6	30	68	117
NVDA	27	92	341	801	30	105	387	840	32	118	440	1,000	30	95	388	937
PFE	13	43	93	183	13	42	91	175	10	38	81	148	11	39	86	160
QQQ	64	324	1,332	3,597	55	234	739	1,857	53	204	635	1,473	59	258	890	2,494
SNAP	16	46	114	293	13	40	96	205	11	38	87	167	12	39	86	167
SOXL	35	110	770	3,955	28	73	396	2,216	25	63	293	1,386	24	60	222	984
SOXS	18	60	169	447	28	91	482	2,872	22	66	200	617	17	53	140	349
SPY	57	241	833	2,488	57	242	725	1,802	58	237	714	1,747	57	243	786	2,191
T	14	42	93	186	11	39	84	155	7	37	82	144	9	38	83	149
TQQQ	22	57	257	2,600	26	72	357	2,676	23	62	221	1,193	20	50	111	394
TSLA	23	80	272	589	25	102	348	774	41	141	531	1,125	36	122	453	1,047
All	46	282	1,927	15,613	53	321	1,964	12,709	55	319	1,796	11,880	53	277	1,330	10,811

To identify the typical length of such reversion events, we analyze all PBBO updates where the mid-price of the PBBO is different within 25us but reverts to the original price within an additional 175us and extract the time interval elapsed until the price reversion. Table 2 shows the distribution of these intervals for the same set of U.S.-listed equity securities and period as Table 1. 75% of all such price reversions occur within at most 100us from the original PBBO update. Stated

otherwise, it only takes a total of 100us for a price to change and revert to the original price for 75% of all such cases. This value of 100us provides a very accurate and reasonable starting estimate for our minimum time increment ***g***.

Table 2

Distribution of time (microseconds) until price reversion for PBBO changes occurring within 50us

	Jul-24			Aug-24			Sep-24			Oct-24		
	P25	P50	P75	P25	P50	P75	P25	P50	P75	P25	P50	P75
AAPL	45	70	101	43	69	100	47	72	101	50	76	109
AMD	55	83	108	52	81	107	56	83	115	54	83	115
BAC	45	73	112	47	75	116	46	74	117	48	77	119
F	50	79	120	51	82	129	51	80	122	54	83	126
NIO	50	79	123	54	83	126	51	78	118	50	79	120
NVDA	48	76	107	48	74	107	47	72	106	46	71	106
PFE	48	76	117	50	77	119	50	78	121	51	78	121
QQQ	51	80	119	52	80	120	53	80	117	51	79	119
SNAP	44	70	107	47	76	117	47	74	115	48	77	117
SOXL	48	73	107	46	72	109	45	70	108	46	75	112
SOXS	45	74	113	46	73	111	47	76	116	45	75	117
SPY	49	74	111	52	79	118	53	81	119	50	77	117
T	48	77	121	50	79	125	50	81	126	52	82	129
TQQQ	50	81	100	45	74	102	47	76	100	53	82	109
TSLA	57	85	114	55	89	138	52	81	111	46	74	108
All	49	78	108	49	78	112	50	78	113	50	78	115

