Short-Term Price Movement Prediction in NASDAQ Closing Auctions

Charlie Morris, Dawson Haddox, Alan Ngouenet, Henry Morris
Dartmouth College Undergraduate Seniors

Abstract

Predicting short-term price movements is a challenging yet critical task in financial markets, particularly during the closing auctions that set official prices. In this study, we analyze NASDAQ closing cross auction data to forecast the 60-second future price movement of individual stocks relative to a market index. Our dataset comprises 200 stocks over 481 trading days, enriched with nearly 150 engineered features that capture market imbalance, liquidity, price dynamics, and urgency. We compare three models—a linear regression model with Lasso regularization, a one-dimensional convolutional neural network (1D CNN), and a CatBoost gradient boosting model—using mean absolute error (MAE) as our loss function and evaluation metric. Our experimental results indicate that while linear regression and CNNs showed limited effectiveness, the CatBoost model significantly outperformed the baseline, reducing mean absolute error (MAE) by 0.83. Feature importance analysis on a preliminary CatBoost model revealed that 95% of predictive power came from just 18 features, which were the only features included in the model reported here.

Project Github: `https://github.com/chmorris37/Engs106FinalProject/tree/main`

1 Introduction

Financial markets are dynamic and constantly evolving. The ability to accurately predict short-term price movements offers a significant competitive edge to trading firms such as Optiver. In this study, we seek to determine whether machine learning models can accurately predict the 60-second price movement of individual stocks relative to a market index during the daily NASDAQ closing cross auction. The NASDAQ cross auction establishes the official closing prices for securities listed on

the exchange by merging the traditional order book with the price auction data. The price is set at the level that maximizes the number of shares that can be matched. In the final ten minutes of the NASDAQ trading session, the auction book state is made public according to the following schedule.

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.

Our goal is to train models to predict short-term price movements. Specifically, we aim to forecast the 60-second future movement of a stock's weighted average price (WAP) relative to the movement of the broader NASDAQ index. The target variable for this prediction is defined as:

$$Target = \left(\frac{StockWAP_{t+60}}{StockWAP_t} - \frac{IndexWAP_{t+60}}{IndexWAP_t} \right) \times 10000$$

Note: The multiplication by 10,000 converts the result into basis points, where one basis point is equal to 0.01%.

This problem is significant because successful predictions will enable better market making by potentially reducing spreads or increasing market liquidity, and exploring the relationship between stock price movements and broader market movements can be leveraged for a variety of trading strategies. Our approach integrates feature engineering, time series analysis, linear regression, and machine learning to capture meaningful signals from market data. These insights not only aid in forecasting price movements but also contribute to a deeper understanding of stock valuation and the liquidity dynamics at market open and close.

2 Data Description

Our dataset comprised historical NASDAQ data capturing short-term price movements during the ten-minute closing auction period. It included 200 unique stocks over 481 trading days (approximately 1.5 years), with 55 data points per stock per day. The raw data include standard order book entries

such as bid and ask prices, sizes, and weighted average prices (WAP), as well as auction-specific metrics including imbalance size, matched size, and reference price. From these original columns, we derived nearly 150 additional features, including ranking metrics, ratio comparisons, and various summary statistics. The key features that we created could be categorized into imbalance, spread, liquidity, movement, urgency, rolling, and pairwise features.

Here are some specific columns we made: imbalance_ratio, which was a ratio of imbalance size to matched size, imbalance_momentum, which was a normalized change in imbalance size over time. As well as a price_spread that marked the difference between the ask and bid prices, and a relative_spread which is the price_spread normalized by the WAP. This feature was useful because it provides a measure of relative liquidity. We then created bid_plus_ask_sizes which is the total volume of asks and bids as well as liquidity_imbalance which is a volume normalized difference between ask and bid sizes. Price movement features were wap_momentum which is the percent change in WAP over a rolling window, and mid_price_movement which captured the direction of movement in the mid-price. Market_urgency combined price spread and liquidity imbalance, and market_urgency_v2 offered an alternative using the difference between mid-price and volume weighted average price. We employed a variety of paired features like far_price, near_price, and reference_price, to name a few, which provided insight into relative positioning of different stock price points. Finally, our rolling and grouped features like matched_size, bid_price, and ask_price captured trends over time specific periods. We also recognized that some columns contained missing values and so we filled missing values with the mean of the nearest available data points to ensure a filled dataset.

