# Short-Term Volatility Prediction Using Financial Market Microstructure Data

BYUNGHWA CHOI
HEEGYU EOM
HYEONJONG HONG
SEHOEN KIM

# CONTENTS


01 Problem Overview

02 System Pipeline

03 Feature Engineering

04 Modeling

05 Results

1

# Problem Overview

# Competition Selection

- **Competition Name**
  - Optiver Realized Volatility Prediction
- **Host**
  - Optiver, a global trading firm that provides liquidity through fast algorithmic trading
- **Objective**
  - Build robust models to predict short-term realized volatility for stocks across various sectors
- **Data**
  - High-frequency financial market data (i.e., price and order book information recorded at extremely short time intervals)
- **Note**
  - Competition had been already ended




Figure 1. Kaggle Competition Page

# Motivation(1)

- Financial markets can change rapidly, with prices fluctuating significantly even within a short period of time.




Figure 2. S&P500 6M movement before and during the Middle East Conflict

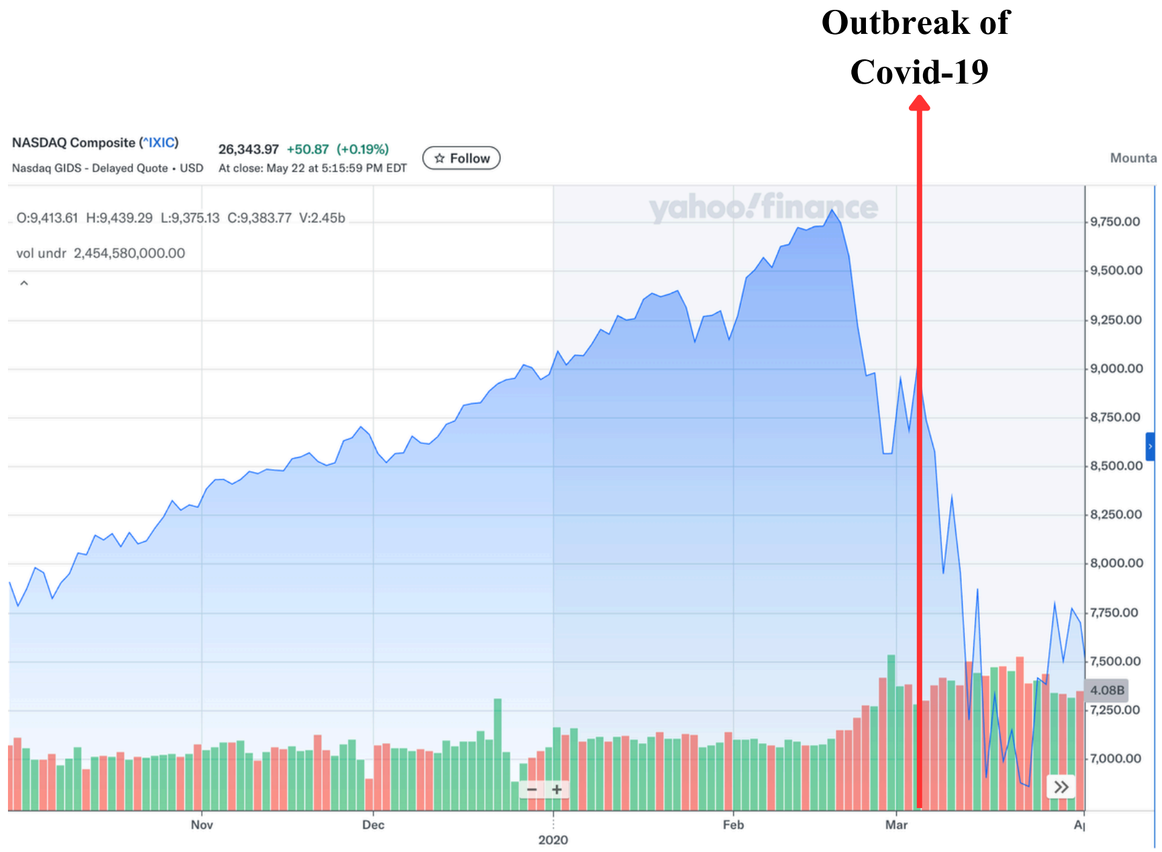



Figure 3. NASDAQ 6M movement before and after Covid-19

# Motivation(2)

- Sudden spikes in volatility can lead to panic selling and large losses.
- Better volatility forecasting can help investors manage risk more effectively.

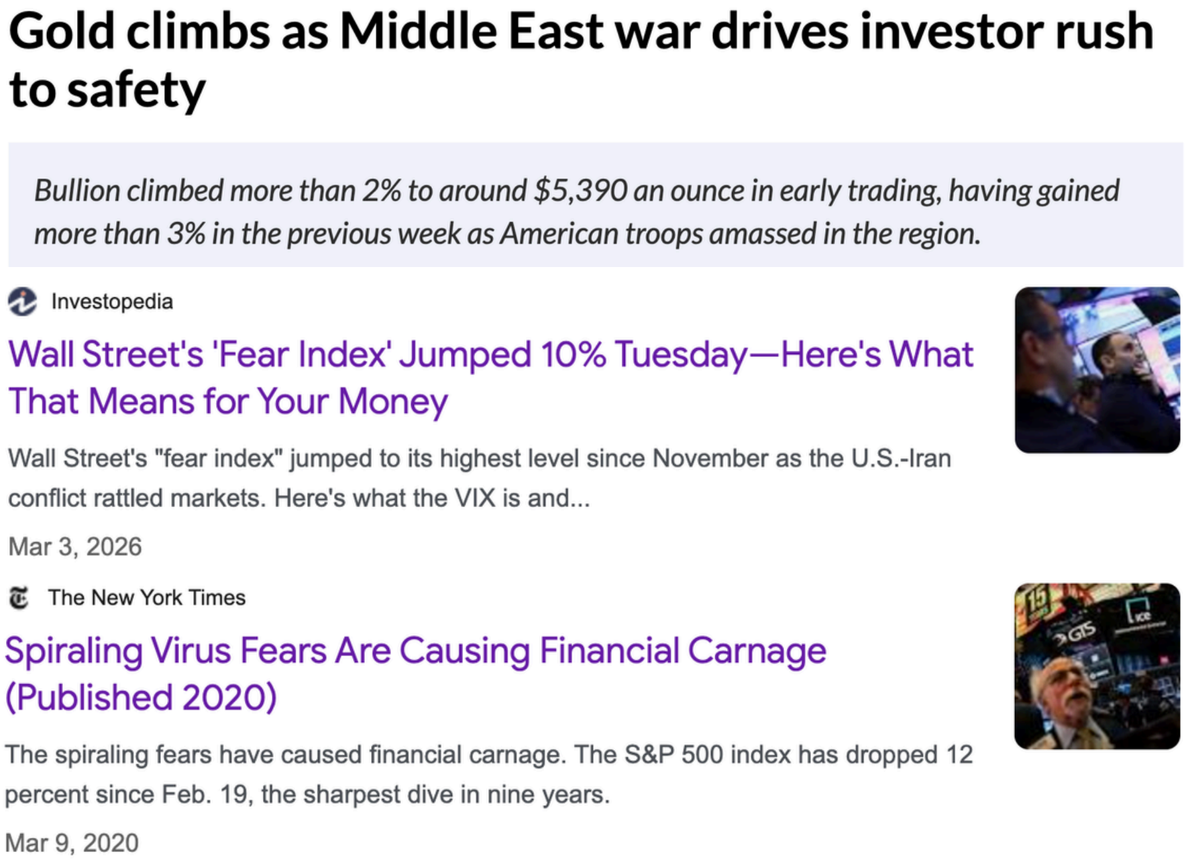



Figure 4. New Headlines during Market Crisises

# Problem Definition

- **Goal**
  - Predict future realized volatility of stock
  - Note that project is not to forecast stock price itself or stock price direction.
  - Rather we aim to predict how much the stock price is expected to fluctuate in the future.

