

Trading At The Close

Each trading day on the Nasdaq Stock Exchange concludes with the Nasdaq Closing Cross auction. This process establishes the official closing prices for securities listed on the exchange. These closing prices serve as key indicators for investors, analysts and other market participants in evaluating the performance of individual securities and the market as a whole. Almost 10% of Nasdaq's average daily volume occurs in the closing auction. Providing true price and size discovery, the closing auction determines benchmark pricing for index funds and other investment strategies.

In the last ten minutes of the Nasdaq exchange trading session, market makers like Optiver merge traditional order book trading with price auction data. This ability to consolidate information from both sources is critical for providing the best prices to all market participants.

We are excited to see how Kagglers approach this unique & challenging pricing problem!

In this notebook we will cover:

- Financial concepts related to trading at the close
- Simple data exploration
- Simple baseline model

Auction

In financial markets, an auction is a mechanism for determining the price of a particular asset by allowing multiple buyers and sellers to interact directly in a controlled, regulated environment. There are many different kinds of auction, but for this competition we are particularly interested in the concept of a closing auction.

In a closing auction, orders are collected over a pre-determined timeframe and then matched at a single price determined by the buy & sell demand expressed by auction participants. For Nasdaq Closing auctions, the exchange begins accepting orders at the start of the trading day and begins publishing the state of the auction book at 3:50pm ET for 10 minutes before the market closes at 4pm ET, at which point the orders are matched instantly at a single price. See below the timeline for Nasdaq closing auctions, taken from public documentation

https://nasdaqtrader.com/content/ETFs/closing_cross_faqs.pdf

(https://nasdaqtrader.com/content/ETFs/closing_cross_faqs.pdf).

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. <ul style="list-style-type: none">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. <ul style="list-style-type: none">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 bookIO 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.

The closing price is determined as: The price at which the maximum number of shares can be ***matched***. In the event the auction would have equal number of matched lots at different levels, Nasdaq uses a proprietary algorithm which takes into account the last traded price, the price-time priority of orders, and the available liquidity at different price levels. It is quite rare that the price at which the maximum shares can be matched is not unique.

We will now dive deeper into the order book mechanics, and define some key terms such as the one italicised above.

Order book

The term order book refers to an electronic list of buy (also known as bid) and sell (also known as ask) orders for a specific security or financial instrument organized by price level.

Below is a simple example of an order book in continuous (non-auction) trading. We can see that at a price level of 9 there are 2 shares on the bid, meaning the market participants are willing to buy 2 shares for a price of 9. Similarly we see that at a price level of 10 there is 1 share on the ask, meaning market participants are willing to sell 1 share for a price of 10.

Bid	Price	Ask
	10	1
2	9	
0	8	

For this kind of order book in continuous trading, the highest bid price will always be less than the lowest ask price. For example, if someone were to put in an ask of 10 shares at a price of 9, then 2 shares would be **matched** since there is someone willing to buy 2 shares at a price of 9. *The new best ask* would be 8 shares at a price of 9, the new state of the book is displayed below.

Bid	Price	Ask
	10	1
	9	8
0	8	

Auction order book

The auction order book behaves slightly differently to the order book for continuous trading. In this book, the orders are not immediately matched, but instead collected until the moment the auction ends. You might notice in the example below that unlike the order book for continuous trading, the highest bid is not greater than the lowest ask. The book in the example below is referred to as **in cross**, since the best bid and ask are overlapping. The closing auction price is therefore referred to as the **uncross** price, the price at which the shares which were in cross are matched.

Bid	Price	Ask
	10	1
3	9	2
4	8	4

For the auction book example above, we can work through definitions for a few key terms. Suppose the auction uncrossed with the book in this state, then:

- At a price of 10, 0 lots would be matched since there are no bids ≥ 10 .
- At a price of 9, 3 lots would be matched, as there are 3 bids ≥ 9 and 6 asks ≤ 9 .
- At a price of 8, 4 lots would be matched, since there are 7 bids ≥ 8 , and there are 4 asks ≤ 8 .

So the price which maximises the number of matched lots would be 8. We would therefore describe the auction order book in the following way:

- The uncross price is 8
- The matched size would be 4
- The **imbalance** would be 3 lots in the buy direction.