While these features were designed based on market structure principles, their predictive utility was uncertain until we tested them.

3 Evaluation

We used mean absolute error (MAE) as our loss function during training and later to evaluate our model. MAE is the average of the absolute differences between the predicted values and the actual targets. Importantly, MAE is highly interpretable because it offers a direct measure of prediction error on the same scale as the output. This means that MAE is computed in units that are meaningful for the specific task. Unlike Mean Squared Error (MSE), which squares the differences and thus disproportionately emphasizes larger errors, MAE treats all deviations equally. This characteristic can be particularly beneficial when the dataset includes anomalies or when large errors are less critical for the model's intended application. Mathematically, MAE can be formulated as:

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

Lower MAE values indicate smaller discrepancies between predicted and true target values and thereby better model performance. To assess performance, we compared the MAE of our models against both a simple baseline. A basic baseline assumes a constant predicted value of 0. A more intuitive approach leverages auction imbalances: predicting 0.1 for a buy imbalance, -0.1 for a sell imbalance, and 0 for equilibrium. Interestingly, the MAEs of these models are quite similar, with the imbalance model improving the metric by just 0.0007 (see the final section for specific MAE values).

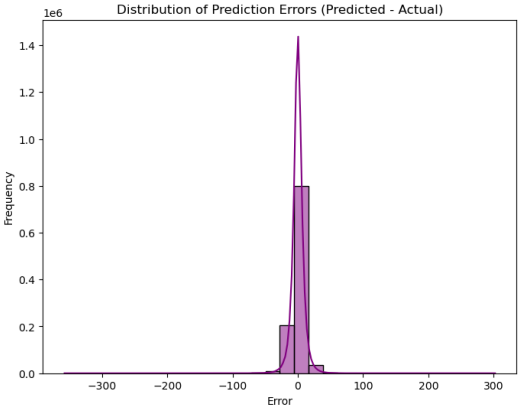
4 Methodology and Results

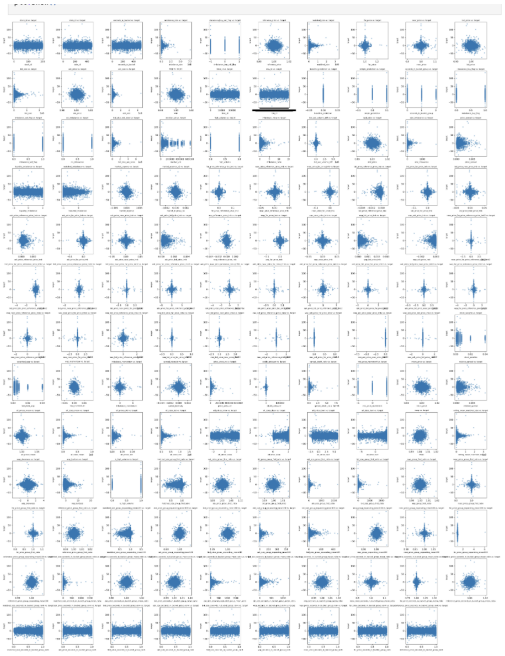
We deployed three main models: a linear model, a one-dimensional convolutional neural network, and a CatBoost model.

4.1 Linear Model

We first explored a linear regression model for forecasting. Specifically, a linear regression model can be defined as $\hat{y} = \vec{w}^T \phi(\vec{x}))$, where $\hat{y}$ refers to the predicted target values, $\vec{w}$ denotes the learned weights, and $\phi(\vec{x})$ represents the basis function transformations of the input feature vector $\vec{x}$. However, this approach proved ineffective due to violations of key assumptions, particularly the lack of strong linear relationships between features and the target. Visual analysis of feature-target scatter plots showed no discernible patterns.