- **Unit of Analysis**
  - Each observation corresponds to a (stock_id, time_id) pair.
  - ‘stock_id’ is the unique identifier of a stock, and ‘time_id’ represents a non-overlapping individual 10-minute window.

- **Input Variables**
  - High-frequency & Micro-structure market data for each (stock_id, time_id) pair.
  - Includes 1-second resolution order book and trade data within each 10-minute window.

- **Target Variable**
  - Realized volatility of the subsequent 10-minute window.

# Dataset : Order Book Data

- Order book data shows buying and selling interest in the market, rather than actual executed transactions.
- The dataset contains top-2 most competitive bid/ask prices and volumes at a 1-second resolution.

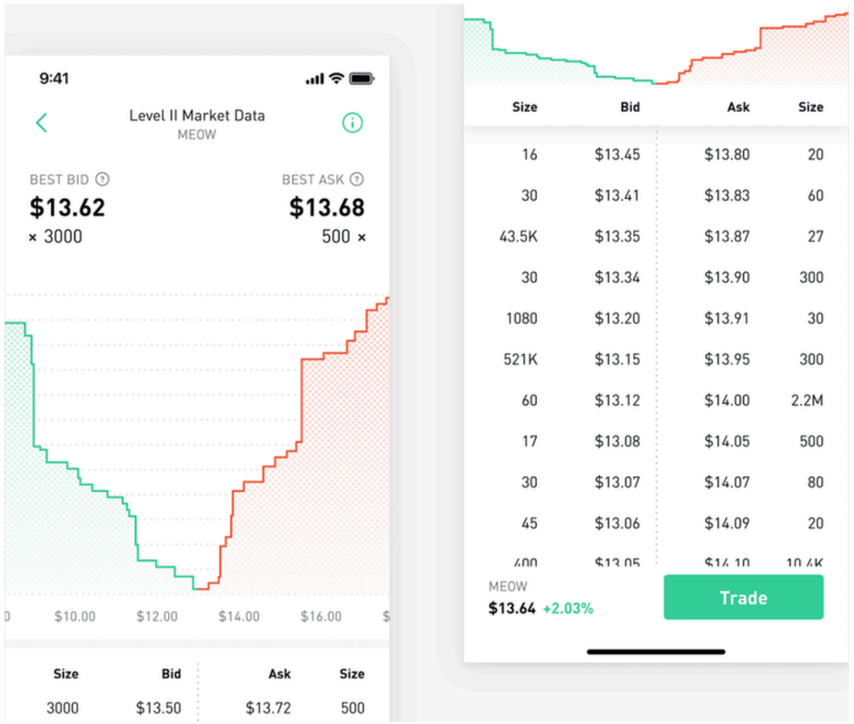



Figure 5. Example of Order Book

| Columns | Explanation |
|---|---|
| stock_id | Unique ID of each stock |
| time_id | ID representing a specific 10-minute time window |
| seconds_in_bucket | Number of seconds elapsed within the 10-minute window |
| bid_price1 | Highest buying price currently offered |
| bid_price2 | Second highest buying price |
| ask_price1 | Lowest selling price currently offered |
| ask_price2 | Second lowest selling price |
| bid_size1 | Number of shares requested at the best bid price |
| bid_size2 | Number of shares requested at the second best bid price |
| ask_size1 | Number of shares requested at the best ask price |
| ask_size2 | Number of shares requested at the second best ask price |

Table 1. Key Columns of the Order Book Dataset

# Dataset : Trade Data

- Trade data contains information on transactions that were actually executed in the market.
- Compared to order book updates, trade events occur less frequently and are therefore more sparse.

| Feature | Explanation |
|---|---|
| stock_id | Unique ID of each stock |
| time_id | ID representing a specific 10-minute time window |
| seconds_in_bucket | Number of seconds elapsed within the 10-minute window |
| price | Average executed transaction price |
| size | Total number of shares traded |
| order_count | Number of executed trade offers |

Table 2. Key Columns of the Trade Dataset

2

# Modeling Pipeline

# Modeling Pipeline

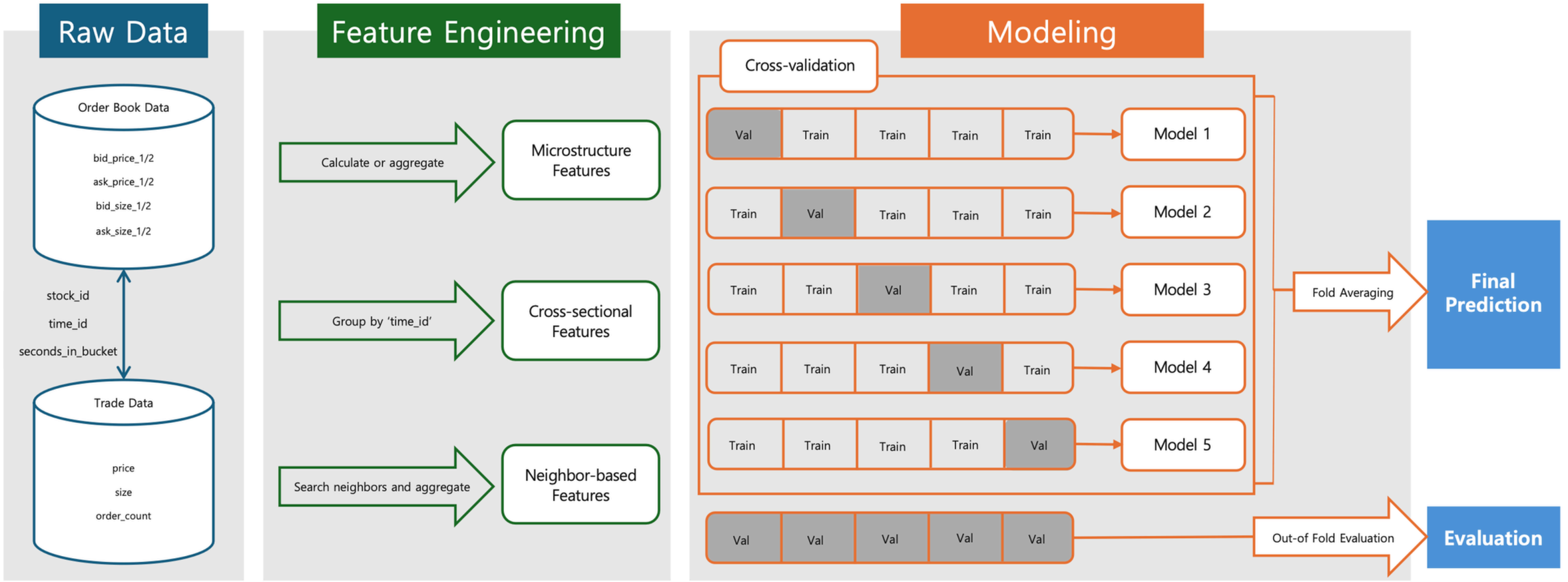



Figure 6. Modeling Pipeline

3

# Feature Engineering

# Why Feature Engineering?

- **Raw data is highly noisy.**
  - Order book and trade activity fluctuate every second. (Figure 7)
  - Directly using raw data may obscure underlying market patterns