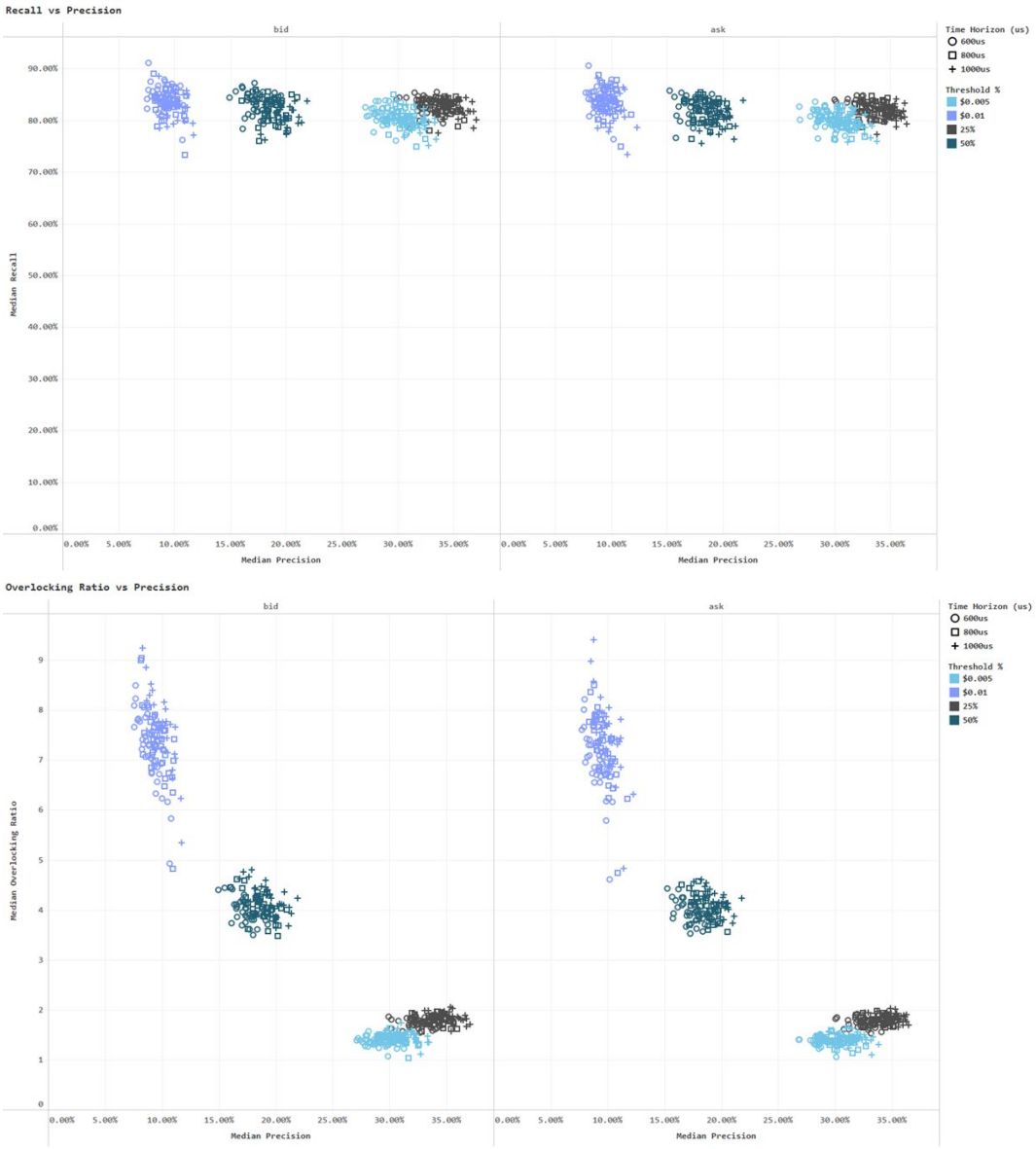
6.2.2 Spread Threshold (*X*)

The spread threshold determines the minimum change in the mid-price of the PBBO that we aim to predict. Figure 5 below shows the relationship between (1) recall and precision and (2) overlocking and precision for various levels of spread thresholds and time horizons, with each point in the plot representing the median values for a given day across the 500 symbols under consideration.

As shown in the scatterplot, a threshold value of 25% greatly outperforms parameter values of $0.01 and 50% in terms of all the metrics under evaluation: maximizing recall, maximizing precision, and minimizing overlocking. In particular, spread threshold values of $0.01 or 50% only marginally increase recall at the cost of multifold decrease in precision and multifold increase in overlocking. Additionally, a spread threshold value of 25% also outperforms the $0.005 parameter value by providing a recall and precision boost at only a slightly higher overlocking ratio. Since over 70% of U.S.-listed equity securities have daily average spreads of greater than $0.02 - meaning that 25% of the spread for these symbols would equal at least $0.05 - we choose the spread threshold level of 25% which maximizes the performance metrics analyzed.

Figure 5

Performance metrics for different levels of model spread threshold



6.2.3 Time Horizon (*G*)

Our selection of time horizon will greatly affect our model performance as shown in Table 3. For both aforementioned levels of spread threshold ($0.005 and 25%) for the 500 symbols in our universe, increasing our time horizon from 600us to 1ms greatly increases our model's precision, with very minimal impact on the model's recall and overlocking performance.