The term imbalance refers to the number of unmatched shares. At the uncross price of 8, there are 7 bids & 4 asks which can be matched, therefore we are left with 3 bids unmatched. Since bids are orders to buy, there is an imbalance of 3 lots in the buy direction.

The term **far price** refers to the hypothetical uncross price of the auction book, if it were to uncross at the reporting time. Nasdaq provides far price information 5 minutes before the closing cross.

Combined book

Combining the two books can result in a more accurate reflection of the market's buying and selling interest at different price levels. This aids in better price discovery, allowing the market to reach a more accurate and fair equilibrium price when the auction uncrosses.

Here we combine our original order book example & our auction book example by aggregating the buying & selling interest across all price levels.

<i>Bid</i>	<i>Price</i>	<i>Ask</i>
	10	2
5	9	2
4	8	4

For this combined book:

- At a price of 10, 0 lots would be matched as there are no bids ≥ 10 .
- At a price of 9, 5 lots would be matched, as there are 5 bids ≥ 9 and 6 asks ≤ 9 .
- At a price of 8, 4 lots would be matched, as there are 9 bids ≥ 8 and 4 asks ≤ 8 .

So, the price which maximizes matched lots would be the price of 9. We would therefore describe the combined order book in the following way:

- The uncross price is 9
- The matched size is 5
- The imbalance would be 1 lot, in the sell direction.

The hypothetical uncross price of combined book is called the ***near price***. Nasdaq provides near price information 5 minutes before the closing cross.

Nasdaq also provides an indication of the fair price called the ***reference price***. The reference price is calculated as follows:

- If the near price is between the best bid and ask, then the reference price is equal to the near price
- If the near price $>$ best ask, then reference price = best ask
- If the near price $<$ best bid, then reference price = best bid So the reference price is the near price bounded between the best bid and ask.

Competition Data

In this competition, Kagglers are challenged to predict the short term price movements during the 10 minute auction period. All the training data is contained within a single train.csv file. Please refer to <https://www.kaggle.com/competitions/optiver-trading-at-the-close/data> (<https://www.kaggle.com/competitions/optiver-trading-at-the-close/data>) & the introduction in this notebook for specific column definitions.

- `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 (in USD).
- `imbalance_buy_sell_flag` - An indicator reflecting the direction of auction imbalance.
 - 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/ask]_price` - Price of the most competitive buy/sell level in the non-auction book.
- `[bid/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.

$$\frac{BidPrice * AskSize + AskPrice * BidSize}{BidSize + AskSize}$$

- `seconds_in_bucket` - The number of seconds elapsed since the beginning of the day's closing auction, always starting from 0.
- `target` - The 60 second future move in the wap of the stock, less the 60 second future move of the synthetic index.
 - The synthetic index is a custom weighted index of Nasdaq-listed stocks constructed by Optiver for this competition.
 - The unit of the target is basis points, which is a common unit of measurement in financial markets. A 1 basis point price move is equivalent to a 0.01% price move.
 - Where t is the time at the current observation, we can define the target:

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

All size related columns are in USD terms.

All price related columns are converted to a price move relative to the stock wap (weighted average price) at the beginning of the auction period.

sample_submission A valid sample submission, delivered by the API. See this notebook for a very simple example of how to use the sample submission.

revealed_targets When the first `time_id` for each date (i.e. when `seconds_in_bucket` equals zero) the API will serve a dataframe providing the true `target` values for the entire previous date. All other rows contain null values for the columns of interest.

public_timeseries_testing_util.py An optional file intended to make it easier to run custom offline API tests. See the script's docstring for details. You will need to edit this file before using it.

example_test_files/ Data intended to illustrate how the API functions. Includes the same files and columns delivered by the API. The first three date ids are repeats of the last three date ids in the train set, to enable an illustration of how the API functions.

optiver2023/ Files that enable the API. Expect the API to deliver all rows in under five minutes and to reserve less than 0.5 GB of memory. The first three date ids delivered by the API are repeats of the last three date ids in the train set, to better illustrate how the API functions. You must make predictions for those dates in order to advance the API but those predictions are not scored.

```
In [1]: import pandas as pd
import numpy as np
import plotly.express as px

train = pd.read_csv('D:/OneDrive/Jupyter_Cloud/kaggle_optiver/kaggle-optiver-t
rading-at-the-close/train.csv')
train.head()
```

