Diogo Franquinho

November 2023

1 Task Overview

In Python, using the data output from part 1, you will create a predictive model. The program should, at a minimum:

- Load the data output from part 1.
- Calculate a collection of predictive features, such as the difference between buy and sell contracts at the top prices ("bq0 - aq0").
- Select and generate data for a future prediction target, documenting the rationale behind the choice.
- Optionally subsample the data, providing justification for this decision.
- Utilize Lasso regression to predict your chosen target and integrate the features into a final predictive signal.

Document the choices made and the reasons behind them. Aim to demonstrate how you would construct reusable library code for research purposes, focusing on organization, structure, and adherence to good coding practices.

Please submit your code as a single archive (.zip or .tar.gz) file, including documentation on how to run it.

2 Part 1: Generating Order Book Data

2.1 Introduction to the OrderBook Class

The `OrderBook` class is meticulously crafted to simulate and manipulate an order book for trading data. It integrates robust libraries such as `pandas`, `numpy`, and `OrderedDict` for efficient management and processing of trading orders. We utilize five CSV files as input, each containing columns for "timestamp", "side", "action", "id", "price", and "quantity".

2.2 Class Definition and Attributes

The `OrderBook` class is characterized by two principal attributes: `buy_orders` and `sell_orders`. These attributes are instances of `OrderedDict`, employed to store buy and sell orders, with the price serving as the key.

2.3 Constructor Method

```
class OrderBook:
    def __init__(self):
        # Initializes the OrderBook with empty buy and sell orders.
```

The `__init__` method is pivotal in initializing an `OrderBook` instance, establishing the foundational structure for storing buy and sell orders.

2.4 Considerations

This section delves into various critical aspects of the `OrderBook` class:

- **Design Choices:** The use of `OrderedDict` is deliberate for maintaining order sequence.
- In the `update_order_book()` function, we sort the dictionary at every iteration. This is essential when a new order arrives at the first level (L1) to ensure immediate access to that order when extracting the top layers using `get_top_k_levels`.
- The decision to continuously store the dictionary with all levels and updates stems from a need for comprehensive data. For instance, if an incoming order involves deleting an order at the first level, information about the new fifth level (previously sixth level) becomes crucial.
- While exploring order matching, we experimented with a function to eliminate orders if a buy order matched the price of a sell order and vice versa. However, as the output files remained unchanged, it appears that actual trade executions (where buy and sell orders match) are processed externally.
- **Potential Improvements:** Enhancing class functionality and efficiency by optimizing data storage and processing algorithms.
- Consider retaining only 10 to 15 levels of the dictionary during data processing for efficiency.

2.5 Results

The following demonstration illustrates the script's usage. As a result, five new CSV files are generated, each containing columns for 'timestamp', 'price', 'side', 'bp0', 'bq0', 'bp1', 'bq1', 'bp2', 'bq2', 'bp3', 'bq3', 'bp4', 'bq4', 'ap0', 'aq0', 'ap1', 'aq1', 'ap2', 'aq2', 'ap3', 'aq3', and 'ap4', 'aq4'.

```
# Usage demonstration
order_book = OrderBook()
day_files = ['res_20190610.csv', 'res_20190611.csv', 'res_20190612.csv', 'res_20
for day_file in day_files:
    order_book.create_output_for_dayfile(day_file, 5)
```

3 Part 2: Modelling and Signal Generation

3.1 Introduction to the PricePredictionModel Class

The `PricePredictionModel` class is designed for creating a logistic regression-based price prediction model. It includes a comprehensive approach to feature selection and model training, aiming to predict price movements accurately.

3.2 Class Definition and Attributes

The `PricePredictionModel` class encompasses several attributes essential for the model, such as `data_files` for storing input data and `features` for the initial set of features considered for the model.

3.3 Constructor Method

```
class PricePredictionModel:
    def __init__(self, data_files):
        # Constructor method initializing the model with data and features.
```

The constructor initializes the model with data files and predefines a set of features to be used in the prediction model.

4 Feature Engineering

Feature engineering is a critical step in preparing the market data for our prediction model. This section provides a detailed description of the engineered features and their formulations.