Table 3

Performance metrics for different levels of model time horizon

		Bid			Ask		
		Median Recall	Median Precision	Median Overlocking Ratio	Median Recall	Median Precision	Median Overlocking Ratio
$0.01	**600us**	80.91%	28.81%	1.355	80.33%	28.95%	1.349
$0.01	**800us**	80.40%	30.45%	1.407	80.07%	30.50%	1.385
$0.01	**1000us**	80.09%	31.54%	1.455	79.69%	31.70%	1.43
25%	**600us**	83.08%	32.32%	1.735	82.59%	32.41%	1.724
25%	**800us**	82.68%	34.04%	1.799	82.23%	34.02%	1.786
25%	**1000us**	82.24%	35.03%	1.865	81.78%	35.06%	1.849

6.3 Symbol-Specific vs. Market-Based Models

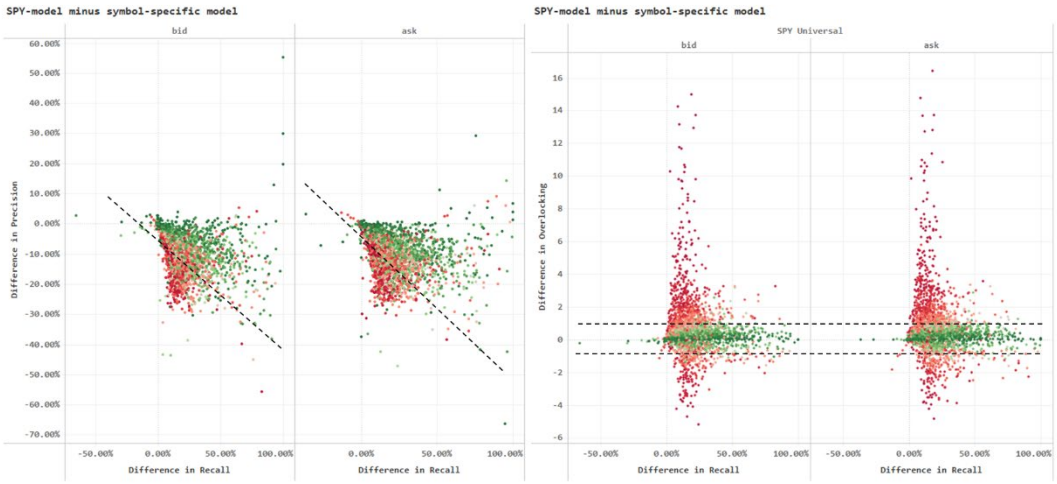
With over ten thousand securities traded listed and traded in the U.S. equity markets, and NYSE Arca trading over 90% of them daily, deciding how to scale our model across this vast universe of symbols is crucial. Even though maintaining a one-model-per-symbol solution could potentially be efficient enough to be incorporated into a trading system such as that of NYSE Arca, such an approach misses the more important question. What is the value-add of creating and maintaining a one-model-per-symbol framework?

To answer this question, we rank our universe of 500 symbols according to their total number of unstable data points that we were able to label for our training purposes. We then compare the performance of our universe of symbols on each symbol's symbol-specific trained model against its performance on the SPY trained model. In this context, we are using the SPY model as a proxy for a "market model" that could potentially be used for symbols with insufficient number of training data points. Figure 6 plots the difference of each symbol's SPY-model performance and symbol-specific performance. More active/volatile symbols (with more unstable datapoints) are color-coded in red, while less active/volatile symbols (with less unstable datapoints) are color-coded in green.

Symbols with fewer unstable data points perform much better on the SPY-trained model; recall is significantly higher, at the cost of a relatively small loss of precision and minimally more overlocking. In sharp contrast, more active/volatile symbols with more unstable data points to train each symbol's symbol-specific model perform noticeably worse on the SPY model; small gains in recall are paired with sharp declines in precision (as much as -30%) and large increase in overlocking behavior. As a result, our framework will need to employ a symbol-specific model for more active symbols, while symbols with less activity will need to default to a 'market' model, i.e. the SPY symbol-specific model.

Figure 6

Performance metrics for symbol-specific models vs SPY-model



6.4 Final Model Performance

To evaluate the performance of our final model, we combine the results and conclusions from Sections 6.1-6.3. We choose 100us as our minimum time increment (***g***), 1ms as our time horizon (***G***), and 25% as our spread threshold (***X***). We split our symbol universe into two separate groups – group 1 consisting of the symbols in the 20th percentile of average daily unstable datapoints during the evaluation period (a total of 100 symbols), and group 2 consisting of the remaining 80% of symbols (a total of 400 symbols). For symbols in group 1, we use each symbol's symbol-specific trained model. For symbols in group 2, we use the model trained on SPY data (the market model).

Table 4 shows the aggregated performance results of this combination of models. In aggregate, our model achieves an average recall rate of 90% and average precision of 30%, while overlocking for an additional 3.8 seconds on average. As discussed in Section 6.1, maximizing the model's recall rate and minimizing the overlocking window are of primary significance when evaluating the performance of our model. The model precision of 30% is well above the baseline of 20% and warranted to achieve high levels of recall without incurring substantial overlocking.