- **Raw data can be transformed into informative microstructure features.**
  - Prior studies suggest that liquidity, order imbalance and trading activity contain useful information about market behavior.
  - Therefore, instead of directly using noisy raw quotes, we engineered microstructure features.
  - $WAP_1 = \frac{bid_price_1 \times ask_size_1 + ask_price_1 \times bid_size_1}{bid_size_1 + ask_size_1}$ (Figure 8)

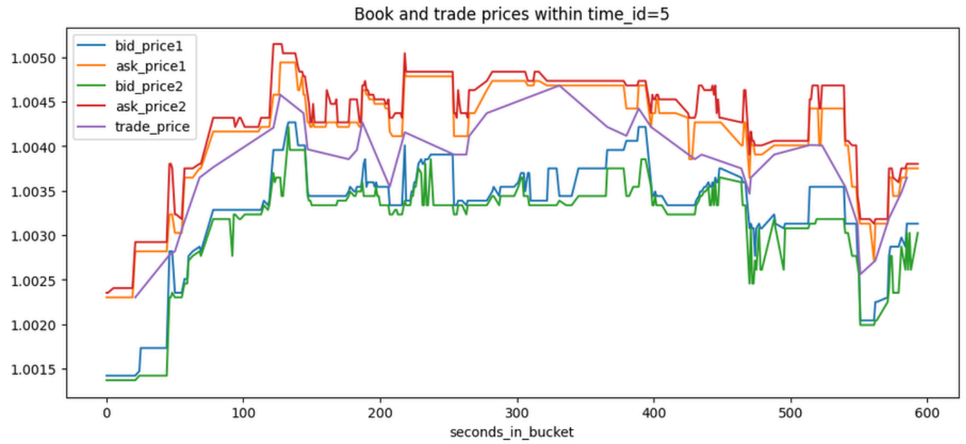



Figure 7. Raw quote and trade dynamics

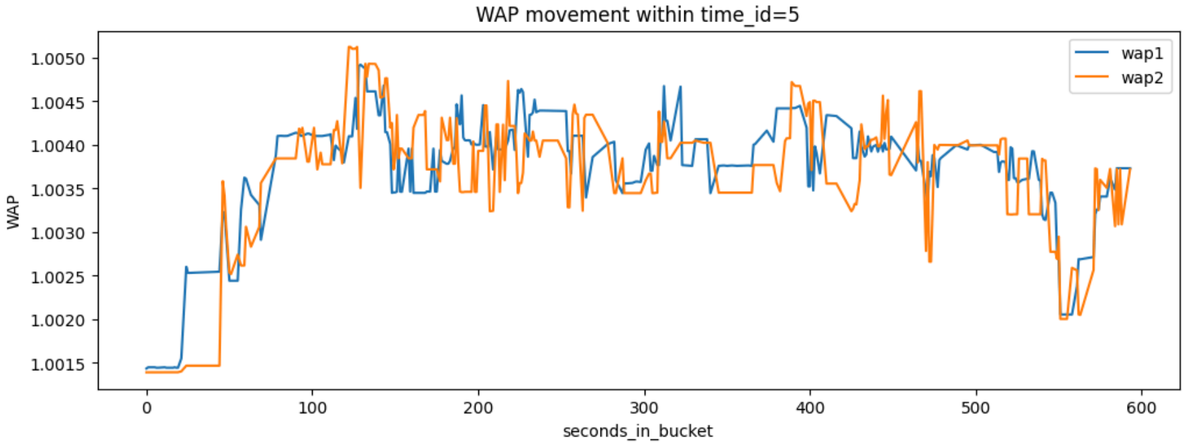
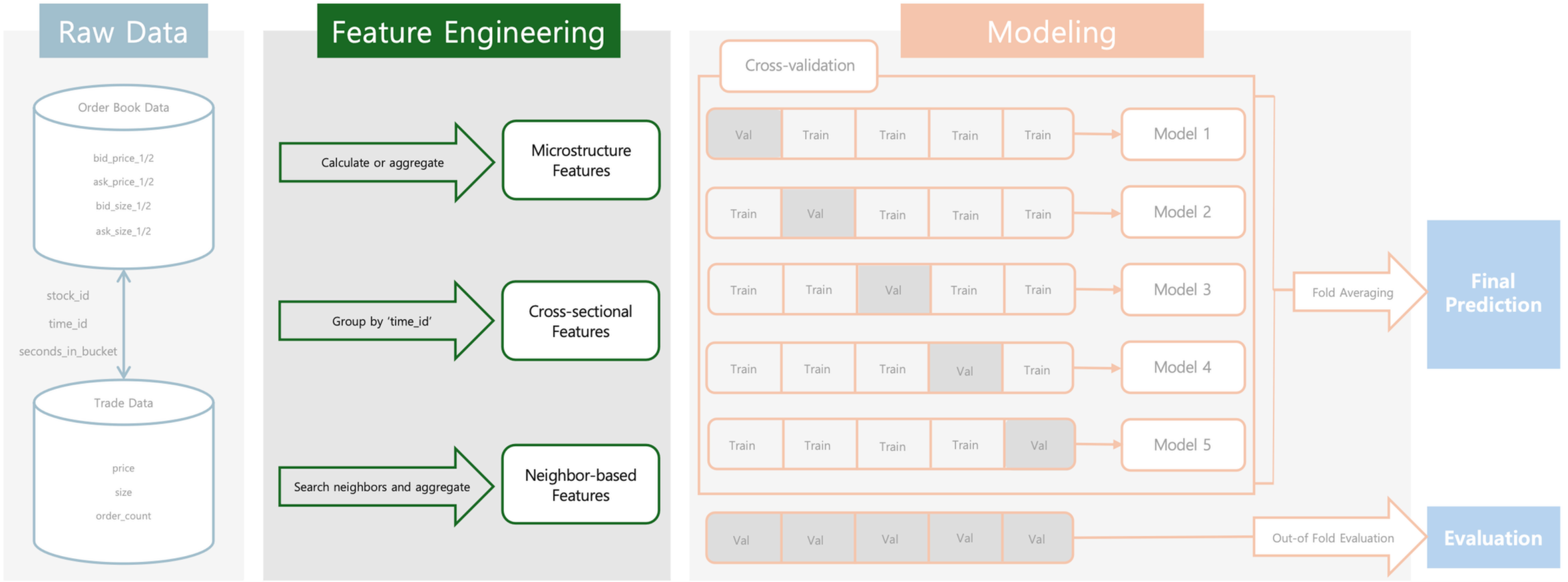