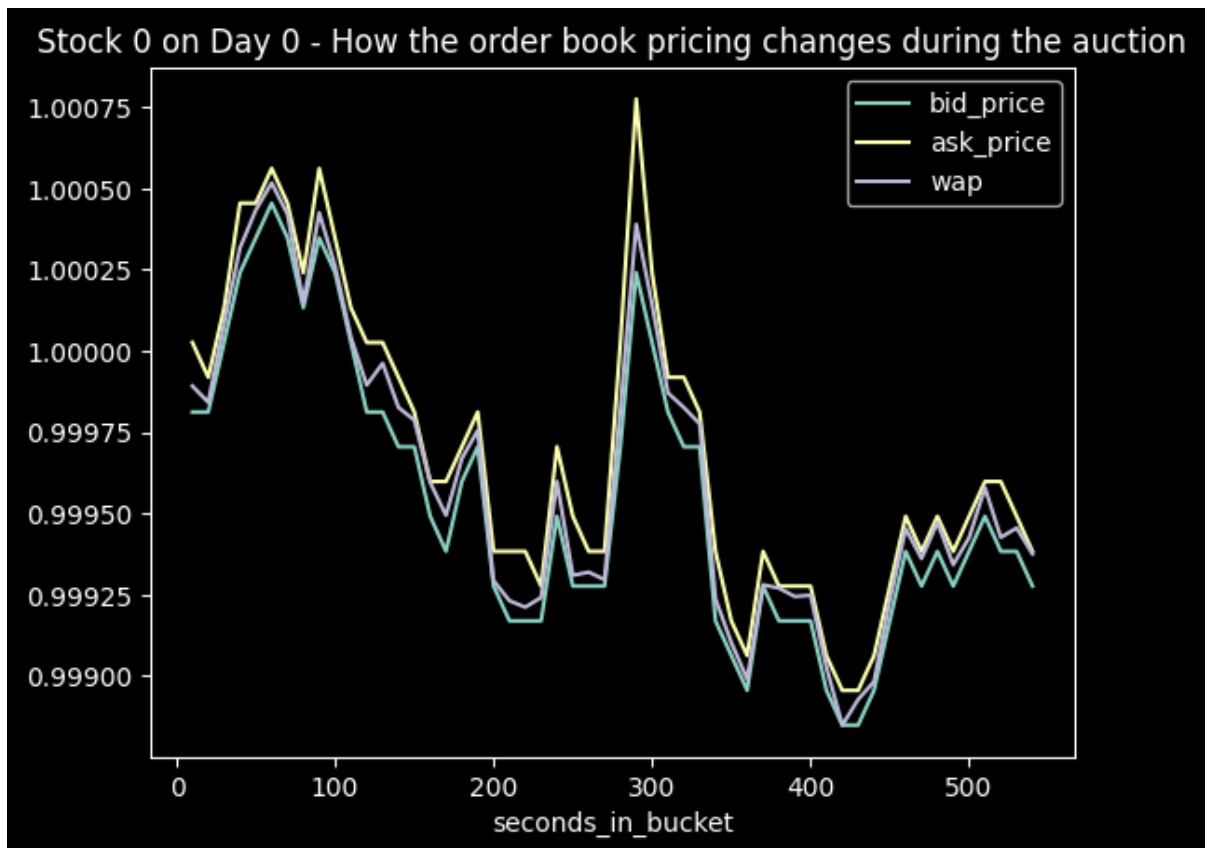
Out[1]:

	stock_id	date_id	seconds_in_bucket	imbalance_size	imbalance_buy_sell_flag	reference_price
0	0	0	0	3180602.69	1	0.999812
1	1	0	0	166603.91	-1	0.999896
2	2	0	0	302879.87	-1	0.999561
3	3	0	0	11917682.27	-1	1.000171
4	4	0	0	447549.96	-1	0.999532



