# Nasdaq Closing Cross Predictive Analytics

Haoran Dong
Data Science Initiative, Brown University
GitHub

## 1 Introduction

### 1.1 Purpose

The project aims to predict the closing price movements of numerous NASDAQ-listed stocks, using the Nasdaq Closing Cross auction data in the last 10 minutes. This prediction is crucial as it assists in adjusting stock prices, assessing supply and demand dynamics, and identifying key trading opportunities in the market. These closing price movements are essential indicators as they will help market participants to make decisions and strategies.

### 1.2 Data Source

The data for this project is sourced from Optiver [2], a prominent global electronic market maker. Optiver provides a comprehensive dataset for 200 stocks specifically during the NASDAQ stock exchange's critical 10-minute closing auction. This dataset is rich in detail, featuring key metrics such as far price, near price, reference price, and imbalance size of each stock, among others.

### 1.3 Target Variables and Features

The primary target variable in the dataset is a synthetic index constructed by Optiver. This dataset comprises approximately 500 million data points and 17 columns and it includes not only price-related features but also encompasses other crucial information like bid and ask prices, their respective sizes, matched sizes, and the weighted average price (WAP). These features collectively provide a comprehensive view of the market dynamics during the closing auction and are integral to developing an effective predictive model.

## 2 Exploratory Data Analysis

Each row in the stock dataset represents a unique combination of stock ID and trading time across different days, where multiple rows can belong to the same stockID. There are 5,237,980 observations and 16 columns in the dataset, with 10 columns considered features. There is 1 categorical feature and the remaining features are all continuous.

### 2.1 Time ID Analysis

The first plot shows the average target value fluctuating over a different trading seconds, suggesting a stable but volatile series.

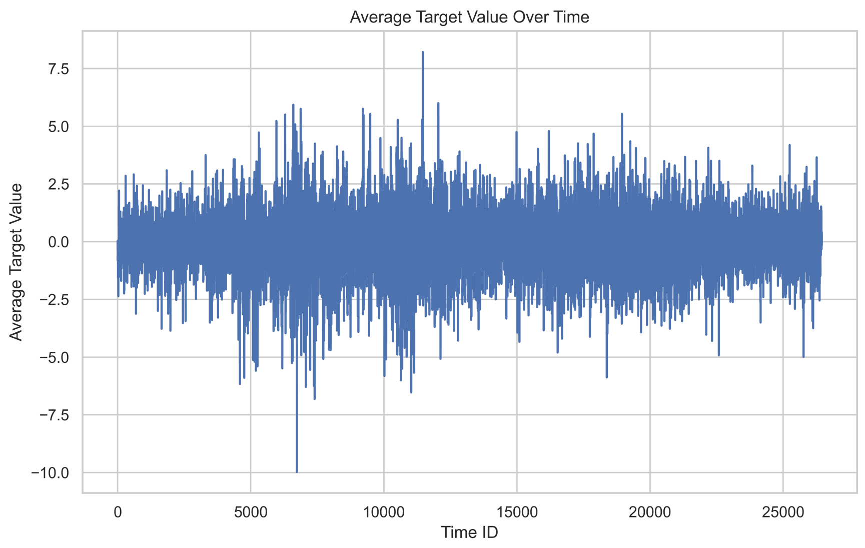

**Figure 1:** Average Target Value Over Time. The vertical axis represents the average target value; the horizontal axis represents different trading seconds.

## 2.2 Imbalance Analysis and Reference Price Relationship

The boxplot in Figure 2 suggests consistent central tendency and variability within each category. The scatter plot in Figure 3 shows a dense cluster of data around a reference price of 1.00, which indicates no clear linear relationship between reference price and target.