Figure 8. Engineered WAP dynamics

# Modeling Pipeline



# Microstructure Features

- Features are designed to capture short-term price movement, liquidity, trading pressure and temporal market dynamics.

| Category | Example features | Reason |
|---|---|---|
| Price movement | WAP, log return | Captures Short-term price fluctuation |
| Liquidity | spread, total volume, order count | Reflects how active and liquid the market is |
| Imbalance | volume imbalance, size imbalance | Indicates dominance of buying or selling pressure |
| Aggregation | mean, standard deviation | Summary of 10-minute window |
| Sub-window Features | Features over 0s, 150s, 300s, 450s | Captures how market condition change within 10-minute window |

Table 3. Summary of microstructure features

# Cross-Sectional Features

- **How were they generated?**
  - For each ‘time_id’, we computed statistics across all stocks.
  - e.g., ‘book_log_return1_realized_volatility_mean_stock’ is average of realized volatility across stocks within same time window.

- **What information do they capture?**
  - Stocks are often affected by common market conditions.
  - Utilizing cross-sectional features, we could incorporate market-wide volatility information during the same period.




Figure 9. News Headlines for cross-sectional features

# Neighbor-based Features

- **How were they generated?**
  - For each stock, five most similar stocks were chosen and their volatility related features were aggregated.
  - Similarity was based on microstructure features which reflect trading behavior and liquidity. (e.g., spread, total volume...)
  - Volatility features were used because feature importance analysis showed they were highly predictable.
  - e.g., ‘book_log_return1_realized_volatility_nn5_median’ is the median volatility of five similar stocks within same time window.

- **What information do they capture?**
  - Stocks are affected not only by overall market conditions, but also by assets with similar trading patterns.
  - These features capture volatility information from structurally similar stocks

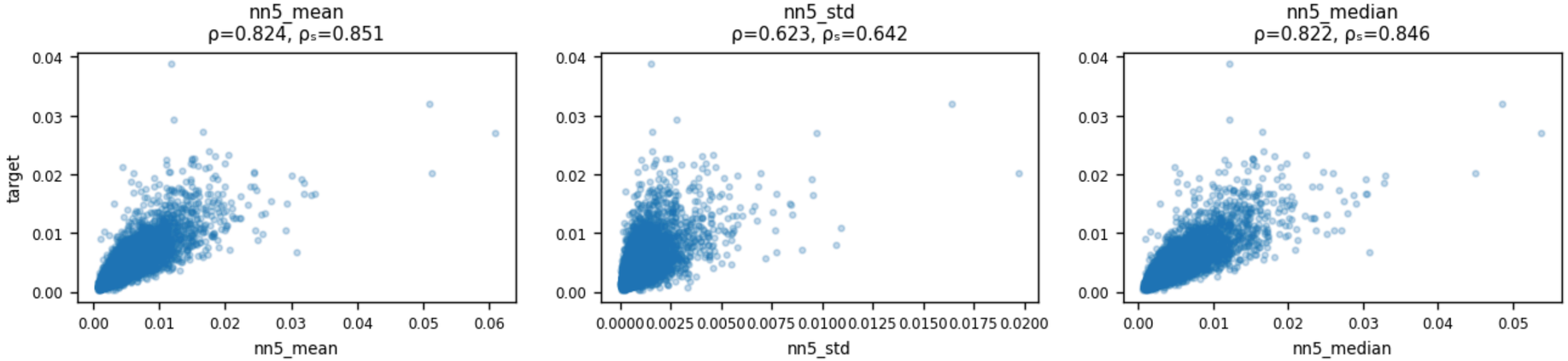



Figure 10. Relationship between neighbor-based features and future realized volatility

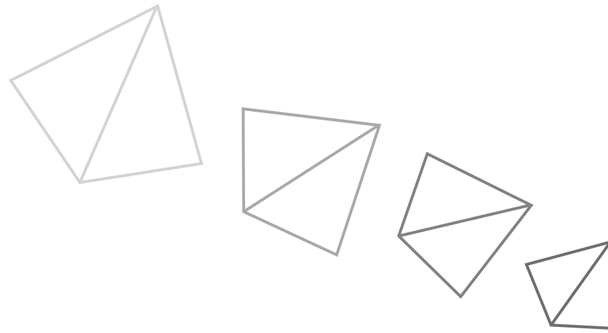
# Summary

- In total, 327 features were generated from order book and trade data.

- Features were designed to capture short-term market microstructure, cross-sectional market conditions, and volatility information from similar stocks.

| Feature Type | Number of features | Description |
|---|---|---|
| Microstructure Features | 264 | WAP, spread, imbalance, volume, sub-window aggregation etc. |
| Cross-sectional Features | 48 | Market-wide volatility related statistics across |
| Neighbor-based Features | 15 | Neighbor’s volatility related statistics within the same time window |
| Total | 327 | |