4.1 Basic Feature Formulations

- **Mid Price:** Calculated as the average of the best bid and ask prices.

$$\text{mid\_price} = \frac{\text{bp0} + \text{ap0}}{2}$$

- **Bid-Ask Spread:** The difference between the best ask price and the best bid price.

$$\text{bid\_ask\_spread} = \text{ap0} - \text{bp0}$$

- **Normalized Spread:** The bid-ask spread normalized by the mid price.

$$\text{normalized\_spread} = \frac{\text{bid\_ask\_spread}}{\text{mid\_price}}$$

- **Total Bid Depth:** Sum of the bid quantities across the top 5 levels.

$$\text{total\_bid\_depth} = \sum_{i=0}^{4} \text{bq}_i$$

- **Total Ask Depth:** Sum of the ask quantities across the top 5 levels.

$$\text{total\_ask\_depth} = \sum_{i=0}^{4} \text{aq}_i$$

4.2 Advanced Feature Formulations

- **Layer-wise Imbalance:** For each of the five layers, the imbalance is calculated as the difference between bid and ask quantities normalized by their sum.

$$\text{layer\_}n\text{\_imbalance} = \frac{\text{bq}_{n\text{-}1} - \text{aq}_{n\text{-}1}}{\text{bq}_{n\text{-}1} + \text{aq}_{n\text{-}1}}$$

- **Rate of Change of Layer 1 Imbalance:** The first derivative of the layer 1 imbalance.

$$\text{roc\_layer\_1\_imbalance} = \Delta\text{layer\_1\_imbalance}$$

- **Exponential Moving Average of ROC Imbalance:** Calculated over a span of 50 data points.

$$\text{exp\_ma\_roc\_imbalance\_span50} = \text{EMA}_{50}(\text{roc\_layer\_1\_imbalance})$$

- **Gap Difference Between Levels 1 and 2:** The difference in gaps between the best bid prices (bp0, bp1) and the best ask prices (ap0, ap1).

$$\text{gap\_difference\_l1l2} = (\text{bp0} - \text{bp1}) - (\text{ap0} - \text{ap1})$$

- **Log Returns:** The natural logarithm of the ratio of consecutive mid prices.

$$\text{log\_returns} = \log\left(\frac{\text{mid\_price}_t}{\text{mid\_price}_{t-1}}\right)$$

- **Realized Variance:** The variance of log returns over a rolling window of 30 data points.

$$\text{realized\_variance} = \text{Var}_{30}(\text{log\_returns})$$

- **Realized Bipower Variation:** Calculated as the rolling sum of the product of absolute log returns and their lagged values, over a window of 30.

$$\text{realized\_bipower\_variation} = \sum_{t=30} \left|\text{log\_returns}_t \times \text{log\_returns}_{t-1}\right|$$

This feature engineering process enhances the dataset by extracting and synthesizing key information from raw market data, thus enabling a more nuanced analysis for the prediction model. The features I sleceted were in part taken from Ntakaris et al. 2019.

5 Choosing a Prediction Target for Trading

5.1 Introduction to Prediction Target Selection

This section delves into the critical process of selecting a suitable prediction target for trading. The aim is to identify a target that provides not only insightful data but also augments the efficacy of our trading strategies.

5.2 Options Considered for Prediction Targets

The evaluation process encompassed several potential targets, including:

- **Target 1:** A fixed target based on the average number of mid-price changes per trading day, inspired by Kolm, Turiel, and Westray 2023 section 3.2. The number of mid-price changes per trading day, denoted as N, and the total number of microseconds per trading day (3.6e+10 in this case) were calculated. This led to the formulation of $\delta = \frac{3.6e+10}{N}$, serving as the target for each order book update. Interval experiments were conducted with $k\delta$, where $k \in \{1, ..., 10\}$.

- **Brief Discussion**: The computational efficiency was challenged when using all the data. Moreover, with updates potentially occurring outside the top 5 levels, there was a risk of repetitive training on identical data, as feature values remained constant. Often, the price change was zero, potentially leading to the loss of information about price fluctuations within the specified timeframe.

- **Target 2:** Prediction triggered by any change in the mid-price.

- **Brief Discussion**: This target was examined each time a change in the mid-price occurred. The prediction of the subsequent mid-price change was based solely on information available up to that point.

Both targets are product-invariant, catering to varying timeframes for more or less liquid products, a characteristic deliberately sought in this research.

Insight into the latency of the trading system was limited, though it was understood that ultra-low-latency systems employing ASICs chips were utilized. Drawing from previous experience with FPGA-based systems, where the latency from the data center to the exchange was approximately 20 microseconds, a round-trip target of 40 microseconds was deemed achievable. Naturally, these figures could vary based on the system in use and its geographical proximity to the exchange. The following figure (1) illustrates quantiles for the second day, estimating the unfeasibility of acting on the signal in less than 10% of cases, given the timeframe constraints.

