IEICE TRANS. INF. & SYST., VOL.E96–D, NO.3 MARCH 2013

LETTER

Winning the Kaggle Algorithmic Trading Challenge with the Composition of Many Models and Feature Engineering

Ildefons MAGRANS DE ABRIL[†a)] ***and*** **Masashi SUGIYAMA**[†b)], ***Members***

SUMMARY This letter presents the ideas and methods of the winning solution[*] for the Kaggle Algorithmic Trading Challenge. This analysis challenge took place between 11th November 2011 and 8th January 2012, and 264 competitors submitted solutions. The objective of this competition was to develop empirical predictive models to explain stock market prices following a liquidity shock. The winning system builds upon the optimal composition of several models and a feature extraction and selection strategy. We used Random Forest as a modeling technique to train all sub-models as a function of an optimal feature set. The modeling approach can cope with highly complex data having low Maximal Information Coefficients between the dependent variable and the feature set and provides a feature ranking metric which we used in our feature selection algorithm.

key words: *Kaggle challenge, model architecture, boosting, feature selection, high frequency trading, liquidity shock, maximal information coefficient*

1. Introduction

The goal of the Kaggle Algorithmic Trading Challenge was to encourage the development of empirical models to predict the short term response of Order-Driven Markets (ODM) following large liquidity shocks [1]. A liquidity shock is defined as any trade that changes the best bid or ask price. Liquidity shocks occur when a large trade (or series of smaller trades) consumes all available volume at the best price.

This letter presents an empirical model meant to predict the short-term response of the top of the bid and ask books following a liquidity shock. This kind of model can be used as a core component of a simulation tool to optimize execution strategies of large transactions. Compared to existing finance research models [2], [3], we were not interested in understanding the underlying processes responsible for the price dynamics. On the other hand, by chasing the optimal predictor we may have uncovered interesting insights that could be a source of research inspiration.

The challenge data consists of training and test datasets. The training dataset is meant to fit a predictive model and contestants are asked to submit predictions based on the test dataset using this model. The training dataset consists of 754018 samples of trade and quote data observations before and after a liquidity shock for several different securities of the London Stock Exchange (LSE). Changes to the state of the order book occur in the form of trades and quotes. A quote event occurs whenever the best bid or the ask price is updated. A trade event takes place when shares are bought or sold.

The test dataset consists of 50000 samples similar to the training dataset but without the post-liquidity shock observations (i.e., time interval 51–100). Due to a data bug, the quotes at time 51 and 50 were the same. Therefore, the final objective was to predict the post-liquidity shock observations in the interval 52–100. In addition to the bid and ask price time series, each training and test sample contains some few variables to distinguish the particular security (*security_id*), to indicate whether the trade has been initiated by a buyer or a seller (*initiator*), the volume-weighted average price of the trade causing the liquidity shock (*trade_vwap*) and the total size of the trade causing the liquidity shock (*trade_volume*) [1].

2. Model

The search for an optimal model was guided by one hypothesis and an additional self-imposed constraint:

Hypothesis: The predictive potential closer to the liquidity shock should be higher and it should degrade with the distance. The rationale of this hypothesis is that future events will depend also on post-liquidity shock events that still need to be predicted. Therefore, the prediction error will tend to increase with the distance from the liquidity shock.

Constraint: Feature extraction should generate semantically meaningful features. This self-imposed constraint was motivated by one of the authors' will to generate a predictive model with the highest possible explanatory capacity.

In the following sections we will show how these two points were strong potentials that helped to reach a good solution and finally to win the competition.

2.1 Architecture

The model architecture consists of separate models for bid and ask. Bid and ask models are each further divided into K sub-models responsible for predicting a constant price at specific future time intervals between 52 and 100. The set P consists of K disjoint time intervals and its union is the full

Manuscript received September 13, 2012.
Manuscript revised November 7, 2012.

[†]The authors are with the Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
a) E-mail: ildefons@sg.cs.titech.ac.jp
b) E-mail: sugi@cs.titech.ac.jp

DOI: 10.1587/transinf.E96.D.742

[*]Solution designed and implemented by Ildefons Magrans de Abril.