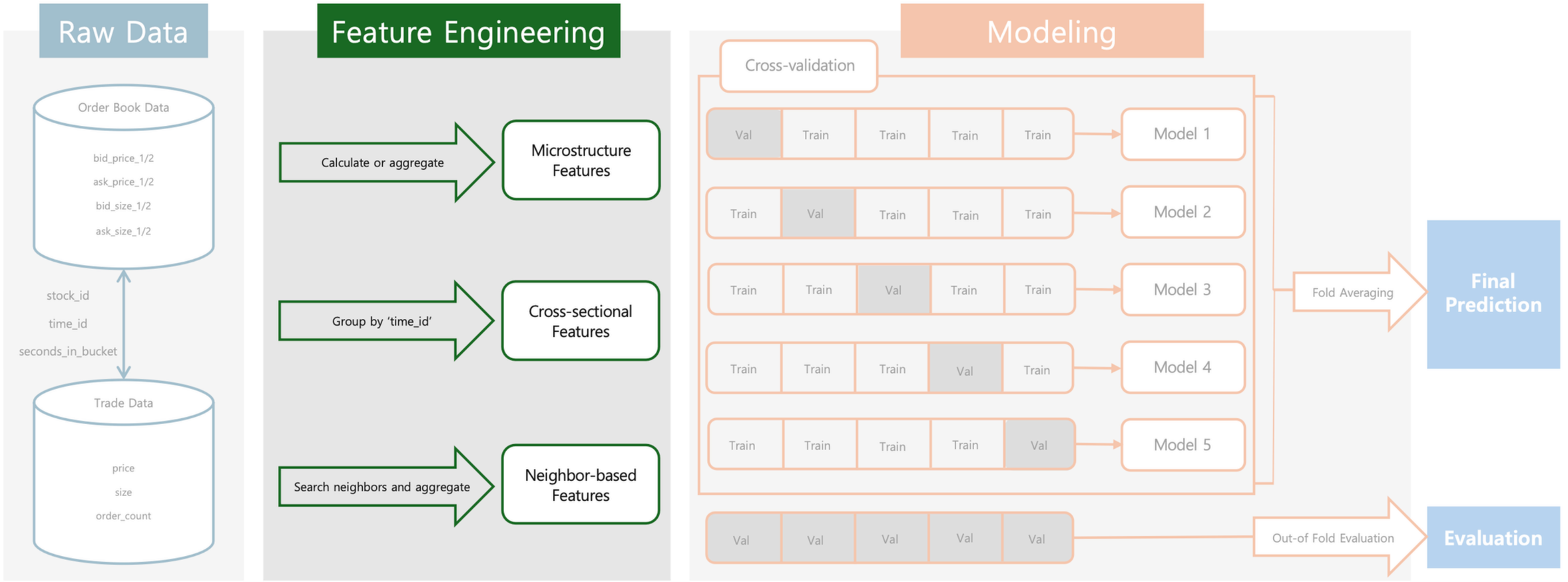
Table 4. Summary of features



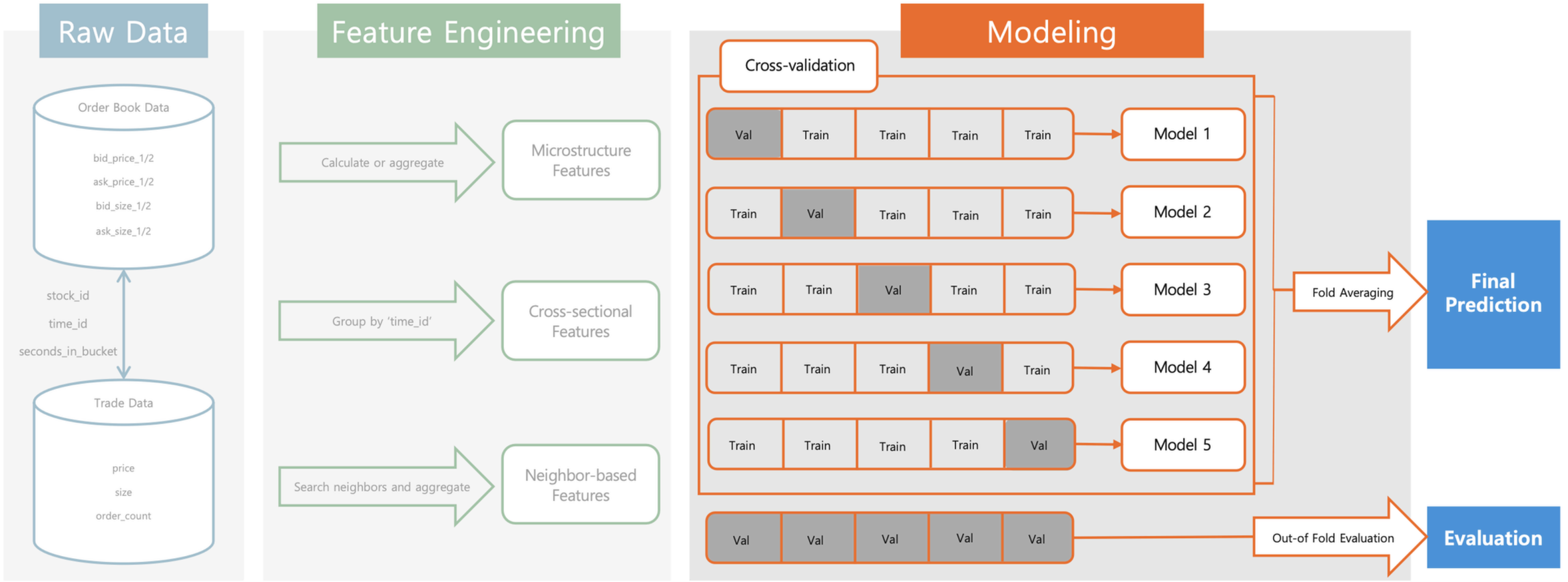
4

# Modeling

# Modeling Pipeline



# Modeling Pipeline



# LightGBM

- **Why was it chosen?**
  - Data mainly consists of highly engineered tabular features.
  - LightGBM is known to perform well for tabular data.
  - It also provides fast training and strong performance with relatively low computational cost.

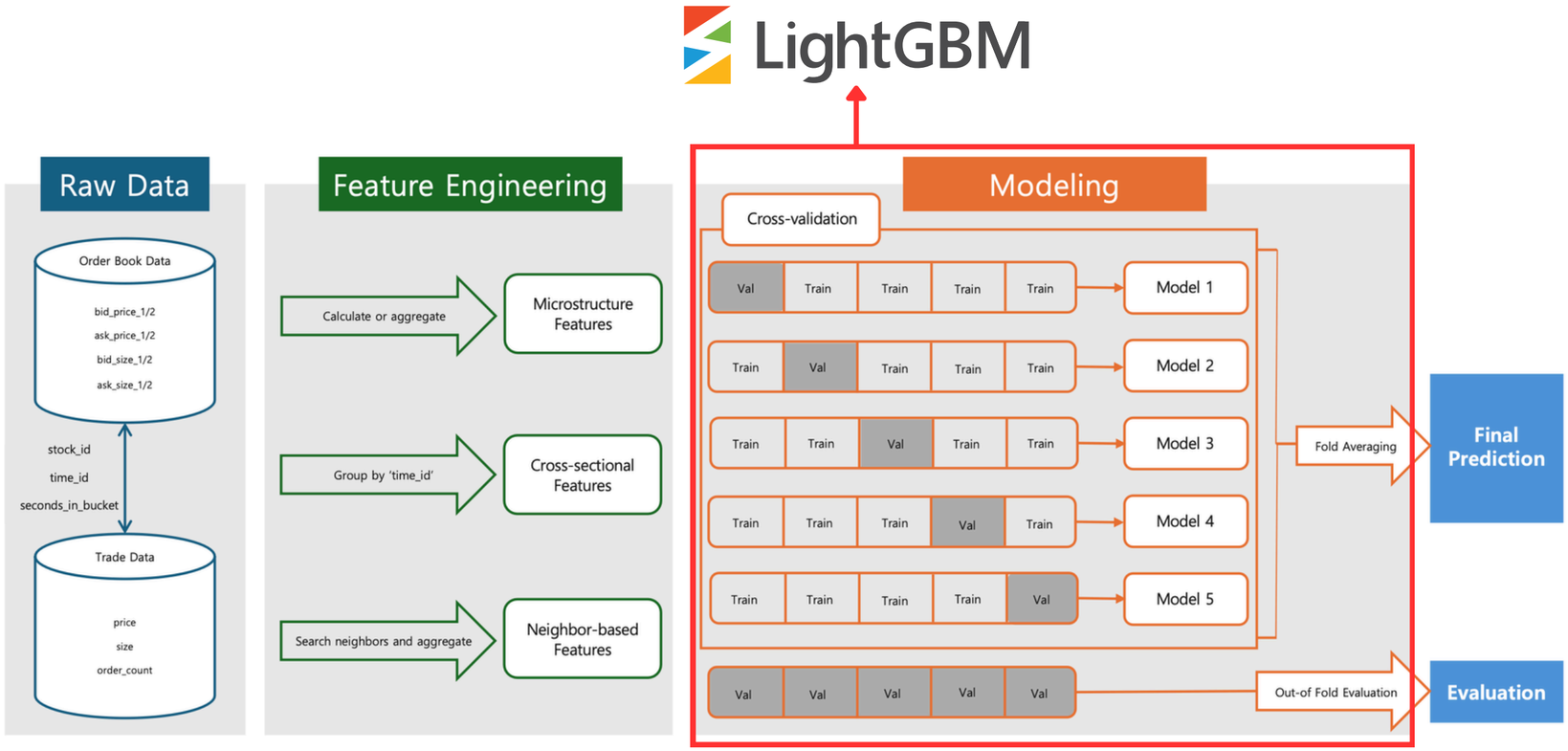



Figure 11. LightGBM training and validation pipeline

# Training & Validation Strategy

- **K-fold cross-validation**

- **Out-of-fold(OOF) evaluation**
  - In K-fold CV, each sample is used as a validation sample once. (i.e., predicted by a model that was not trained on that sample)
  - This produces more reliable validation estimates.
  - Since the hidden test set was unavailable, OOF evaluation was used for fair performance estimation.

