Finance Project
Stat 222, Spring 2016

Mingyung Kim, Weiyan Shi, Chih-Hui Wang

March 30, 2016

Abstract

We model the spread crossing and mid-price movement of the Apple stock price using Support Vector Machine and Random Forest. Based on the predictions from both machine learning techniques, we design a trading strategy to gain profits in the stock market. Furthermore, because the interpretability of the machine learning method, we use the logistic regression to fit the data and try to figure out what features influence the stock price movement the most.

1 Introduction

In this project, we explore the high-frequency trading activity in the stock market. The data is the limit order book of Apple on May 22nd, 2012. Our primary task is to predict and model the mid-price movement and spread crossing. Furthermore, we will use the model to develop the trading strategy and implement it using the real world data. We use Python to do the data preprocessing and modeling, especially scikit-learn [1], and the ggplot2 package in R to do the visualization. The codes are all available in our Github repository [1].

In Section 2, we explain how we extract the features from the limit order book and describe how we define the mid-price movement and spread crossing. For modeling, we mainly focus on Support Vector Machine (SVM) and Random Forest. In Section 3, we show the modeling procedures for both models. For Section 4, we present the evaluation of the model performance on the test set. One of the problems of machine learning algorithms is that they normally are hard to interpret, so we also fit the logistic model and try to explain some crucial features of the model which show in Section 5. In Section 6, we implement our trading strategy based on the models and discuss the profit and potential drawback of the models. Finally, we will give an overall conclusion and discussion in Section 7.

2 Data Preprocessing

2.1 Feature Extraction

We preprocess the given dataset into a data appropriate for our further modeling. We add features that can be useful for our prediction. We followed Kercheval and Zhang (2015) [2] and include the six feature vector sets as shown in Table 1.

Table 1: Feature Vector Sets

Feature set	Description (ı=level index, n=10)
v_1=$\{P_i^{ask}$, V_i^{ask} , P_i^{bid} , $V_i^{bid}\}$	price and volume (n levels)
v_2=$\{($ P_i^{ask}-P_i^{bid} $)$, $($ P_i^{ask}+P_i^{bid} $)/2\}$	bid-ask spreads and mid-prices
v_3=$\{($ P_n^{ask}-P_1^{ask} $)$, $($ P_n^{bid}-P_1^{bid} $)$, $\lvert P_{i+1}^{ask} - P_i^{ask}\rvert$, $\lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}$	price differences
v_4=$\{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask}$, $\frac{1}{n}\sum_{i=1}^{n} P_i^{bid}$, $\frac{1}{n}\sum_{i=1}^{n} V_i^{ask}$, $\frac{1}{n}\sum_{i=1}^{n} V_i^{bid}$ $\}$	mean prices and volumes
v_5=$\{\sum_{i=1}^{n}($ $P_i^{ask} - P_i^{bid}$ $)$, $\sum_{i=1}^{n}($ $V_i^{ask} - V_i^{bid}$ $)\}$	accumulated differences
v_6=$\{dP_i^{ask}/dt$, dP_i^{bid}/dt , dV_i^{ask}/dt , $dV_i^{bid}/dt\}$	price and volume derivatives