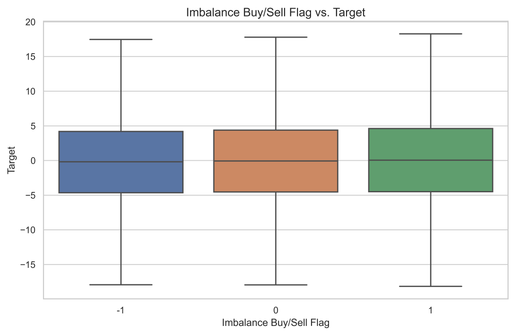

**Figure 2:** Imbalance Buy/Sell Flag vs. Target.

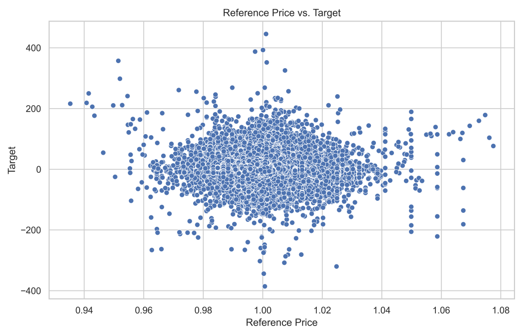

**Figure 3:** Reference Price vs. Target.

## 2.3 Feature Correlation Analysis

The heatmap shows correlationships between different features. From Figure 4, we can tell that some features show strong correlation between each other:bid price, ask price, wap and reference price; while other features show nearly no correlation between each other: far price, near price and imbalance sell flag.

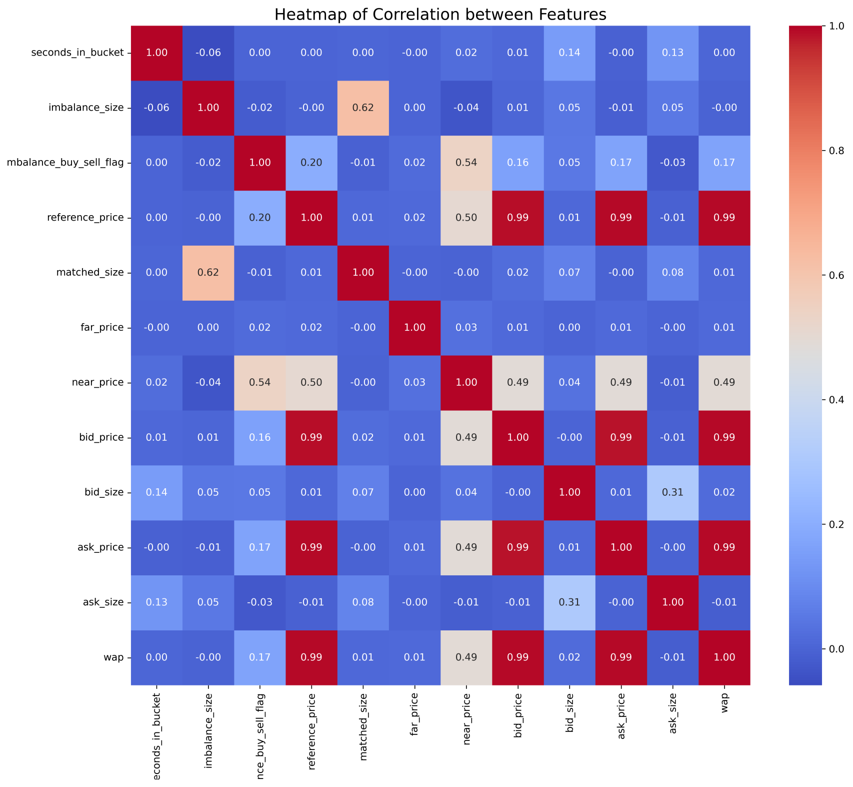

**Figure 4:** Heatmap of Correlation between Features. This visual representation highlights the correlations among different trading features, with color intensity representing the strength of the relationship.