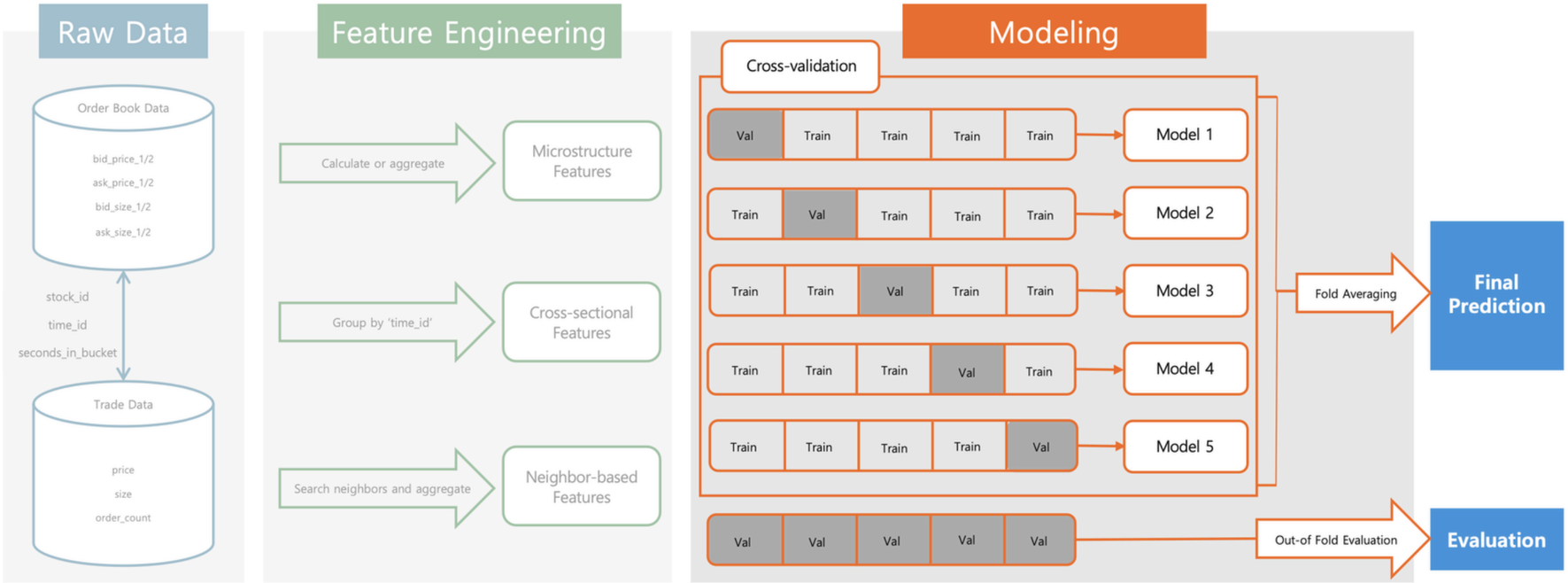



Figure 12. K-fold cross-validation, OOF evaluation and Fold Averaging

# Prediction Strategy

- **Fold averaging**
  - Final predictions are obtained by averaging predictions across all fold models.
  - This ensemble strategy reduces variance and improves prediction robustness.

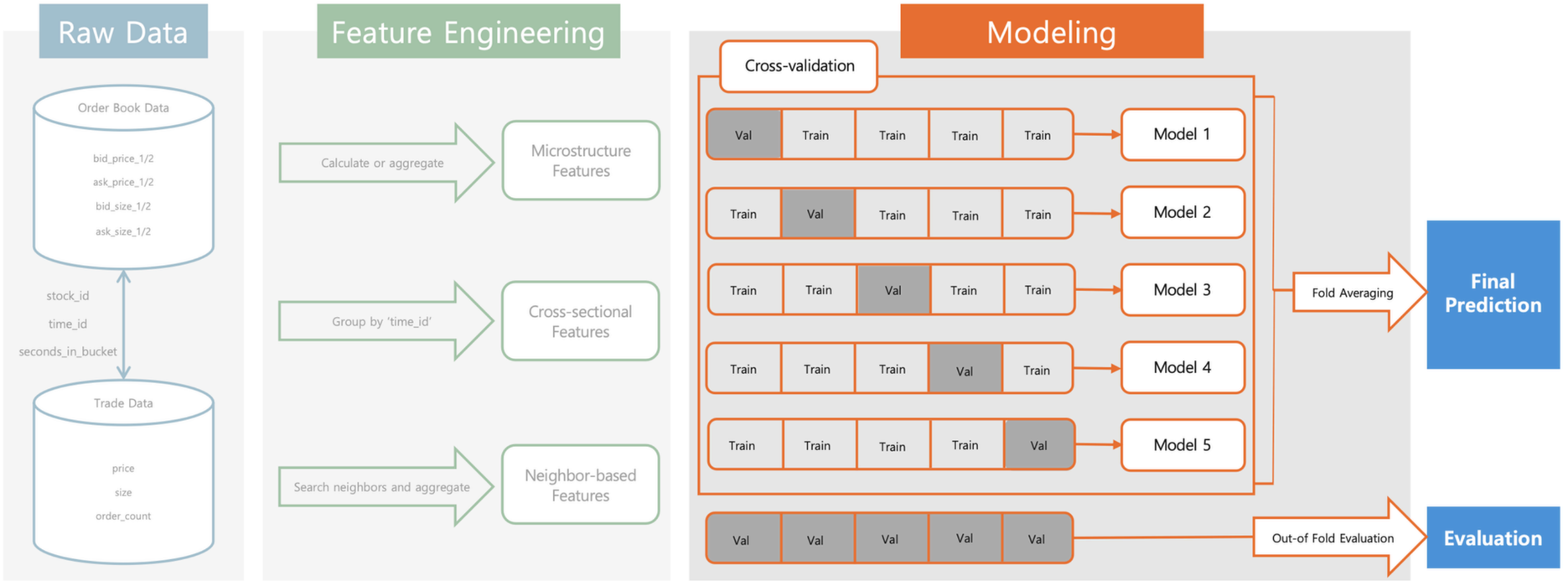



Figure 12. K-fold cross-validation, OOF evaluation and Fold Averaging

5

# Results

# Evaluation Metric : RMSPE

- **Definition of RMSPE**
  - Root Mean Squared Percentage Error
  - Commonly used when target variable has a wide range or a highly skewed distribution since relative error is more meaningful.

