

# Winning Solution of Kaggle Algorithmic Trading Challenge

Jing Guo

Strats Associate, Goldman Sachs

PhD in Financial Engineering, Columbia University

January 11, 2019

# Outline

- 1 Data Introduction
- 2 Time Interval Partitioning
- 3 Feature Construction & Selection
- 4 Prediction & Validation

- Kaggle Algorithm Trading Challenge was organized by the Capital Markets Cooperative Research Center (CMCRC - [www.cmcrc.com](http://www.cmcrc.com)) in Macquarie University, Sydney, Australia.
- The timeline is from Nov 11, 2011 to Jan 8, 2012.
- Winning prize is cash \$8,000 and consideration for entry to the CMCRC PhD program.
- The winning team goes to Ildefons Magrans de Abril and Masashi Sugiyama de Abril and Sugiyama (2013), two engineer researchers in Tokyo Institute of Technology, Japan.

- The competition was to predict the short term response following large liquidity shocks.
- A liquidity shock is defined as any trade that changes the best bid or ask price.
- Liquidity shocks occur when a large (series of small) trade consumes all available volume of best offer.
- This kind of model can be used to optimize execution strategies of large transactions.
- Following a liquidity shock the spread may be temporarily widened, and/or result in permanent price shifts.

# Training & Testing Dataset

- The training dataset consists of 754,018 samples of trade and quote data observations before and after a liquidity shock.
- There are 102 different securities of the London Stock Exchange (LSE) included.
- The liquidity shock takes place at time interval 51, i.e., time interval 1-50 are pre-liquidity and time interval 52-100 are post liquidity.
- The test dataset consists of 50000 samples similar to the training dataset but without the post-liquidity shock observations.

# Training & Testing Dataset (Cont'd)

- Further variables
  - security id (*security\_id*)
  - indicator of buyer or seller (*initiator*)
  - the volume-weighted average price causing the liquidity shock (*trade\_vwap*)
  - the total size of the trade causing the liquidity shock (*trade\_volume*)
- The test dataset consists of 50,000 samples similar to the training dataset but without post-liquidity-shock observations (i.e. time interval 51-100).
- The goodness of fit is measured by root-mean-square error (RMSE).

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

# Outline

- 1 Data Introduction
- 2 Time Interval Partitioning
- 3 Feature Construction & Selection
- 4 Prediction & Validation

# Time Interval Partitioning Algorithm

- Recall that we are to predict post-shock observations in time interval 51-100.
- Intuitively, prediction power/signal decays as time goes forward.
- Therefore we should not use the same model for each time interval. However, we may risk overfitting if we specify one model for each interval.
- We are to divide the 50 time intervals into several subgroups within each of which one model is applied.



# Time Interval Partitioning Algorithm (Cont'd)

- Greedy algorithm is applied for time interval partitioning.

---

**Algorithm 1** Time interval partitioning algorithm

---

```
1:  $b \leftarrow e \leftarrow 52$ ;  $P \leftarrow \text{NULL}$ ;  $i \leftarrow 1$ 
2: while  $b < 100$  do
3:    $C_i \leftarrow \text{NULL}$ ;  $e \leftarrow b + \text{length}(C_i)$ ;  $\text{bestError} \leftarrow \infty$ 
4:   repeat
5:      $C_i \leftarrow \text{createTimeInterval}(b, e)$ 
6:      $C_{all} \leftarrow \text{createTimeInterval}(e+1, 100)$ 
7:      $\text{error} \leftarrow \text{evaluateModel}(P, C_i, C_{all})$ 
8:     if  $\text{bestError} > \text{error}$  then
9:        $\text{bestError} \leftarrow \text{error}$ 
10:    end if
11:     $e \leftarrow e + 1$ 
12:  until  $\text{bestError} \neq \text{error}$ 
13:   $\text{addTimeInterval}(P, C_i)$ 
14:   $i \leftarrow i + 1$ ;  $b \leftarrow e$ 
15: end while
```

---

# Time Interval Partitioning Algorithm (Cont'd)

- Time interval 52-100 are divided into  $K$  sub-groups. And thus  $K$  sub-models are fitted.
- Let  $C_i \subseteq \{52, 53, \dots, 100\}$  be the  $i$ 'th subgroup. Then the prediction model is:

$$M_{\text{bid}(t)} = \sum_{i=1}^K \delta_{t,C_i} M_{\text{bid},i}(t),$$
$$M_{\text{ask}(t)} = \sum_{i=1}^K \delta_{t,C_i} M_{\text{ask},i}(t).$$

where  $t \in \{52, 53, \dots, 100\}$ , and  $M_{\text{bid}(t)}/M_{\text{ask}(t)}$  is the sub-model to be fitted, and

$$\delta_{t,C_i} = \begin{cases} 1 & \text{if } t \in C_i, \\ 0 & \text{otherwise.} \end{cases}$$