## 3 Methods

### 3.1 Data Split

In the Optiver dataset, given that the data is chronologically provided with 'date_id', basic train_test_split does not work because it will lead to data leakage. Thus, I first split the data based on their 'date_id', which means that the first 478 dates are used as training data and the last two dates are used as test data. Different from IID datasets, we can't simply set 10 random states to get N RMSE scores for n loops because this method will lead to serious data leakage in time-series dataset. Thus, I decide to use expanding window method [1][3][4]. In this method, I start with a 150-day training window[5], expanding by 30 days for each loop. Then, I employ TimeSeriesSplit to do 5 times splits into the training window set for cross-validation, which is equivalent to basic

K-Fold on IID dataset. Finally, for 11 loops, 11 RMSE for each model will be calculated in the list, which will help us to select the best model.

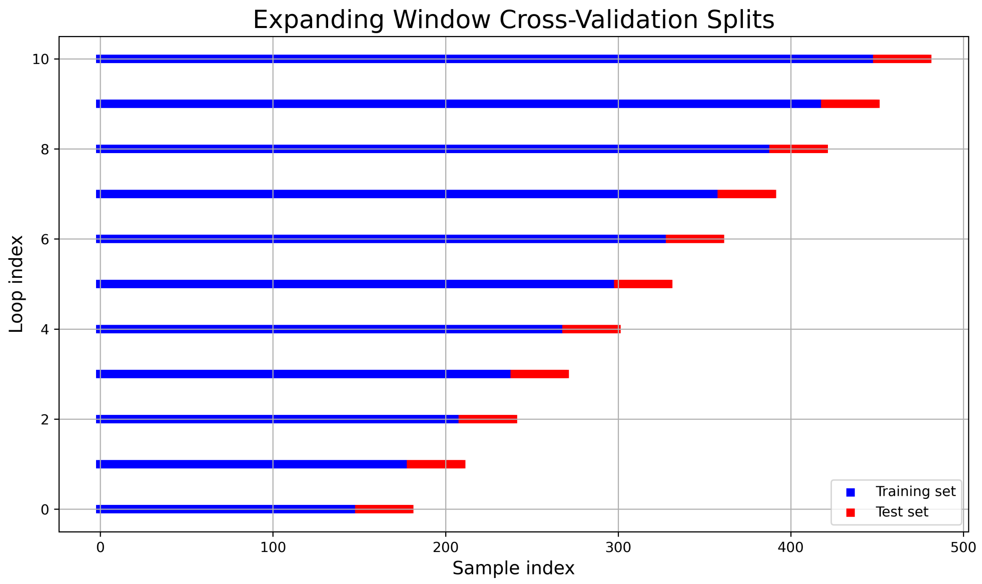

**Figure 5:** Blue represents the training data;Red represents the test data for each Loop

## 3.2 Feature Engineering

Based on some financial domain knowledge, I decide to add some new features into existing features: 2 imbalance metrics (imb_s1 and imb_s2) for trading pressure and net imbalance;8 additional features grounded in Financial Economics principles, enriching the dataset's analytical depth.

## 3.3 Data Preprocessing

The dataset's single categorical feature, 'imbalance_buy_sell_flag', will be processed using OneHotEncoder, while the remaining features which are continuous will be standardized using StandardScaler to ensure consistent scaling across the dataset.

## 3.4 Missing Values

There is only a small fraction of missing values in reference price, bid price and ask price. To address this, based on financial knowledge and expertise, I simply impute those missing values with values in the last trading seconds.

## 3.5 ML Pipeline

Five models(Linear Regression with penalty L1, XGBoost Regressor, Support Vector Regressor, Random Forest Regressor and Kneighbors Regressor) are implemented in the analysis with GridSearchCV for hyperparameter tuning with cross-validation.