For feature selection, we applied Lasso regularization, which modifies the loss function by adding a penalty proportional to the absolute values of the coefficients. This constraint forces some coefficients to shrink to exactly zero, effectively selecting a subset of features. In our approach, Lasso reduced the number of non-zero coefficients to fewer than 20. Despite this, the model showed only a marginal improvement, reducing the Mean Absolute Error (MAE) by approximately 0.01 from the baseline. The following visualizations highlight the poor performance of the linear model, including the error distribution and the weak feature-target associations. The following visual shows these results (i.e., error distribution and individual relationships between feature and target):





4.2 1D Convolutional Neural Network

Next, we explored a variety of deep learning architectures, as we hypothesized that increased model complexity would facilitate improved modeling of the data. Interestingly, the best of these models was not a Transformer or Long Short-Term Memory model but rather a 1D Convolutional Neural Network (CNN). While we did not learn about 1D CNNs in class, we hypothesized that the approach should theoretically work for time-series data. The core idea behind a 1D CNN is to extract meaningful features by sliding a set of learnable filters (kernels) over the input sequence, detecting local patterns, trends, and dependencies at different scales. Our architecture consisted of multiple convolutional layers with increasing filter sizes to capture hierarchical patterns in the data. Each convolutional layer was followed by a ReLU activation to introduce non-linearity. Then, pooling layers were incorporated to reduce dimensionality and mitigate potential overfitting, complemented by a dropout layer (0.25 rate) for further regularization. Lastly, connected layers mapped the features to our final stock predicting output.

To optimize learning, we used the AdamW optimizer with Mean Absolute Error (MAE) as the loss function. Additionally, a ReduceLROnPlateau scheduler dynamically adjusted the learning rate, decreasing it by a factor of 0.1 if validation loss did not improve for 5 consecutive epochs. For robustness, we trained and evaluated the model using cross-validation with a 10-time-set group gap to prevent data leakage. The data was split into 5 folds, with each fold independently used for training and validation. Surprisingly, the deep learning approach and our CNN model fell short of

expectations. While our CNN model outperformed the baseline, it achieved only a 0.28 improvement in the MAE.

4.3 CatBoost Model

Our final pivot was to a CatBoost model, a gradient boosting algorithm designed for handling categorical data efficiently while mitigating overfitting. Specifically, gradient boosting is an ensemble learning technique that builds models sequentially, where each new model corrects the errors of the previous ones to improve predictive performance. CatBoost improves on traditional gradient boosting by introducing ordered boosting to prevent target leakage, automatic categorical feature encoding, symmetric trees for faster training, and advanced regularization techniques. We trained the model with 3,000 iterations and implemented early stopping if the validation loss failed to improve for 100 rounds. To ensure robust evaluation and prevent data leakage, we used a 10-time-set gap split and validated the model across multiple folds. We ran the model once with all features before using the model from the fold with the lowest MAE to determine feature importance. Based on this analysis, we retained only the top 18 most relevant features that accounted for 95% of the model's performance. Our final CatBoost model significantly outperformed our previous models, achieving a 0.83 improvement in MAE over the baseline, which is a markedly better result than achieved by both the linear regression and CNN models.

Table 1: Feature Importance Scores from CatBoost Model

Index	Feature	Importance
0	vwap	33.679160
1	seconds_in_bucket	15.107620
2	micro_price	10.752042
3	wap	10.696879
4	wap_kurtosis	7.321572
5	market_urgency	2.979449
6	market_urgency_v2	2.576827
7	wap_skewness	2.572450
8	date_id	2.317449
9	time_id	1.987843
10	mid_price	1.874867
11	bid_price	1.845523
12	ask_price	1.511020
13	wap_near_price_imb	1.245032
14	seconds_in_bucket_group	1.062274
15	matched_size_group_expanding_mean100	0.961635
16	wap_reference_price_imb	0.798465
17	wap_group_first_ratio	0.709892

5 Comparison of Model Results

Model	MAE	Baseline MAE Improvement
Predict 0	**6.4078**	-
Imbalance Model	**6.4071**	0.0007
Linear Model	**6.3965**	0.0112
1D CNN Model	**6.1233**	0.2845
CatBoost Model	**5.5800**	0.8277

Table 2: Model Performance Comparison

Important Note: Because of the huge size of the data, we train and test the models on just a fourth of the data because otherwise, our computers crashed every time. The data, with all the features, is over 500 MB when zipped as a CSV.

References

[1] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. arXiv preprint arXiv:1706.09516.