# Outline

- 1 Data Introduction
- 2 Time Interval Partitioning
- 3 Feature Construction & Selection**
- 4 Prediction & Validation

- *Price Features*

- exponential moving average (EMA) of the last  $n$  bid/ask prices
- the price rises/drops of the last  $n$  bid/ask prices

- *Liquidity Features*

- The *depth* of the order book

- *Spread Features*

- exponential moving average (EMA) of the last  $n$  bid-ask spreads

- *Arrival Rate Features*

- the number of quotes/trades during the last  $n$  time intervals.

# Moving Average

- *Moving Average* is a calculation to analyze data points by creating a series of averages of different subsets of the full data set.
- *Simple Moving Average*

- 

$$\bar{p}_{SMA} = \frac{p_M + p_{M-1} + \dots + p_{M-(n-1)}}{n}$$

- Examples: 5-day moving average, 10-day moving average, etc
- *Cumulative Moving Average (CMA)*

$$\bar{p}_{CMA}(n+1) = \frac{p_1 + \dots + p_n}{n} = \frac{1}{n+1}p_{n+1} + \frac{n}{n+1}\bar{p}_{CMA}(n)$$

- *Exponential Moving Average (EMA or EWMA)*

$$\bar{p}_{EMA}(n+1) = \begin{cases} p_1, & n = 0, \\ \alpha \cdot p_{n+1} + (1 - \alpha) \cdot \bar{p}_{EMA}(n), & n > 0. \end{cases}$$

# Feature Selection

- Feature selection is based on the *feature importance* ranking in Random Forest.
- A selection algorithm similar to the *backward-forward* algorithm in linear regression is implemented.

---

**Algorithm 2** Feature selection algorithm

---

- 1: Train a single piece model using all  $S$  features
  - 2: Compute model performance against the test set
  - 3: Rank features importance (RF importance method)
  - 4: **for** each subset size  $S_i = S, S - 1, \dots, 1$  : **do**
  - 5:     Retrain the model with only  $S_i$  most important features
  - 6:     Re-compute individual variable importance and re-rank
  - 7:     Fit the model to  $S_f$  features and rank individual features
  - 8: **end for**
  - 9: Determine which  $S_i$  yielded the smallest RMSE. Call this  $S_f$
  - 10: **repeat**
  - 11:     Choose a set of semantically similar features from  $S_f$
  - 12:     Select the feature with less rank not selected before
  - 13:     Evaluate the model performance
  - 14:     If smaller RMSE, then remove the feature
  - 15: **until** no improvement
  - 16: **repeat**
  - 17:     Choose a feature set among the already removed in Steps 1)-9) considering only those semantically orthogonal with the already selected in Steps 1)-15)
  - 18:     If smaller RMSE, then we add the feature to  $S_f$
  - 19: **until** no improvement
- 



# Outline

- 1 Data Introduction
- 2 Time Interval Partitioning
- 3 Feature Construction & Selection
- 4 Prediction & Validation

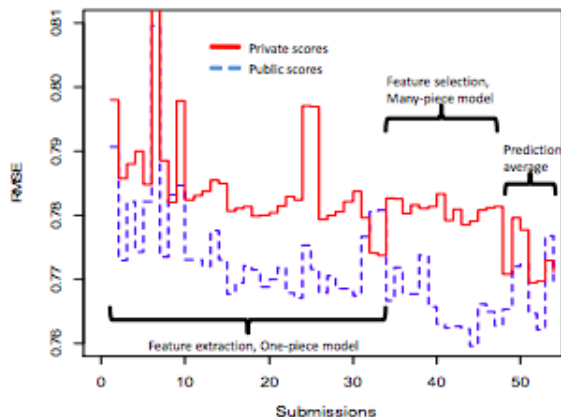
# Model Approach

- Gradient Boosting Machines and Random Forest are implemented separately for this project (Back then Xgboost was not developed yet).
- A 4-fold cross validation with a 75%-25% proportion for training and testing respectively is used.
- The optimized Gradient Boosting Machine and Random Forest models delivered respectively and average cross-validated RMSE of 1.163 and 1.156.



# Validation

- See the evolution of public and private scores in the following plot:



**Figure:** Evolution of public (dashed line) and private (solid line) scores. Feature construction, selection, partitioning and final prediction average were performed sequentially as indicated in the plot.

de Abril, I. M. and Sugiyama, M. (2013). Winning the kaggle algorithmic trading challenge with the composition of many models and feature engineering, *IEICE Transactions on Information and Systems* (3): 742–745.