Table 4

Final Model Results

	Recall	Precision	Overlocking (seconds)	Overlocking Ratio
Bid	90.49%	30.18%	3.977	2.863
Ask	89.70%	29.95%	3.621	2.953
Aggregate	90.10%	30.06%	3.799	2.908

Symbol-specific model for 100 symbols with the highest number of unstable datapoints & Market-model for the rest 400 symbols
Time horizon: 1000us (1ms)
Spread threshold: 25%

7. Model In Production Environment

7.1 Prediction Time

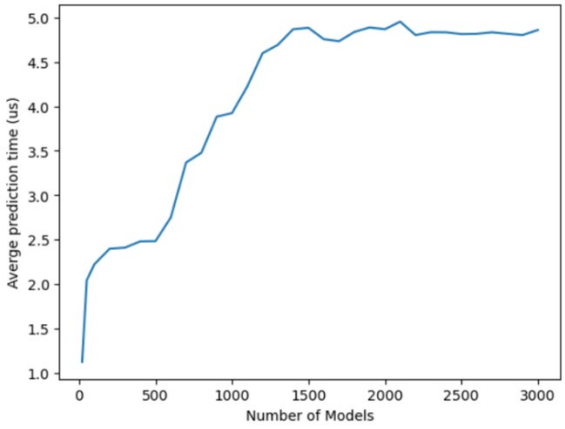
The prediction speed of our model is critical for its implementation within a production trading environment. The NYSE Pillar matching engines on average match orders within a few microseconds. If the model's process required more time than the matching time, locked symbols could fail to unlock quickly enough to execute against incoming orders, potentially causing missed opportunities and low fill rates. The model's current implementation is both efficient and fast. The model output is produced fast enough for real time trading, does not place an undue burden on the trading system, and does not degrade performance.

To achieve our fast prediction speeds, we maintain a compact model size, achieved through the incorporation of early stopping mechanisms during the training phase to limit the number of trees in the model. Additionally, we compile the model into a C static library. This strategic approach yields a substantial reduction in average prediction time – from 10ms to a mere 0.5us – when using one model to predict continuously with an Intel Xeon Platinum 8259CL CPU.

In practice, we have multiple models running sequentially in one process, one for each symbol (according to process described in section 6.4). Figure 7 illustrates the average prediction time when multiple such models are running within a single process. We selected one model at a time using a round-robin approach and did not utilize cache space when the combined size of all models exceeded the cache size, underscoring the importance of maintaining small model sizes. We additionally employ the existing symbol sequence of real production orders within NYSE Pillar to assess speed, resulting in an average prediction time of approximately 2us. Such efficiency allows seamless integration of the model prediction process into the NYSE Pillar trading system.

Figure 7

Average model prediction time by number of models utilized



7.2 Daily model maintenance

The daily model maintenance includes data collection, data preprocessing, training, model compiling, and shared library compiling. The first three steps are the same as described in Section 5; we collect historical exchange data, preprocess them on multiple instances on cloud or on-prem servers, and train the models. Furthermore, as discussed in Section 5, we proceed to compile all models into C static libraries. Subsequently, we consolidate these static libraries into a single shared library featuring just one function entry point. This function accepts a symbol as one of its parameters, facilitating the invocation of the respective model associated with the symbol.