Algorithm 1 Time interval partitioning algorithm

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

interval 52–100:

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

$$\text{where } a_{i,t} = \begin{cases} 1 \text{ if t} \in C_i, \\ 0 \text{ otherwise,} \end{cases} \qquad t \in [52, 100].$$

C_i represents the i-th K post-liquidity shock time interval and $M_{\text{bid/ask},i}(t)$ is the sub-model responsible for predicting the constant price on the interval C_i.

Dividing the future time interval into the set of intervals P should avoid mixing the low signal-to-noise-ratio of "far-away" prices with the more predictable prices close to the liquidity shock. An additional model feature is that time intervals should have an increasing length (i.e., length(C_{i+1})≥length(C_i)). This feature is a consequence of the main hypothesis because an always increasing prediction error may require averaging longer price time series to obtain a constant price prediction with an acceptable error.

Algorithm 1 is responsible for dividing the future time interval 52–100 into disjoint and consecutive sub-intervals of increasing length (line 3). It is implemented as a greedy algorithm and it is able to partition the post-liquidity shock time interval with $O(n)$ time complexity.

2.2 Feature Engineering

Our feature engineering strategy is, together with the model architecture, one of the relevant contributions of this letter and an important piece of our success. The following two sections describe in detail the feature extraction and selection methods. However, a key component of the feature selection method, the feature selection algorithm, will be presented later in Sect. 2.3 because it has a strong dependency on the modeling approach that we have chosen.

2.2.1 Feature Extraction

The semantically meaningfulness constraint discussed in the beginning of Sect. 2 encouraged the development of more than 150 features (predictors) with well-known characteristics:

Price: Price features provide information about the bid/ask normalized price time series (price values divided by the post liquidity shock price). Technical analysis [4] and statistical estimators are the fundamental instruments to compute these predictors (e.g., the detrended price oscillator, the exponential moving average of the last n [bid/ask] prices before the liquidity shock, the number of price increments during the last n [bid/ask] prices before the liquidity shock).

Liquidity book: This class contains all features able to provide information about the depth of the liquidity book (e.g., liquidity book improvements in the last n time periods understood as bid/ask price increases between two consecutive quotes).

Spread: Spread related features are meant to distill information about the bid/ask spread. As price features, technical analysis and statistical estimators are the fundamental instruments to compute these predictors. Before computing the predictor, spread time series were divided by the minimum price increment allowed for the particular *security_id* (e.g., exponential moving average of the last n spreads before the liquidity shock).

Rate: This class of features provides information about the arrival rate of orders and/or quotes (e.g., number of quotes and/or trades during the last n events).

2.2.2 Feature Selection

Not all features have the same predictive potential. Moreover, just choosing a set of features which are most related to the predicted variable could generate a highly collinear feature set which could be very confusing for our model fitting algorithm. Overfitting could also be a consequence of having too many features as we may tend to describe random errors. In addition, as discussed in Sect. 2.1, we have to fit bid and ask models composed in turn by several sub-models. An exhaustive search of the optimum feature set for all bid and ask sub-models is not feasible. This section discusses the feature selection strategy. It is meant to identify a suitable minimal subset of features with low mutual interdependencies and we want to achieve that goal with affordable computing resources in an acceptable amount of time. Having those goals and constraints, we have defined a feature selection method which consists of the following components:

Relaxation of the feature selection optimization problem: A highly time consuming feature selection strategy would consist in choosing an optimal feature sub-set for each of the many sub-models of the architecture described in Sect. 2.1. We simplified this step by choosing only two feature sub-sets: A sub-set common to all sub-models that describe the future bid price (F_b) and a second feature sub-set common to all sub-models that describe future ask price (F_a).

Feature selection algorithm: It is an algorithm to choose the suitable feature sets (i.e., F_b and F_a). The details of this algorithm will be presented in the following section together with our modeling approach.