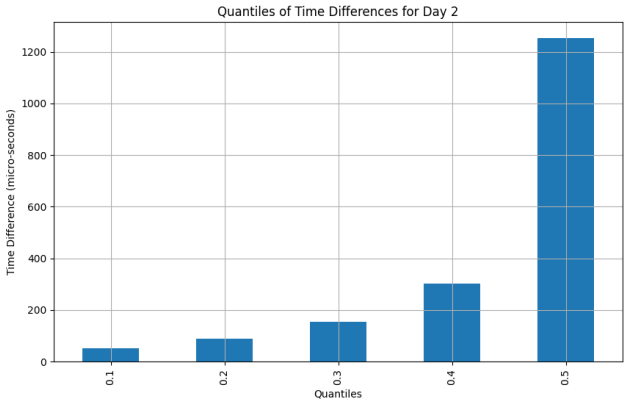

Figure 1: Quantiles of the microseconds until the prediction target for day 2

5.3 Rationale Behind the Chosen Target

The decision to select Target 2, which focuses on changes in the mid-price, stemmed from a comprehensive evaluation of its advantages, particularly when contrasted with the limitations of Target 1.

- **Drawbacks of Target 1:** The first target, while theoretically sound, presented practical challenges. The computational inefficiency of using the entire dataset, combined with the risk of redundant training due to unchanging features outside the top 5 levels, significantly diminished its utility. Furthermore, the frequent occurrence of zero price change within the specified intervals could lead to the omission of vital information about price dynamics.
- **Simplicity and Efficacy of Target 2:** In contrast, Target 2 offers a more streamlined approach by triggering predictions with every mid-price change. This method ensures that the model is consistently engaged with meaningful data changes, enhancing the relevance and timeliness of the predictions. It simplifies the modelling framework by focusing on direct and significant market movements, thereby reducing the complexity inherent in handling extensive datasets with minimal changes.

In light of these considerations, Target 2 was chosen as it effectively addresses the limitations of Target 1 while offering a more practical and efficient framework for predicting price movements in trading. The simplicity of this approach, coupled with its direct relevance to market fluctuations, makes it a robust choice for the predictive model.

6 Model Training

For the division of data into Training, Validation, and Test Sets, a Temporal Split method was employed. This approach is particularly congruent with the time series nature of the data, ensuring that the model is tested on unseen, future data, which mimics real-world trading scenarios.

- **Training Set:** Data from the first three days constituted the Training Set. This segment of the data was utilized to train the model, providing it with the initial learning phase and acquainting it with the fundamental trends and patterns in the data.
- **Validation Set:** The fourth day's data was designated as the Validation Set. This set is pivotal in the model development process, offering a means to fine-tune parameters and assess the model's predictive capabilities, thus ensuring that it is not overfitting to the training data.
- **Test Set:** Lastly, the data from the fifth day was used as the Test Set. This final evaluation phase is crucial for gauging the model's effectiveness in predicting future market movements, providing a stringent test of its generalizability to new, unseen data.

Upon establishing this data segmentation, the model was then fitted to a Logistic Regression framework with L1 regularization. The use of L1 regularization, often referred to as Lasso Regression, is instrumental in enhancing the model's generalization by imposing a penalty on the absolute size of the coefficients. This approach not only helps in feature selection by driving some coefficients to zero but also combats overfitting, making it highly suitable for our predictive modeling in a complex and dynamic trading environment.

6.1 Results and Observations

The optimal C value of 1 suggests a balance, where the model is complex enough to capture the underlying patterns in the data without overfitting.

The classification report provides deeper insights:

- **Precision**: Indicates the proportion of positive identifications that were actually correct. In the context of mid-price prediction, it reflects the model's accuracy in correctly predicting price movements.
- **Recall**: Shows the proportion of actual positives that were identified correctly. This is crucial in financial contexts, as failing to predict a significant price movement could mean missing out on potential opportunities.
- **F1-Score**: Balances precision and recall. A high F1-score suggests a robust model that is accurate and reliable in predicting mid-price movements.

The classification report for the best model (at $C = 1$) is as follows:

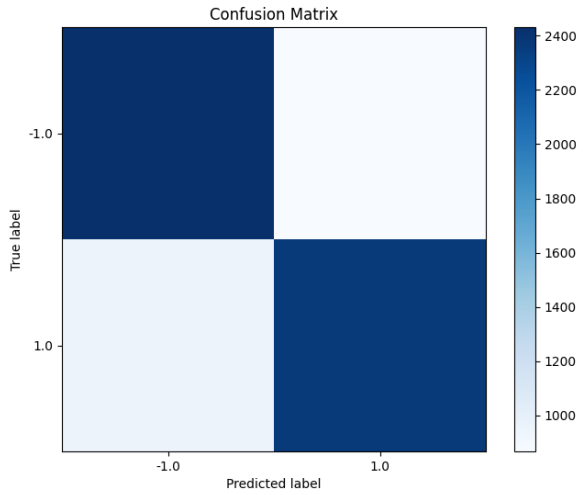

Figure 2: Confusion Matrix of the Prediction Model

	precision	recall	f1-score	support
-1.0	0.72	0.74	0.73	3299
1.0	0.73	0.71	0.72	3330
accuracy			0.72	6629
macro avg	0.73	0.72	0.72	6629
weighted avg	0.73	0.72	0.72	6629

The balanced values across these metrics indicate that the model performs relatively consistently across both classes (-1.0 and 1.0). The accuracy of 0.72 on the test set also supports the model's effectiveness in classifying the data.