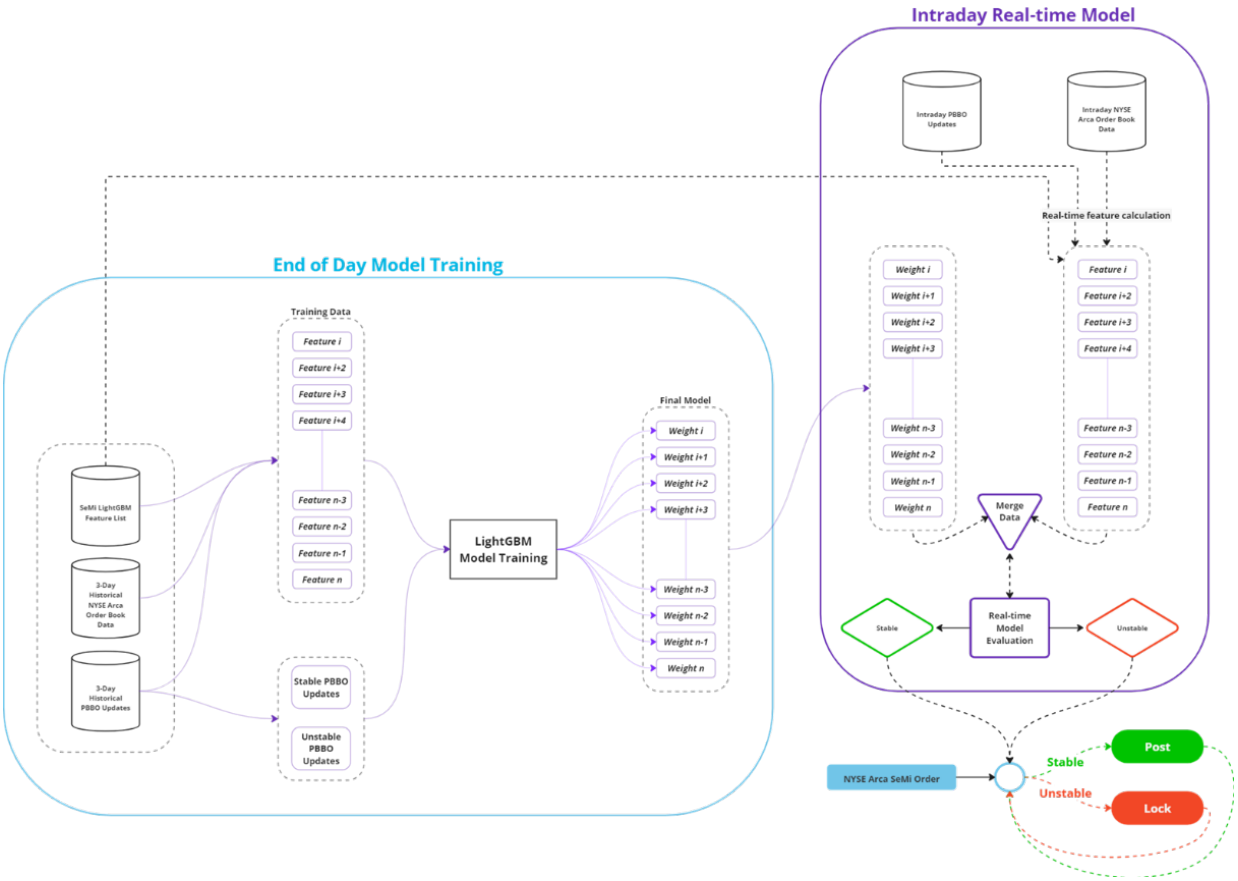
7.3 Intraday prediction: event driven pipeline

An internal trading system process which has access to the full real time trading data continuously maintains all the required features for this model, including but not limited to PBBO updates, order entries, order cancellations and book depth information. This process invokes the model for inference as soon as an evaluation trigger is received. Evaluation triggers primarily include PBBO updates as received by the NYSE Pillar matching engine. Since the model incorporates PBBO-agnostic features (e.g., individual order entries/cancellations), the PBBO trigger is complemented by a timer-based trigger. The inclusion of a timer-based trigger ensures that the model remains up-to-date when NYSE Arca's order book changes without an associated PBBO update, such as when non-displayed liquidity is added to or removed from the order book. This end-to-end process concludes by sending a

message to the NYSE Pillar matching engine to indicate the beginning and end of an unstable period (Figure 9).

Figure 9

Model integration with NYSE Pillar trading system (simplified)



8. Conclusion

We have proposed a new machine learning indicator, the SMI, to predict market instability at any given point in time for a given security. Our solution is not just a machine learning model, but a machine learning system. The SMI uses a gradient boosted decision tree model to predict market instability and provides a very fast and explainable model. The speed of prediction of the model allows us to provide a near continuous inference pipeline to move from the traditional timer paradigm to a real-time solution.

The SMI uses data from the consolidated feed as well as NYSE Arca book data to provide these predictions. Liquid symbols with enough training data points

retrieve predictions from each symbol's symbol-specific SMI model, while less active symbols use a market SMI model for inference. The SMI achieves recall rates of at least 90% for the majority of liquid symbols while minimizing the overlocking time windows within a given day.

NYSE Arca's SeMi Orders will leverage this signal to provide an improved trading experience, especially for low latency market participants. When the SMI signals an unstable market, a SeMi Order will be prevented from executing against liquidity on the opposite side of the market until the SMI indicator predicts stable market conditions. Once the SMI signals a stable market, a SeMi Order is allowed to execute discretion up to the midpoint of the PBBO to trade. SeMi Orders avoid undesirable order matching during times of high market volatility, increasing the potential for price improvement for such orders at the cost of slightly reduced fill rates. Our strong performance results and new technology capabilities suggest further opportunities to improve client outcomes using real-time signals. We seek to continue to improve market quality, especially when benefits can be shared among multiple market participants.