**Table 1:** Hyperparameter tuning values for different ML models

| ML Model | Hyperparameter | Values |
|---|---|---|
| Linear Regression(L1) | alpha_Lasso | 0.001,0.015,0.025,0.035,0.045, 0.055,0.065,0.075,0.085,0.095 |
| Random Forest Regression | max_depth | 10,20,30 |
| | min_samples_split | 2,5,10 |
| Support Vector Regression | svr_gamma | 0.001,0.1,1,10,1000 |
| | svr_C | 0.1,1,10 |
| KNeighborsRegressor | n_neighbors | 1,5,10,50,100,500 |
| XGBRegressor | learning_rate | 0.01,0.1,0.2 |
| | max_depth | 3,8,20,40 |

The metric used to evaluate five models' performance is RMSE since RMSE is highly sensitive to large errors, which is crucial in this regression problem where large prediction errors can have significant consequences. In this time-series problem, uncertainties from splitting and non-deterministic models were addressed by performing Expanding window method. Models are fitted using GridSearchCV and their performance are assessed by calculating RMSE on the test set for each window.Finally, the best model along with its validation and test scores are recorded in a list.

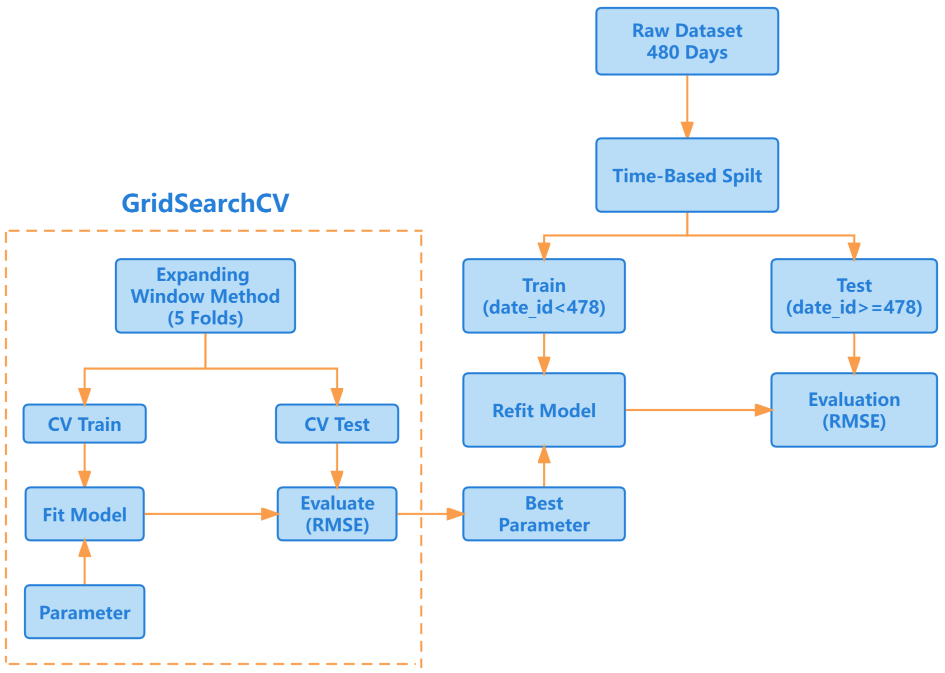

**Figure 6:** Flow chart of the model development and cross validation pipeline.

## 4 Results

The baseline RMSE is 9.64, which is calculated by filling the predicted value with mean value of the target variable. The results of different models's performance are shown below.

| Model | Mean Test Score | Standard Deviation |
|---|---|---|
| XGBoost Regressor | 7.1301 | 0.6204 |
| Random Forest Regressor | 7.5497 | 0.6216 |
| Support Vector Regressor | 7.3988 | 0.6265 |
| Kneighbors Regressor | 7.4246 | 0.6449 |
| Linear Regression (L1) | 7.7909 | 1.5640 |

**Table 2:** Comparison of regression models

From the figure 7, we can tell that all models showed relatively low RMSE, and out of the models, Linear Regression with Penalty L1 showed the highest RMSE and standard deviation while XGBoost Regressor showed the best prediction based on its low RMSE and standard deviation.