2.3 Modeling Approach

An initial analysis of the mutual information between features and the dependent variable revealed a very low mutual information and non-functional relationships according to the Maximal Information Coefficient (MIC [5]): More than 90% of features had mutual information and functional metric values below 10% of the maximum possible. This initial analysis suggested us the need for a modeling approach able to cope with the complex non-linear non-functional nature of the challenge data.

Gradient Boosting Machines [7] and Random Forest [6] have been successfully used in other challenges of similar complexity [8]. A performance comparison between these two approaches was carried out. For this exercise, our modeling dataset consisted of 50000 samples randomly sampled from the training dataset with the same *security_id* proportions than the test dataset. All features were computed for each sample. We used a 4-fold cross validation with a 75%–25% proportion for training and testing respectively. Each model performance was measured by separately computing the root mean squared error (RMSE) for the bid and ask at each time step following a liquidity shock.

The optimized Gradient Boosting Machine and Random Forest models delivered respectively an average cross-validated RMSE of 1.163 and 1.156. This initial result suggested us that Random Forest could be more appropriate for this particular problem. Besides, the R implementation of the Random Forest [6] has the embedded capability to measure the importance of each feature using out-of-bag observations. This was an additional motivation to select this technique as a modeling approach.

2.4 Feature Selection Algorithm

Our feature selection algorithm (see Algorithm 2) is inspired by a similar method [9] already applied to the Heritage Health Prize dataset [8]. It is a backward feature elimination method able to select an accurate feature subset with acceptable time and computing resources. Our variant computes the model performance of Step 2) using the evaluation metric discussed in Sect. 2.3, and Steps 3) and 6) compute the feature rank using the capabilities of the Random Forest R package. We have also added Steps 10)–19). These additional steps require a human expert. Steps 10)–15) correspond to a backward feature elimination method and Steps 16)–19) correspond to a method to add features. For this challenge, the semantic similarity among features was evaluated by one of the authors with amateur trading skills (e.g., two similar features are the exponential moving averages of the last 3 and 5 bid prices. Two semantically orthogonal features are the exponential moving average of the last 3 bid prices and the bid/ask spread before the liquidity shock).