Appendix A - Model Feature List

Feature Name	Explanation
ord_buy_qty_10_ms[3]	Total ordered shares on buy side in the last 10 ms
ord_buy_qty_20_ms	Total ordered shares on buy side in the last 20 ms
ord_buy_qty_50_ms	Total ordered shares on buy side in the last 50 ms
ord_buy_qty_100_ms	Total ordered shares on buy side in the last 100 ms
ord_sell_qty_10_ms[3]	Total ordered shares on sell side in the last 10 ms
ord_sell_qty_20_ms	Total ordered shares on sell side in the last 20 ms
ord_sell_qty_50_ms	Total ordered shares on sell side in the last 50 ms
ord_sell_qty_100_ms	Total ordered shares on sell side in the last 100 ms
n_ioc_ord_10ms[3]	Number of IOC distinct orders submitted in the last 10 ms
n_ioc_ord_20ms	Number of IOC distinct orders submitted in the last 20 ms
n_ioc_ord_50ms	Number of IOC distinct orders submitted in the last 50 ms
n_ioc_ord_100ms	Number of IOC distinct orders submitted in the last 100 ms
n_ioc_share_10ms	Number of IOC shares in the last 10 ms
n_ioc_share_20ms	Number of IOC shares in the last 20 ms
n_ioc_share_50ms	Number of IOC shares in the last 50 ms
n_ioc_share_100ms	Number of IOC shares in the last 100 ms
pbbo_updates_1ms[3]	Number of PBBO updates in last 1ms
bid_bps_0[1,3]	Price on book depth level 0 on bid side
bid_bps_1[1,3]	Price on book depth level 1 on bid side
bid_bps_2[1,3]	Price on book depth level 2 on bid side
ask_bps_0[1,3]	Price on book depth level 0 on ask side
ask_bps_1[1,3]	Price on book depth level 1 on ask side
ask_bps_2[1,3]	Price on book depth level 2 on ask side
bid_vol_0[3]	Volume on book depth level 0 on bid side
bid_vol_1[3]	Volume on book depth level 1 on bid side
bid_vol_2[3]	Volume on book depth level 2 on bid side
ask_vol_0[3]	Volume on book depth level 0 on ask side
ask_vol_1[3]	Volume on book depth level 1 on ask side
ask_vol_2[3]	Volume on book depth level 2 on ask side
price_jump_1ms[3]	Sum of price jumps (with direction) within a 1ms sliding window prior to the current PBBO update
price_jump_abs_1ms[3]	Number of price jumps within a 1ms sliding window prior to the current PBBO update
bid_momentum_100us[2,3]	All bid side momentum which is calculated within a 100us sliding window prior to the current PBBO update
bid_momentum_500us[2,3]	All bid side momentum which is calculated within a 500us sliding window prior to the current PBBO update
bid_momentum_1ms[2,3]	All bid side momentum which is calculated within a 1ms sliding window prior to the current PBBO update
bid_momentum_2ms[2,3]	All bid side momentum which is calculated within a 2ms sliding window prior to the current PBBO update
bid_momentum_5ms[2,3]	All bid side momentum which is calculated within a 5ms sliding window prior to the current PBBO update
bid_momentum_10ms[2,3]	All bid side momentum which is calculated within a 10ms sliding window prior to the current PBBO update
ask_momentum_100us[2,3]	All ask side momentum which is calculated within a 100us sliding window prior to the current PBBO update
ask_momentum_500us[2,3]	All ask side momentum which is calculated within a 500us sliding window prior to the current PBBO update
ask_momentum_1ms[2,3]	All ask side momentum which is calculated within a 1ms sliding window prior to the current PBBO update
ask_momentum_2ms[2,3]	All ask side momentum which is calculated within a 2ms sliding window prior to the current PBBO update
ask_momentum_5ms[2,3]	All ask side momentum which is calculated within a 5ms sliding window prior to the current PBBO update
ask_momentum_10ms[2,3]	All ask side momentum which is calculated within a 10ms sliding window prior to the current PBBO update
bid_level_change_1ms[3]	Sum of level shifts (with direction) on bid side in the book within 1ms sliding window prior to the current PBBO update
bid_level_change_abs_1ms[3]	Number of level shifts on bid side in the book within 1ms sliding window prior to the current PBBO update
ask_level_change_1ms[3]	Sum of level shifts (with direction) on ask side in the book within 1ms sliding window prior to the current PBBO update
ask_level_change_abs_1ms[3]	Number of level shifts on ask side in the book within 1ms sliding window prior to the current PBBO update
hour[3]	Hour of current PBBO update time
minute[3]	Minute of current PBBO update time
second[3]	Second of current PBBO update time
ord_buy_n_10_ms	Total orders on buy side in the last 10ms
ord_buy_n_20_ms	Total orders on buy side in the last 20ms
ord_buy_n_50_ms	Total orders on buy side in the last 50ms
ord_buy_n_100_ms	Total orders on buy side in the last 100ms
ord_sell_n_10_ms	Total orders on sell side in the last 10ms
ord_sell_n_20_ms	Total orders on sell side in the last 20ms