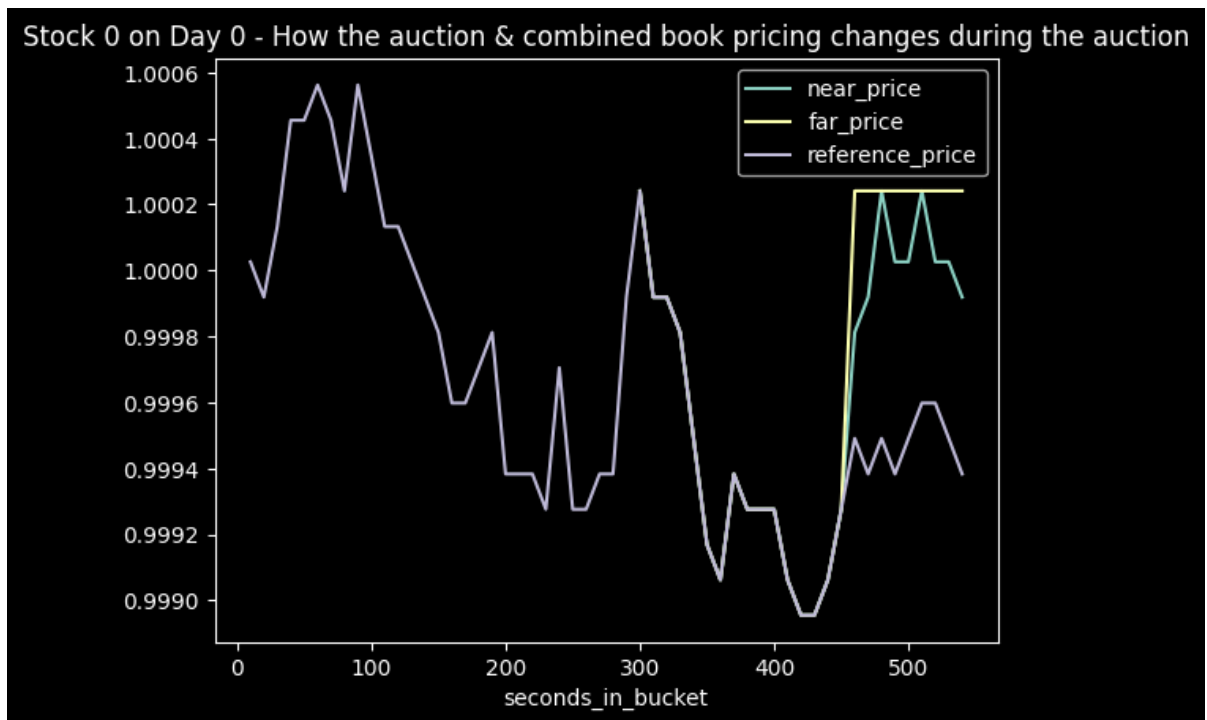
```
In [2]: (
    train
    .query('stock_id ==0 & date_id ==0')
    [['seconds_in_bucket', 'bid_price', 'ask_price', 'wap']]
    .replace(0, np.nan)
    .set_index('seconds_in_bucket')
    .plot(title='Stock 0 on Day 0 - How the order book pricing changes during
the auction')
)
```

```
Out[2]: <Axes: title={'center': 'Stock 0 on Day 0 - How the order book pricing change
s during the auction'}, xlabel='seconds_in_bucket'>
```