- **Why RMSPE?**
  - Target variables has a highly right-skewed distribution, and volatility scale substantially differ across stocks and time windows.
  - The same absolute value error can have very different impacts depending on the volatility level.
  - Therefore, relative percentage error is more meaningful than absolute error.

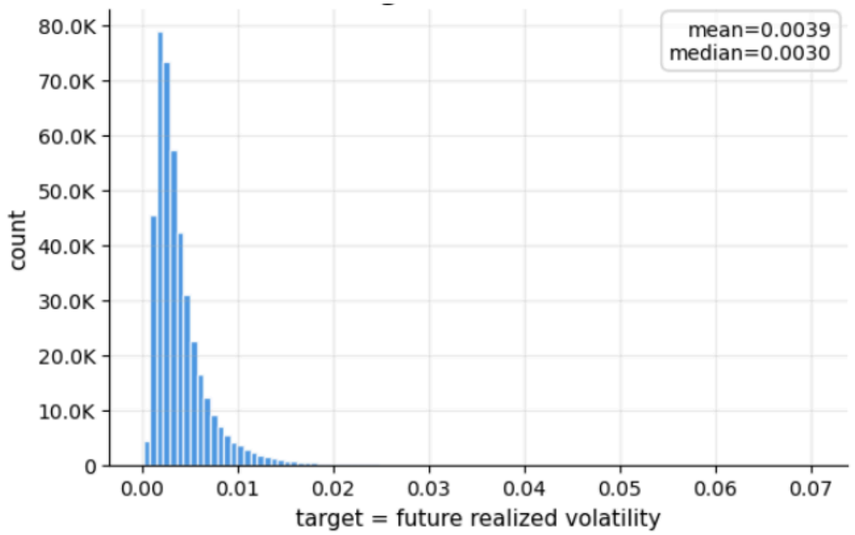



Figure 13. Distribution of the target variables

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

where

- $y_i$ : Actual realized volatility
- $\hat{y}_i$ : Predicted realized volatility
- $n$ : Number of observations
- $\frac{y_i - \hat{y}_i}{y_i}$ : Relative percentage error

Figure 14. Formula of RMSPE

# Model Performance

- **Performance of our team**
  - OOF RMSPE : 0.20144
  - Our OOF RMSPE was lower than the public score of the 9th-place solution

- **Additional strengths (To be explained further)**
  - Achieved with a computationally efficient model
  - Explicitly designed to avoid information leakage

| # | Team | Members | Score | Entries | Last | Solution |
|---|---|---|---|---|---|---|
| 1 | nyanp | | 0.19548 | 2 | 5y | |
| 2 | YoonSoo | | 0.19720 | 2 | 5y | |
| 3 | Eduardo Peynetti | | 0.19883 | 2 | 5y | |
| 4 | [PKSHA]life is volatile | | 0.19899 | 2 | 5y | |
| 5 | jiebao | | 0.19905 | 2 | 5y | |
| 6 | Light | | 0.19960 | 2 | 5y | |
| 7 | Michael Poluektov | | 0.20013 | 2 | 5y | |
| 8 | Hantian Zheng | | 0.20126 | 2 | 5y | |
| 9 | 冯博小迷弟&KT | | 0.20238 | 2 | 5y | |

Figure 15. Screenshot of Kaggle Leaderboard Page

# Feature Importance Analysis

- **Key observations(1)**
  - Realized volatility related features dominate the model.
  - Suggests that recent volatility is highly informative for predicting future volatility.

- **Key observations(2)**
  - Order-book based volatility features were especially important. (book_log_return1_realized_volatility_* ranked at the top)
  - Suggests that order book data provides richer short-term market signals than transaction data.

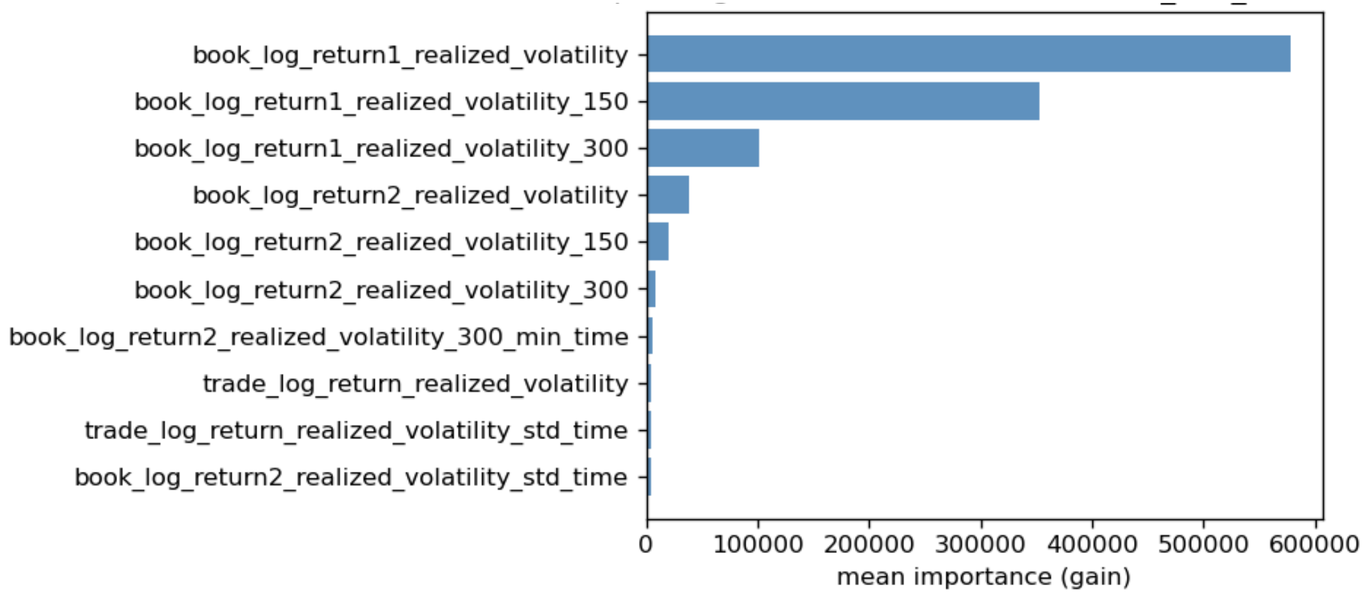



Figure 16. Top 10 feature importances from LightGBM

# Feature Importance Analysis

- **Key observations(1)**
  - Order book derived features accounted for most of the predictive power.
  - This was consistent with the earlier individual feature importance analysis

- **Key observations(2)**
  - Trade data derived and cross-sectional features provided complementary information.
  - Neighbor-based features had relatively small importance, but still improved OOF performance. (To be explained further)