Feature Name	Explanation
ord_sell_n_50_ms	Total orders on sell side in the last 50ms
ord_sell_n_100_ms	Total orders on sell side in the last 100ms
cxl_buy_10ms[3]	Cancelled shares on buy side in the last 10ms
cxl_buy_20ms	Cancelled shares on buy side in the last 20ms
cxl_buy_50ms	Cancelled shares on buy side in the last 50ms
cxl_buy_100ms	Cancelled shares on buy side in the last 100ms
cxl_sell_10ms[3]	Cancelled shares on sell side in the last 10ms
cxl_sell_20ms	Cancelled shares on sell side in the last 20ms
cxl_sell_50ms	Cancelled shares on sell side in the last 50ms
cxl_sell_100ms	Cancelled shares on sell side in the last 100ms
exec_share_10ms	Executed share in the last 10ms
exec_share_20ms	Executed share in the last 20ms
exec_share_50ms	Executed share in the last 50ms
exec_share_100ms	Executed share in the last 100ms
exec_order_10ms	Executed order in the last 10ms
exec_order_20ms	Executed order in the last 20ms
exec_order_50ms	Executed order in the last 50ms
exec_order_100ms	Executed order in the last 100ms
ioc_exec_share_10ms	Number of IOC shares executed in the last 10 ms.
ioc_exec_share_20ms	Number of IOC shares executed in the last 20 ms.
ioc_exec_share_50ms	Number of IOC shares executed in the last 50 ms.
ioc_exec_share_100ms	Number of IOC shares executed in the last 100 ms.
ioc_exec_ord_10ms	Number of IOC distinct orders executed in the last 10 ms.
ioc_exec_ord_20ms	Number of IOC distinct orders executed in the last 20 ms.
ioc_exec_ord_50ms	Number of IOC distinct orders executed in the last 50 ms.
ioc_exec_ord_100ms	Number of IOC distinct orders executed in the last 100 ms.

[1] Price on book depth level X is normalized by following formula:

- Bid side:

$$D_{bid,\,Lvl=l} = \frac{P_{mid} - P_{bid,Lvl=l}}{P_{mid}} * 10^6$$

- Ask side:

$$\mathrm{D}_{\mathrm{ask,\,Lvl=l}} = \frac{P_{ask,Lvl=l} - P_{mid}}{P_{\mathrm{mid}}} * 10^6$$

[2] Momentum of volume change in book levels within a time window. This feature consists of book depth information in a continuous period and shows the trend of volume and price movement. For example, if we see volume increasing aggressively on the bid side and decreasing on the ask side, then the price will be more likely to change upwards. After we calculate momentum on each PBBO update, we add all momentum within a 0.1/0.5/1/2/5ms sliding window prior to each PBBO update as additional data points.

[3] Subset of features with high significance during evaluation period and subsequently used in model performance testing.