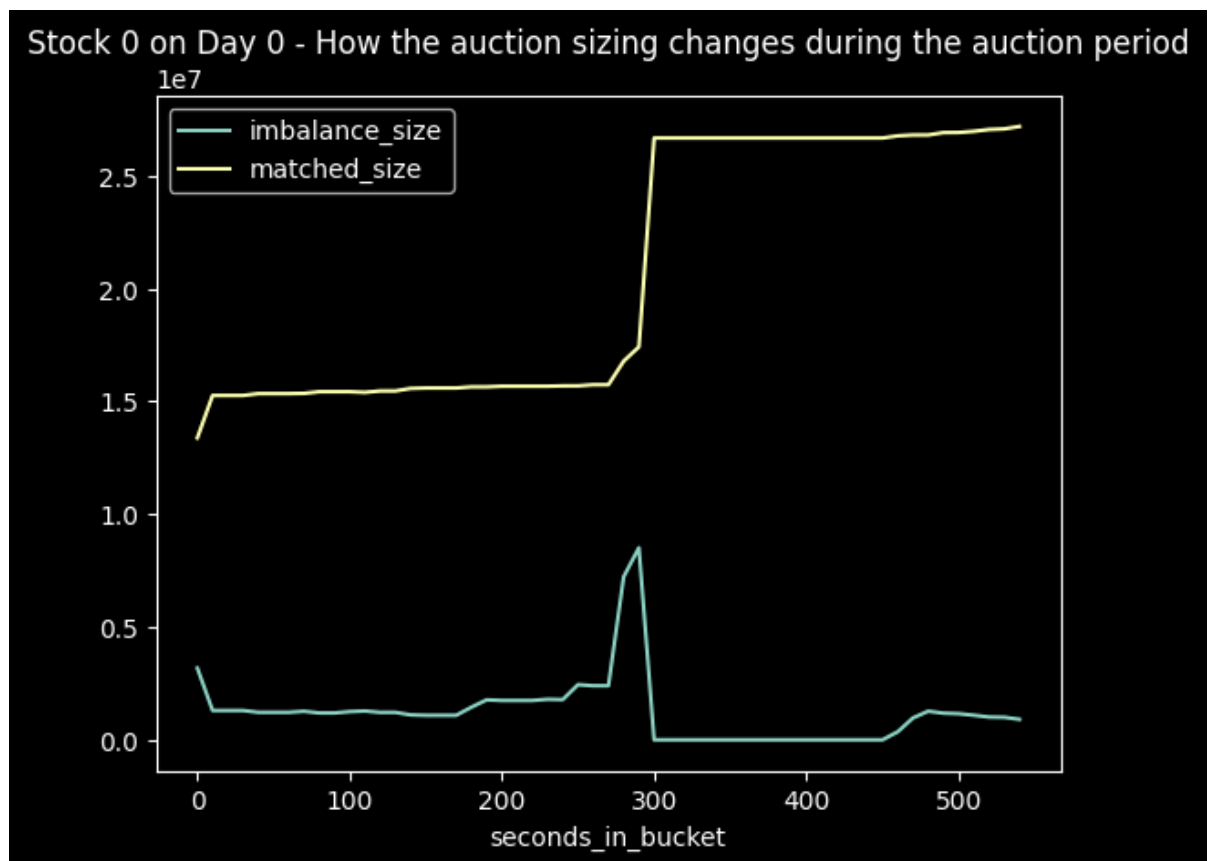

```
In [3]: (
    train
    .query('stock_id ==0 & date_id ==0')
    [['seconds_in_bucket', 'near_price', 'far_price', 'reference_price']]
    .replace(0, np.nan)
    .set_index('seconds_in_bucket')
    .plot(title = 'Stock 0 on Day 0 - How the auction & combined book pricing
changes during the auction')
)
```

Out[3]: <Axes: title={'center': 'Stock 0 on Day 0 - How the auction & combined book pricing changes during the auction'}, xlabel='seconds_in_bucket'>



```
In [4]: (
    train
    .query('stock_id ==0 & date_id ==0')
    [['seconds_in_bucket', 'imbalance_size', 'matched_size']]
    .set_index('seconds_in_bucket')
    .plot(title='Stock 0 on Day 0 - How the auction sizing changes during the
    auction period')
)
```

Out[4]: <Axes: title={'center': 'Stock 0 on Day 0 - How the auction sizing changes during the auction period'}, xlabel='seconds_in_bucket'>



Baseline Solutions

A simple baseline is to assume we have no valuable information about the direction any stock moves, which translates to a predicted value of 0 for all observations. This baseline is quite hard to beat in the context of financial markets.

However, we have some information in our dataset that should help us to beat this baseline. If we observe an auction imbalance, it indicates that at the current price there is buying or selling interest that will currently not get matched in the auction. We can therefore adjust our prediction upwards if there is a buy imbalance & downwards if there is a sell imbalance.

```
In [5]: simple_mapping = {
        1: 0.1,
        0: 0,
        -1: -0.1
    }

train['baseline_prediction'] = 0
train['simple_prediction'] = train['imbalance_buy_sell_flag'].map(simple_mapping)
```

Evaluation

Our evaluation metric is Mean Absolute Error (MAE). The formula is given by:

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

Where:

- n is the total number of data points.
- y_i is the predicted value for the i^{th} data point.
- x_i is the observed value for the i^{th} data point.

```
In [6]: baseline_mae = (train['baseline_prediction'] - train['target']).abs().mean()
simple_prediction_mae = (train['simple_prediction'] - train['target']).abs().mean()

print(baseline_mae, simple_prediction_mae)
print('MAE improvement in basis points: ', (baseline_mae-simple_prediction_mae))
```

```
6.40777074811524 6.407056596608261
```

```
MAE improvement in basis points: 0.000714151506978844
```