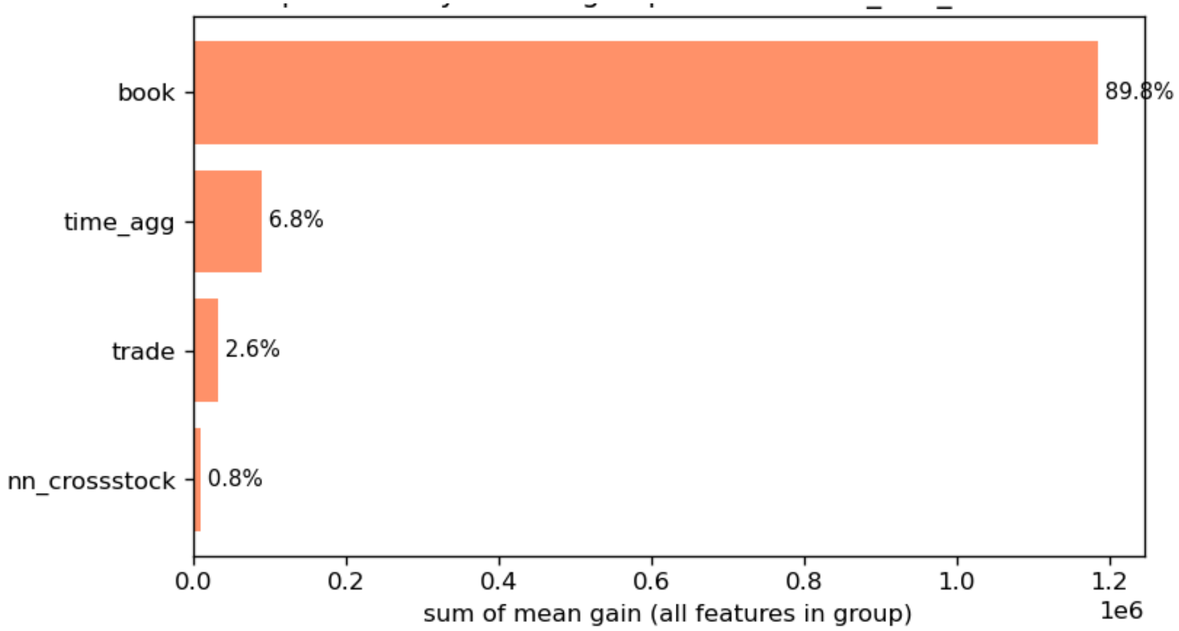



Figure 17. Importance by feature group from LightGBM

# Ablation Study

- **Key observations(1)**
  - Incorporating cross-sectional features significantly improved OOF RMSPE.
  - Suggests that volatility behavior is not purely stock-specific and market-wide information contains meaningful predictive signals.

- **Key observations(2)**
  - Neighbor-based features provided additional improvements despite their relatively small importance.
  - Indicates that it’s not main driver, but provided useful complementary signals when combined with cross-sectional information.

| Setting | Microstructure | Cross-sectional | Neighbor-based | Number of Features | OOF RMSPE |
|---|---|---|---|---|---|
| Baseline | O | | | 264 | 0.233666 |
| + Cross-sectional | O | O | | 312 | 0.201570 |
| + Neighbor-based | O | | O | 279 | 0.230766 |
| Our model | O | O | O | 327 | 0.201445 |

Table 5. Summary of ablation study

# Lessons from Top Kaggle Approaches

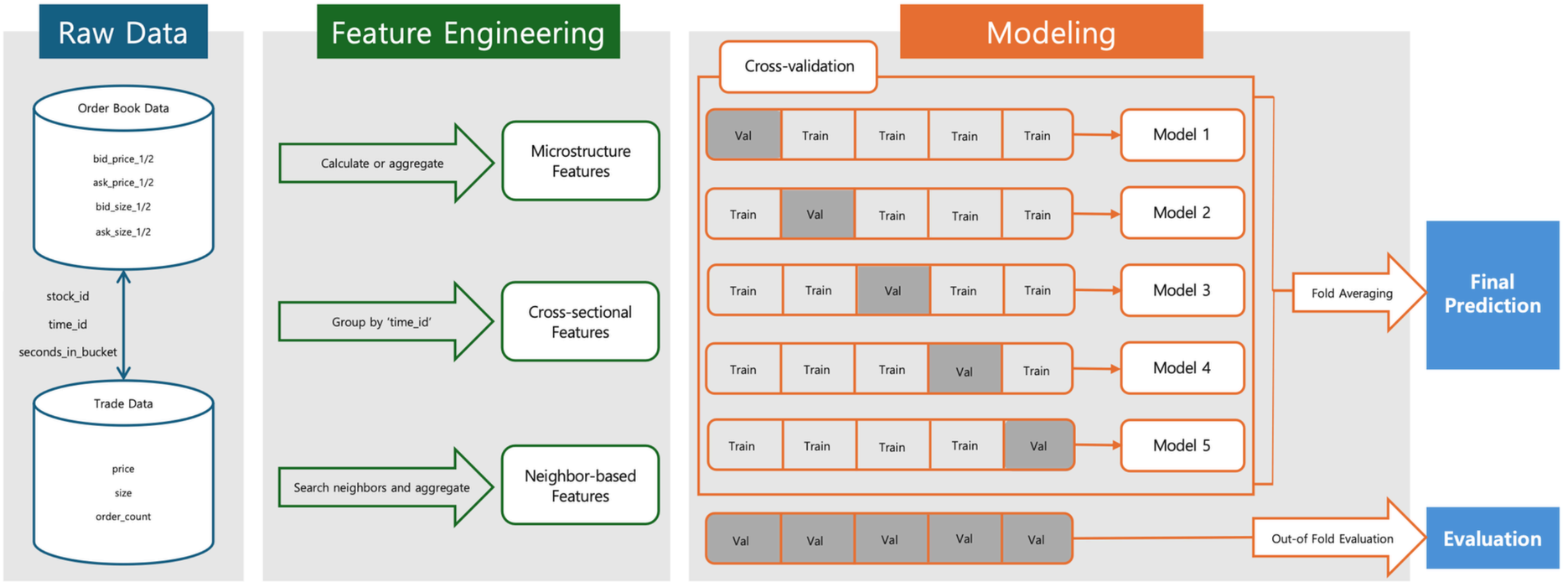
- Several top-ranked solutions attempted to recover hidden temporal structure from anonymized data.
- Many approaches required large ensemble architectures.
- Our approach prioritized leakage-aware and computationally efficient forecasting

| Rank | Main Strategy | Explanation | Limitation |
|---|---|---|---|
| 1st | Nearest Neighbor<br>Hidden Time Reconstruction | Recovered hidden order of time periods and used similar past market situations together | Recovering hidden order may accidentally use future market information |
| 3nd | Massive Ensemble<br>Market-state Features | Combined may different machine learning and deep learning models together | High computational cost |
| 7th | Graph-based Chronology Recovery | Connected similar market periods like a network to guess their hidden time order | Recovering hidden order may accidentally use future market information |

Table 6. Summary of top-ranked Kaggle approaches

# Our Contribution

- Designed a leakage-aware forecasting pipeline by explicitly avoiding hidden chronology reconstruction and future-like information usage.
- Achieved competitive OOF performance using computationally efficient LightGBM instead of massive deep ensembles.
- Demonstrated that carefully engineered market microstructure features remain highly effective for volatility forecasting.



# Thank you for listening!

# Any questions?