Appendix B - Test Symbol Universe

NVDA	ARKB	IYR	CMCSA	DAL	PINS	VONG	IYE	SCHD	BKAG
SPY	PDD	C	S	BMY	U	LUMN	PTON	EFA	BTF
QQQ	PLTR	IYW	KSS	NOVA	MTUL	FNDA	UTEN	GOLD	CALF
TQQQ	BABA	NVDY	BAC	CCL	NCLH	IP	PM	D	NOBL
SOXL	ASTS	UCO	WDC	TMV	AAPB	CL	MTCH	VST	FHLC
IWM	USO	AVLV	IREN	MS	SO	URA	UITB	YINN	AIQ
QQQM	IJR	TLH	MDLZ	VTV	AAXJ	Z	DOW	MSFU	VNQI
AAPL	AAPU	FCX	IWN	GME	COLB	CM	DUK	QQQJ	TPR
GOOGL	DDM	VNQ	SPXU	LQD	BSX	UPS	MTUM	ZIM	AOR
AMD	IWD	CLSK	VTI	UPST	MODL	CONL	ADMA	COIN	SDS
IVV	TSM	TLT	XLU	SMCI	TCOM	GNOM	META	VFH	SLV
QLD	SVXY	WFC	CVX	FTNT	BBWI	IWX	DBND	PSTG	DFUV
SPXL	SCHG	RIVN	APDN	TSLL	TSLY	ALLY	CGGO	VCLT	XRAY
VOO	BITU	WMT	VYM	AA	RTX	BHP	EW	FETH	UNL
SOXS	UBER	RDFN	GAP	GDXD	MNST	AVIG	SQQQ	KDP	W
VXX	UWM	KO	TFC	PICB	IOVA	ETHE	GUSH	ULTY	NDAQ
GOOG	GDXJ	SPYG	COP	ONON	EQT	BITO	ROKU	UNG	AMLP
UPRO	QID	SPMO	MMIN	MAGS	APA	FMDE	IGM	MOS	CARR
TNA	VTWO	MSFT	MCHP	SPHB	VZ	HIMS	RETL	GIS	QGRW
TSLA	SBUX	ON	ASST	PG	LCDS	EFIV	VIXY	DIG	MSTR
XLK	TSLZ	QCOM	VOXX	FELG	SU	OVV	BBY	CNQ	EXC
UDOW	JDST	JD	XLB	SPYI	FNDX	EWT	JCI	LYFT	AVTR
AMZN	IWFL	CHWY	SLB	MDT	TSLR	XP	USSG	WTAI	WPM
DIA	INTC	UAL	EBAY	RIO	TRFM	EWY	RIOT	X	CHX
FBTC	IBIT	SOXX	AMDY	ZM	IWP	XLF	DLN	MOAT	TGT
SPYU	ITOT	SQ	CFG	USPX	LUNR	PFE	FLEX	IAT	MMM
XLY	BITX	XHB	FITB	QQQG	SPTM	TECK	ORCL	BAX	ARKF
XLI	SARK	CVS	EWW	IWR	SNPE	FNGO	ENVX	BRRR	BILL
NVDL	JNUG	BERZ	BBAX	XEL	CZR	LRGF	VFC	SE	MMLG
UVXY	FNGD	AMDL	FFIE	EWZ	RKT	JGRO	GDS	CLF	NWS
MU	CMG	AVGO	CPRT	TJX	AI	SPSM	BZ	DPST	EMLP
NVDS	DIS	LI	OXY	AEP	GLW	JNJ	KHC	PENN	IUSG
GLD	DKNG	TSLT	HOOD	QUAL	MO	FCOM	BLV	RPG	RXRX
GBTC	QQXT	XME	TSDD	MRK	BK	ABT	SCHB	FTV	RY
TMF	NKE	CSCO	GM	SNOW	AEO	ITB	AR	BOX	KEY
MRVL	XRT	HPQ	AGQ	MGM	NUAG	HAIN	MNTS	AAOI	FDD

NVDX	ACWI	VT	DVN	VRAX	JPM	SPUS	BNRG	SDVY	DJT
RSP	SHOP	UFIV	CCJ	LUV	KR	IJH	SCHV	ESGU	CAG
TSLQ	GILD	PST	DYNF	DBB	BNS	LQDW	HPE	NVS	BE
SSO	VIXM	CONY	HUT	VPL	FBL	JQUA	SYF	CLS	STNE
AFRM	NEE	GDX	SRTY	TZA	STM	TSLS	TECL	CPB	IYG
XLV	IGV	AMZU	PSQ	LVS	O	ZION	CRM	FAST	XLRE
SMH	NUGT	RBLX	XLC	DFSV	IYH	CRH	DFAR	ERX	XRTX
XLE	BITB	DELL	WOLF	SPLG	GSK	CSX	WBA	OARK	PCAR
KRE	RUN	AZN	FL	SOXQ	HAL	DG	JWN	APH	IHDG
XOM	XOP	URTY	BKR	VRT	ETHA	AGGY	ARKG	CLSM	TXN
PYPL	IVW	TD	NVDU	CTSH	CORZ	TOST	NBSM	ICF	FE
XBI	FNGS	NEM	XLP	BILI	NVAX	EWJ	PAAS	SPYX	OMFL
SVIX	ARKK	SCHW	DGRW	NU	SW	EVRG	AEM	BOIL	KWEB
MARA	KBE	USB	PEP	SHEL	BN	WMB	MET	HIBS	DLTR