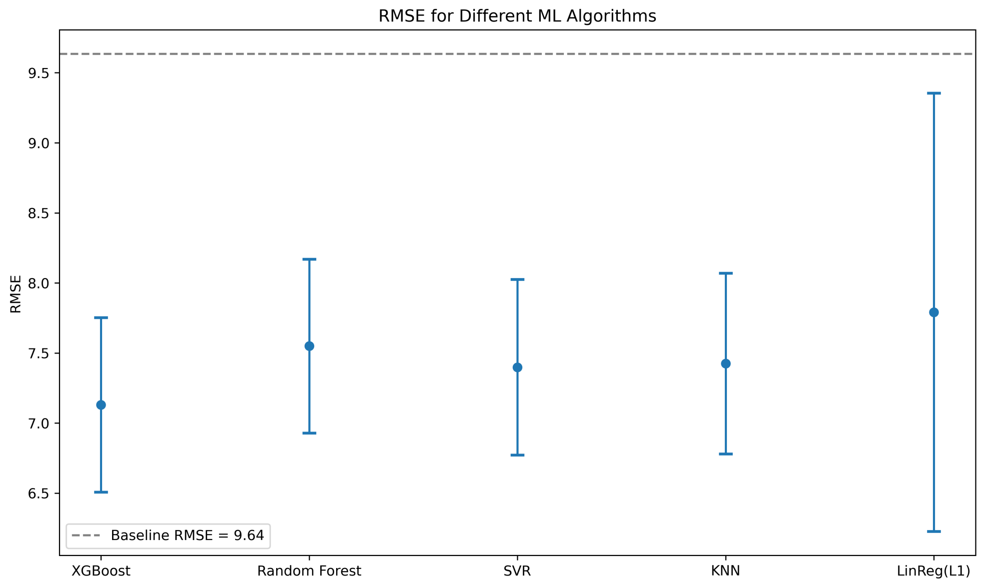

**Figure 7:** Comparison of RMSE values for different machine learning algorithms with a baseline for reference.

### 4.1 Global Importance

In order to to study which features are most important in the best performing model, different metrics were explored. 3 types of importance metrics for the best XGBoost model are:

#### 4.1.1 Permutation feature importance

From the figure, we can tell that std_imbalance_size, std_bid_price and std_matched_size are top 3 features as per permutation importance.

