Optiver - Trading at the Close

Post-match Review

Kevin (Zhongkai) Xue

The Chinese University of Hong Kong, Shenzhen

December 31, 2023

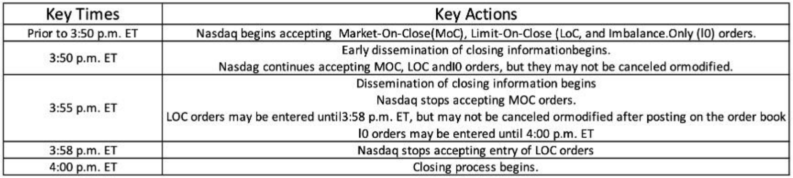
Description

Featured Code Competition
Optiver - Trading at the Close
Predict US stocks closing movements
$100,000
Prize Money

- *Trading at the Close (2023)* is organized by the well-known market maker Optiver and hosted on the *Kaggle* platform.
- Participants are challenged to develop a model utilizing **stock order book** and **closing auction data** to predict the **closing price trends** of hundreds of Nasdaq-listed stocks.
- The model can help integrate signals from the auction and the order book, thereby enhancing market efficiency and accessibility, especially during the critical last ten minutes.

Background

The Nasdaq **closing auction** mechanism determines the closing price at the end of the trading day and handles a large volume of orders. The process and purpose of the closing auction are to ensure that the market closes at a fair and representative price.

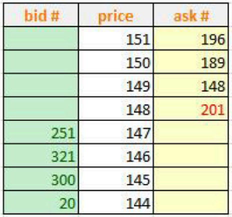


Key Times	Key Actions
Prior to 3:50 p.m. ET	Nasdaq begins accepting Market-On-Close(MoC), Limit-On-Close (LoC, and Imbalance.Only (I0) orders.
3:50 p.m. ET	Early dissemination of closing informationbegins. Nasdag continues accepting MOC, LOC andI0 orders, but they may not be canceled ormodified.
3:55 p.m. ET	Dissemination of closing information begins Nasdaq stops accepting MOC orders. LOC orders may be entered until3:58 p.m. ET, but may not be canceled ormodified after posting on the order book I0 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.

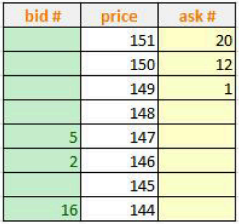
Background (cont'd)

Order Book: The order book is an electronic record system that lists all the unfulfilled buy and sell orders in the market, displaying **each** order's price and quantity **in real-time**.

Auction Book: The auction book is used during specific times, such as market opening or closing, to collect and match **a large volume of** buy and sell orders in a **single** transaction to determine a balanced price.



bid #	price	ask #
	151	196
	150	189
	149	148
	148	201
251	147	
321	146	
300	145	
20	144	



bid #	price	ask #
	151	20
	150	12
	149	1
	148	
5	147	
2	146	
	145	
16	144	

Data Overview

The organizer offers 3 price references with characteristics:

- **Reference Price:** The reference price is a price determined by considering the following order: maximizing the number of matched shares, minimizing the imbalance, and minimizing the distance from the midpoint between the best bid and best ask prices. It can also be understood as the price between the best bid and best ask prices.
- **Far Price:** The far price is calculated based on auction interest to determine the cross price that maximizes the number of matched shares, excluding continuous market orders. This calculation excludes continuous market orders.
- **Near Price:** The near price is calculated based on both auction and continuous market orders to determine the cross price that maximizes the number of matched shares.

Data Overview (cont'd)

Other data consists of common order flow trading features:

- `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.

Label

target refers to the 60 second future move in the WAP of the stock, less the 60 second future move of the synthetic index. Only provided for the train set.

- The synthetic index is a custom weighted index of Nasdaq stocks constructed by Optiver.
- 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.
- *target* could be defined as (t for current observation):

$$\textit{target} = \left(\frac{\text{StockWAP}_{t+60}}{\text{StockWAP}_t} - \frac{\text{IndexWAP}_{t+60}}{\text{IndexWAP}_t} \right) \times 10000$$

Evaluation Metrics

The outputs are evaluated on the Mean Absolute Error (MAE) between the predicted return and the observed target.
The formula is given by:

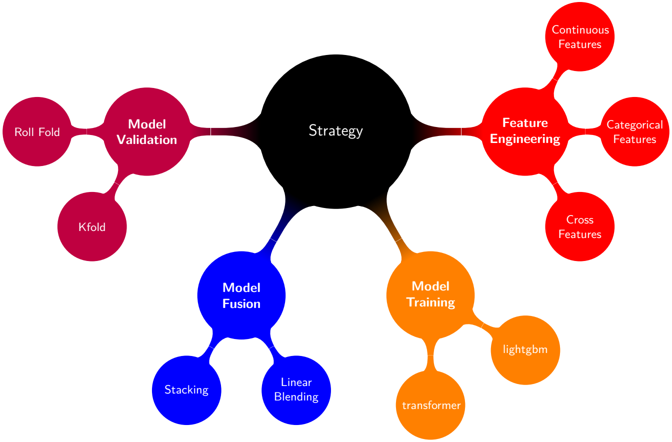
$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i|$$

Where:

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

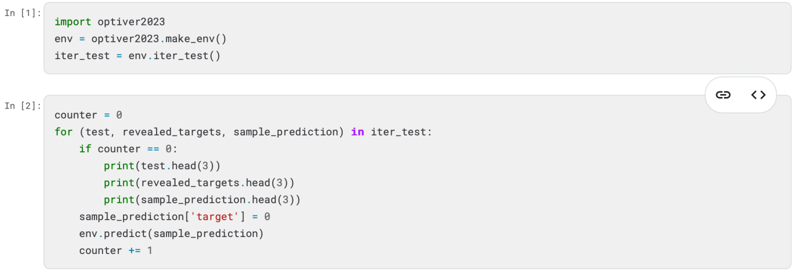
Pipeline Explanation

Here is an intuitive mind map for a frame-like data mining task:



Baseline Model: Single Value Imputation

The baseline indicates the required data format for submission:

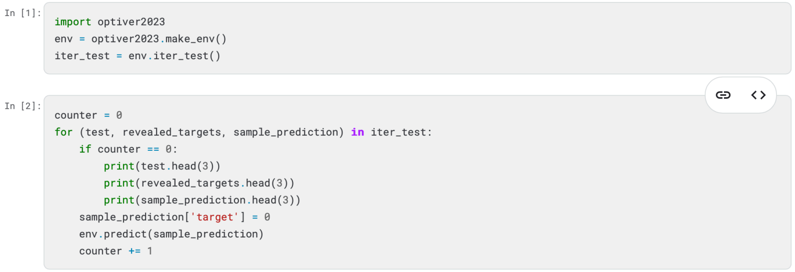


```
In [1]:
import optiver2023
env = optiver2023.make_env()
iter_test = env.iter_test()
```

```
In [2]:
counter = 0
for (test, revealed_targets, sample_prediction) in iter_test:
    if counter == 0:
        print(test.head(3))
        print(revealed_targets.head(3))
        print(sample_prediction.head(3))
    sample_prediction['target'] = 0
    env.predict(sample_prediction)
    counter += 1
```

Baseline Model: Single Value Imputation

The baseline indicates the required data format for submission:



```
In [1]:
import optiver2023
env = optiver2023.make_env()
iter_test = env.iter_test()
```

```
In [2]:
counter = 0
for (test, revealed_targets, sample_prediction) in iter_test:
    if counter == 0:
        print(test.head(3))
        print(revealed_targets.head(3))
        print(sample_prediction.head(3))
    sample_prediction['target'] = 0
    env.predict(sample_prediction)
    counter += 1
```

Index Coefficient Decomposition

The weights of the synthetic index can be rebuilt by applying linear regression on the stock & index return:

```
lr = LinearRegression()
y = index_return.reshape(-1)
X = stock_returns.reshape((num_stocks, -1)).T

mask = ~((np.isnan(y) | np.isnan(X).any(axis=1)))
X, y = X[mask], y[mask]
```

By predicting the weighted average price (WAP) of each stock at time *t* for the next 60 seconds and then combining them into an index, we can utilize more granular information, reduce prediction errors, and potentially achieve better performance in a complex time series forecasting task.

Financial Features

- **Order Book Imbalance Features:** Bid/Ask Price, Reference Price, Imbalance Ratio
- **Rolling Features:** Moving Average (MACD), Bollinger Band Breakout, Historical Volatility, Relative Volatility (RSI)
- **Index Features:** Market Status, Index - Stock Features
- **Correlation Features:** KMeans
- **Shift Features:** Target Shift

Model Training and Fusion

- LightGBM is a high-performance, efficient gradient boosting framework based on **decision tree algorithms**.
- It is well-suited for the task due to its speed, accuracy, and ability to handle large datasets efficiently.
- Additionally, it effectively handles **categorical features** and **missing values**, which are common in financial data.

Reflection and Summary

Advantages:

- Combines a normal data analysis pipeline with optimizations.
- Performs feature engineering on multiple main aspects.
- Finally integrates model ensemble techniques, resulting in good accuracy and robustness.

Disadvantages:

- The approach is still somewhat conventional.
- The code contains redundant parts, lacking vectorization.
- Without special management, there is a risk of data leakage.

Thanks!