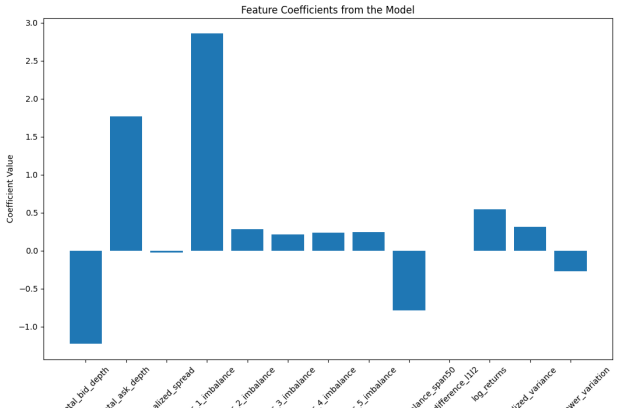

Figure 3: Coefficients of the 15 Features Used

7 Additional Considerations

7.1 Feature Selection using SBFS (Sequential Backward Feature Selection)

To identify the most influential features in our model, a function employing Sequential Backward Feature Selection (SBFS) from the mlxtend library was developed. This approach effectively reduces the feature space while retaining significant predictors. For an in-depth understanding of the Sequential Feature Selector, one can refer to the mlxtend user guide.

This function, when applied to select three key features, determined that 'layer_1_imbalance', 'exp_ma_roc_imbalance_span50', and 'log_returns' were the most critical.

7.2 Coefficients

A notable observation was the negative coefficient for the rate of change of volume imbalance in the first layer. This could suggest a correlation between volume imbalance and its rate of change, warranting further analysis. On the other hand, the positive coefficient associated with 'log_returns' might imply a momentum effect in the market, though this remains speculative and requires more rigorous investigation.

```
Classification Report:
              precision    recall  f1-score   support

        -1.0       0.71      0.74      0.73      3447
         1.0       0.73      0.70      0.72      3457
```

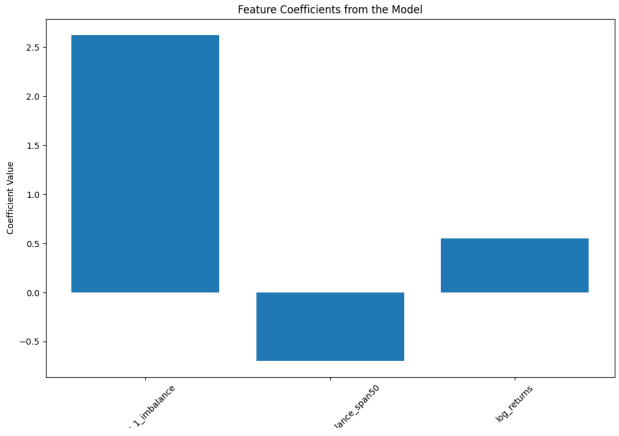

Figure 4: Coefficients of the model with 3 features

accuracy			0.72	6904
macro avg	0.72	0.72	0.72	6904
weighted avg	0.72	0.72	0.72	6904

7.3 Overall Performance

The overall accuracy of the model is 72%, indicating its substantial predictive power. This level of accuracy, coupled with consistent macro and weighted averages for precision, recall, and F1-score, underscores a balanced performance across different classes.

The refinement achieved through $L1$ regularization, resulting in a 'Best C value' of 0.1, confirms that the selected features are potent predictors, requiring less regularization.

8 Conclusion and Implications for Trading Strategies

Our model effectively predicts both rises and falls in mid-prices, crucial for algorithmic trading. With a 72% accuracy, it outperforms random guessing, though the fast-paced order book changes present practical challenges in signal implementation.

Notably, reducing the feature set to three didn't compromise accuracy and highlighted key predictive features. This also streamlined the model, evident

from the Lasso regularization parameter dropping from 1 to 0.01.

Future work could focus on further feature engineering and exploring correlations among the selected features. This paves the way for more efficient, real-time trading models, bridging the gap between theoretical finance and practical trading.

References

[1] Petter N Kolm, Jeremy Turiel, and Nicholas Westray. "Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book". In: *Mathematical Finance* 33.4 (2023), pp. 1044–1081.

[2] Adamantios Ntakaris et al. *Feature Engineering for Mid-Price Prediction with Deep Learning*. 2019. arXiv: `1904.05384 [q-fin.ST]`.