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 $S_i$=$S, S-1, \ldots, 1$ : 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 se-
        lected in Steps 1)-15)
18:     If smaller RMSE, then we add the feature to $S_f$
19: until no improvement
```

3. Validation

The first step to validate the model ideas was to select a suitable feature subset. We applied the feature selection algorithm described in Sect. 2.4 to the same sample dataset used in the same section to evaluate the suitability of different modeling approaches. According to the discussion in Sect. 2.2.2, we should have applied our feature selection algorithm separately to single piece bid and ask models defined in the full time interval 52–100. However, due to time constraints, we only optimized one side F_b. F_a was estimated from F_b by just taking the ask side of price features. The selected feature sets F_b and F_a were used by all bid and ask sub-models respectively as suggested by the optimization problem relaxation described in Sect. 2.2.2.

The following step was to learn the optimal set P of time intervals. We used again the same sample dataset used in Sect. 2.3 to evaluate the suitability of different modeling approaches and computed the F_b feature subset found in the previous step. The application of the partitioning algorithm (Sect. 2.1) on the bid model (i.e., $M_{\text{bid}}(t)$) delivered the following set of time intervals: {52–52, 53–53, 54–55, 56–58, 59–64, 65–73, 74–100}.

Our final model fitting setup consisted of three datasets of 50000 samples each randomly sampled from the training dataset and with the same *security_id* proportions as the test dataset. We computed F_b and F_a for each sample and trained a complete model with each training dataset. Finally, the models were applied to the test dataset and the three predictions from each sub-model were averaged. Figure 1 shows the evolution of the public and private scores during the two months of the competition. The private score is the RMSE on the full test set. The private score was only made public after finishing the competition. The public score is the RMSE calculated on approximately 30% of the test data.

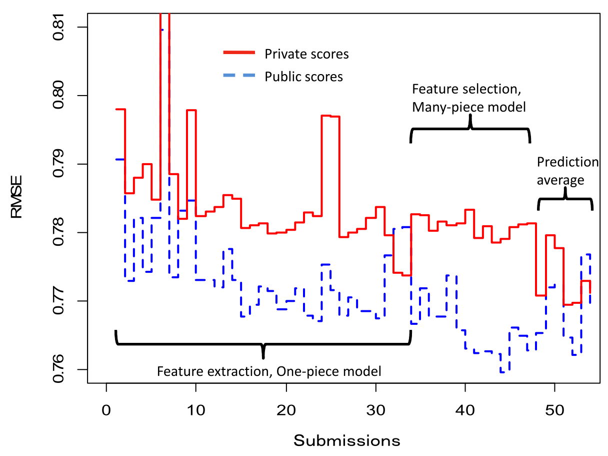

Fig. 1 Evolution of public (dashed line) and private (solid line) scores. Feature extraction, selection, model partitioning and final prediction average were performed sequentially as indicated in the plot.

The public score was available in real time to all competitors.

Public and private scores are more correlated during the initial process of adding new features than during the second process of selecting the optimal feature subset. This lack of correlation could be due to a methodological bug during the execution of the feature selection algorithm: During this step the authors used a single dataset of 50000 samples instead of the three datasets used in the other validations stages. This could have biased our feature selection towards those features that better explain this particular dataset. Finally, the best solution was obtained by averaging the predictions obtained from each of three models fitted respectively with the three sample datasets (last three submissions). In order to test the main hypothesis, we also submitted an additional solution based on a single-piece model (4$^{\text{th}}$ and 5$^{\text{th}}$ last submissions). The many-piece model solution clearly provided the final edge to win. Therefore, we consider that this is a strong positive indicator for the model hypothesis discussed in Sect. 2. It is worth mentioning that, according to information disclosed in the challenge forums, a common modeling choice among most competitors was to use a single time interval with constant post liquidity shock bid and ask prices. This is an additional clue that points to the suitable implementation of our main hypothesis as a key component of our solution.

4. Conclusions

This letter presented our solution for the Kaggle Algorithmic Trading Challenge. Our main design hypothesis was that the predictive potential close to the liquidity shock should be higher and it should be degraded with the distance. This hypothesis guided the design of our model architecture and it required a complex feature extraction and selection strategy. An additional self-imposed constraint on this strategy was to uniquely generate semantically meaningful features. This self-imposed constraint was motivated by the authors' will to generate a predictive model with the highest explanatory potential, but it also helped to identify features which had not been initially selected using a simple backward feature elimination method.

Acknowledgements

The authors were partially supported by the FIRST program.

References

[1] Kaggle Algorithmic Trading Challenge, http://www.kaggle.com/c/AlgorithmicTradingChallenge/data
[2] F. Lillo, J.D. Farmer, and R.N. Mantegna, "Master curve for price-impact function," Nature, vol.421, pp.129–130, 2003.
[3] T. Foucault, O. Kadan, and E. Kandel, "Limit order book as a market for liquidity," Review of Financial Studies, vol.18, no.4, pp.1171–1217, 2005.
[4] J. Ulrich, "Package TTR: Technical trading Rules," CRAN Repository: http://cran.r-project.org/web/packages/TTR/TTR.pdf
[5] D. Reshef, Y. Reshef, H. Finucane, S. Grossman, G. McVean, P. Turnbaugh, E. Lander, M. Mitzenmacher, and P. Sabeti, "Detecting novel associations in large datasets," Science, vol.334, no.6062, pp.1518–1524, 2011.
[6] A. Liaw, "Package randomForest: Breiman and Cutlers random forest for classification and regression," CRAN Repository: http://cran.r-project.org/web/packages/randomForest/randomForest.pdf
[7] G. Ridgeway, "Package gbm," CRAN Repository: http://cran.r-project.org/web/packages/gbm/gbm.pdf
[8] Heritage Provider Network, Health Prize, Site: http://www.heritagehealthprize.com/c/hhp
[9] T. Van Nguyen and B. Mishra, "Modeling hospitalization outcomes with random decision trees and bayesian feature selection," Unpublished, Site: http://www.cs.nyu.edu/mishra/PUBLICATIONS/mypub.html