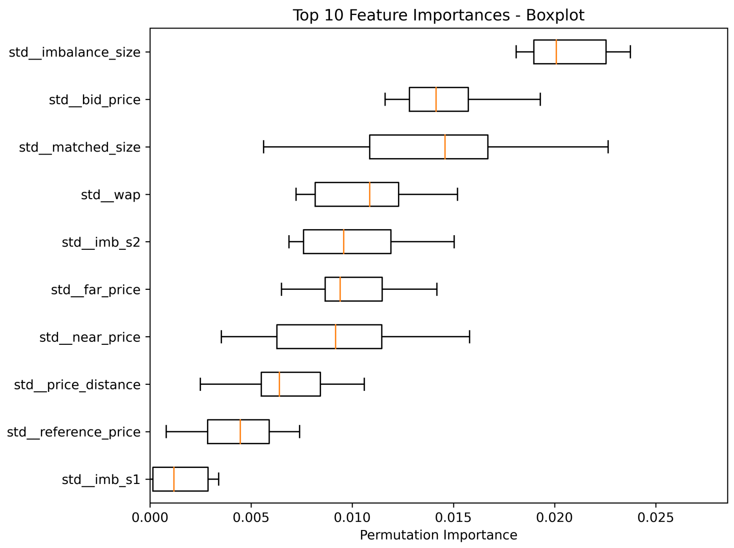

**Figure 8:** Boxplot of the top 10 feature importances illustrating the distribution of permutation importance.

### 4.1.2 Feature importance by weight

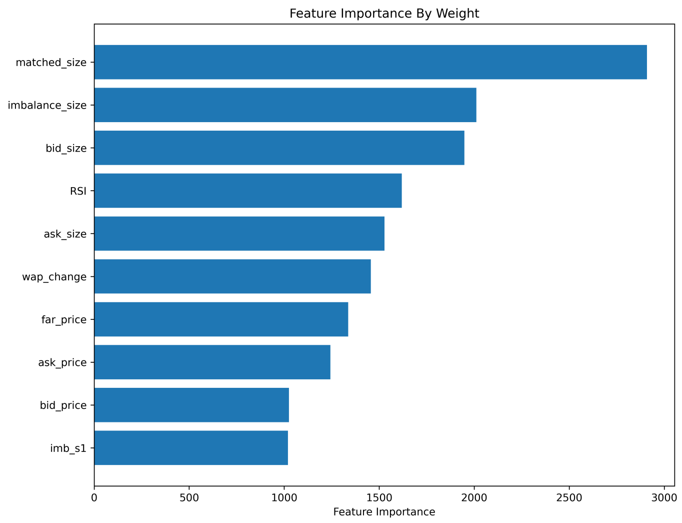

**Figure 9:** Top 10 most important features as per XGBoost relative importance measure- weight

From the figure, we can tell that matched_size, imbalance_size and bid_size are the most important for prediction as per XGBoost's relative importance measure- Weight.

### 4.1.3 Feature importance by total_gain

From the figure, we can tell that matched_size, ask_size and RSI are the most important for prediction as per XGBoost's relative importance measure- total_gain.

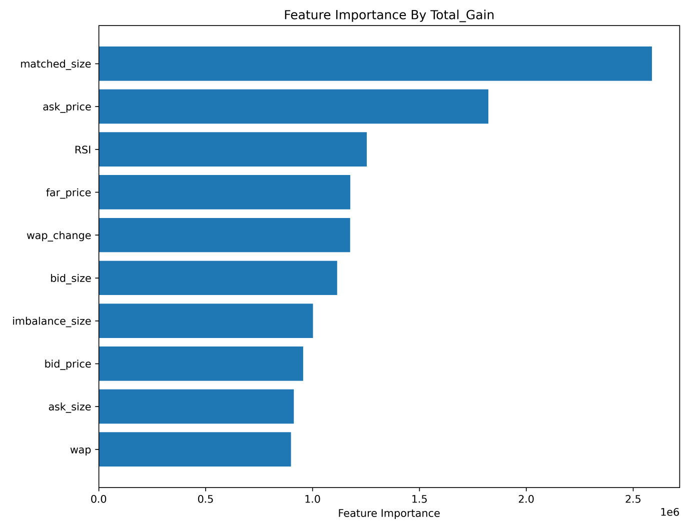

**Figure 10:** Top 10 most important features as per XGBoost relative importance measure-total_gain.

#### 4.1.4 SHAP Global Feature Importance

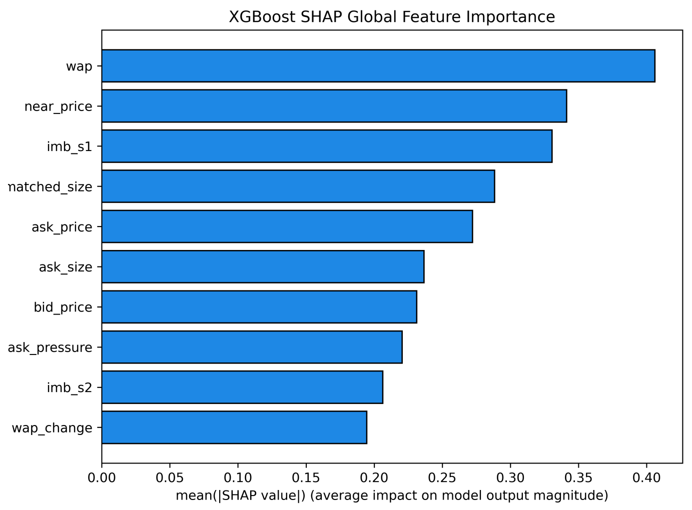

**Figure 11:** Top 10 most important features as per mean Shap value

Finally, from figure, we can tell that wap, and imb_s1 are the most important features as per mean Shap values.

### 4.2 Local Importance

Local feature importance was used to see which features influence the prediction for a single observation.A SHAP force plot is shown in Figure plots for different data points is generated for one observation.

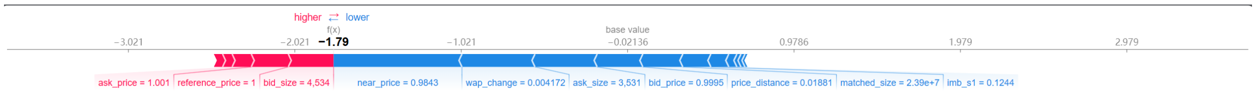

**Figure 12:** Local feature importance shap values for point 0th

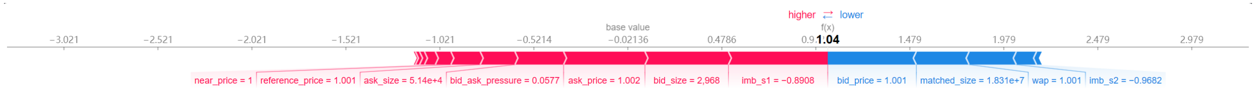

**Figure 13:** Local feature importance shap values for point 3684th

In the SHAP Force plot for data point with index 0, the features such as near_pricewap_change and ask_price contribute negatively to the prediction, pushing the output value lower than the base value.While in the data point with index 3684, features such as near_price, reference_price and ask_size contribute positively to the prediction, pushing the output higher than the base value.

## 5 Outlook

Given additional time, there are several key improvements that could be made to refine the model:

- **Hyperparameter Optimization:** Because of the computational demands of XGBoost and large volume of my dataset, current hyperparameter tuning is limited to learning rate and max_depth. Given more time, I will Expand the hyperparameter tuning scope to include parameters such as subsample, colsample_bytree, and min_child_weight that could potentially enhance the model's performance.
- **Model Experimentation:** Except for models that we covered in class, I will explore different models like Light GBM, LSTM, or a Reduced Features Model.
- **Missing Value Strategy:** Replacing missing values with the last observed values might be simplistic. I will try more sophisticated imputation techniques, such as KNN imputation or multiple imputations, which may yield better results.
- **Feature Engineering:** I will also develop more features with more financial relevance and delete some highly correlated features to improve predictive power of my model.
- **Data Collection:** The dataset currently has a 60-second interval between observations. If it is possible, I will add data in every second to fill the gap between 60-second interval.

## 6 GitHub Repository

The source code and related resources for this project can be found in the following GitHub repository: github.com/QIQIZHANG852/final$_{r}eport$.

## References

[1] Herman-Saffar, Or. Time Based Cross Validation. 2021, `https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8`.

[2] Optiver-Trading at the Close. `https://www.kaggle.com/competitions/optiver-trading-at-the-close/data`.

[3] Raschka, Sebastian. GroupTimeSeriesSplit: A scikit-learn compatible version of the time series validation with groups. 2019, `https://rasbt.github.io/mlxtend/user_guide/evaluate/GroupTimeSeriesSplit/`.

[4] TimeSeriesSplit-sklearn. `https://sklearn.vercel.app/docs/classes/TimeSeriesSplit`.

[5] Egor, Howell. How To Correctly Perform Cross-Validation For Time Series. *Towards Data Science*, 2023. Available at: `https://towardsdatascience.com/how-to-correctly-perform-cross-validation-for-time-series-b083b869e42c`.