[1] https://github.com/jJasonWang/STAT_222_Finance

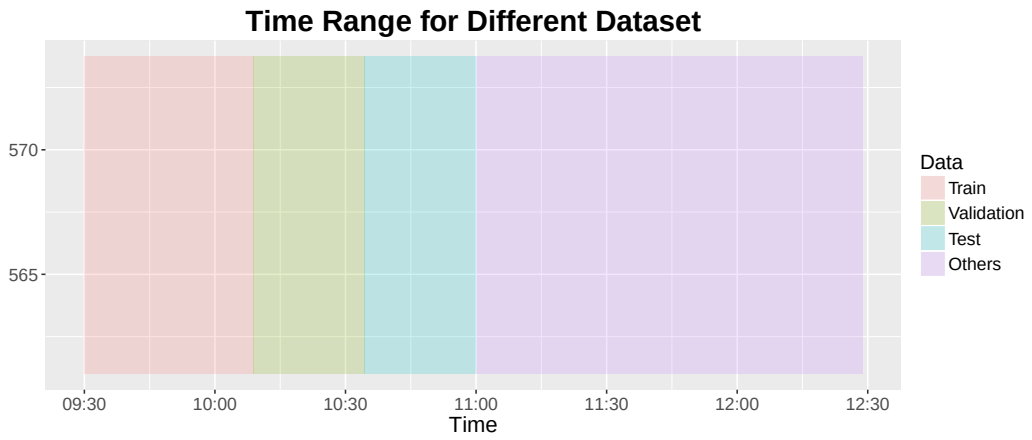

Figure 1: Time Range for Different Dataset

Attributes in v_1 are prices and volumes up to n=10 levels at both ask and bid sides. They can be closely related to price movements. It is because investors make decisions based on prices and volumes and those decisions can affect price afterwards. Attributes in the v_2, v_3, v_4, v_5 are summary statistics (e.g. sum, mean, difference) of prices and be useful in predicting price movements. Of note is that the attributes except for those in v_6 are not dependent on time. We can consider the recent movements of prices and volumes by measuring the time-sensitive attributes in v_6. Those are average time derivatives of price and volume over the most recent one second.

2.2 Dependent Variables: Mid-price Movement and Spread Crossing

Our first target is the mid-price movement. The mid-price of the stock is defined as the average of the best (highest) bid price and the best (lowest) ask price. At time t, we label the observation as up, stationary, or down if the mid-price at $t + \Delta t$ is bigger than, equal to, or smaller than the mid-price at t. We define time in terms of the number of time-stamped trade events. In the limit order book, each record represents one transaction. It may be the execution or cancellation of the market order. To balance the number of observation in each group, we choose Δt equal to 10 trading events, which, by using $\Delta t = 10$, the number of observations for up, stationary and down is around 1:1:1.

Predicting spread crossing is our second goal. The spread crossing is up if the best bid price at $t+\Delta t$ is bigger than the best ask price at t. It is down if the best ask price at $t + \Delta t$ is larger than the best bid price at t. Otherwise, it is stationary. Same as the mid-price, we also define the time in term of the number of time-stamped trade events. When there is a movement for spreading crossing, it gives us the high chance to make a profit if predicted correctly in advance. For example, if we predict that the spread crossing is up at t, then we can buy the share at t and sell them at $t + \Delta t$. The profit will be the difference between the best bid price at $t + \Delta t$ and the best ask price at t. Similarly, if the prediction at t is down, we can short sell the share at t and buy it back at $t + \Delta t$. To make the number of observations balanced across three categories, we set the Δt as 1000.

3 Model Fitting

For the model fitting, we split the dataset into two parts: 9:30 to 11:00 and 11:00 to 12:00. We use the first part of the dataset to train the model and assess the model performance. Then, we develop the trading strategy based on the model and implement on the second part of the dataset. The details of our trading strategy show in Section 6. There are 203,349 observations in the first part and 129,324 in the second part.

We further split the first part of the dataset into three parts: training (50%), validation (25%) and testing sets (25%). We use time order to separate the dataset. In other words, the first 50% of the data will be in the train set, the next 25 % of the data will be in the validation set, and the last 25% of the data will be in the test set. We use Figure 1 to summarize how we separate the data into different parts.

We implement two machine learning techniques: Support Vector Machine (SVM) and Random Forest. To tune the parameters in each model, we fit the models with the different parameters for both machine learning methods on the training set and compare their performances on the validation set. Afterward, we can choose the optimal parameter based on the accuracy. Finally, we make the prediction on the testing set to present the model assessment.

3.1 SVM

For SVM, we use Gaussian kernel. The paper gives comprehensive explanations how SVM algorithm works, and we only focus on how we tune the parameters in this project. We have to tune two parameters C and γ. The choice of the parameters shows the bias and variable trade-off for the model. If C is too large, the model will underfit the data which leads a wrong trading strategy. Similarly, if γ is big, the decision boundary will be more smooth which indicates more bias and less variance.

We try $C = 100, 1000, 10000$ and $\gamma = 10^{-5}, 10^{-6}, 10^{-7}$ for both mid-price movement and spreading crossing. Since we have two parameters, we use the grid search to find the ideal pair of the parameters. In other words, we will fit a model for each pair of the parameter and compare their performances. However, due to the computational issue, we do not fit the model with the full training set. Instead, we use first 50,000 samples in the training set. We fit the model and tune the parameters for both mid-price movement and spread crossing. For the mid-price movement, it turns out that the optimal parameters are $C = 100$ and $\gamma = 10^{-6}$. Most of the accuracies are around 33%. For the spread crossing, the optimal parameters are $C = 1000$ and $\gamma = 10^{-5}$. Most of the accuracies are around 35%.

3.2 Random Forest

For the random forest, the parameters we need to select are the number of features in each split and number of trees. As the name suggest, the idea of random forest is to grow lots of trees. Each tree will make a prediction, and it will output the majority class as the predicted value. Also, to decorrelate the features, in each split of the tree, it only uses the subset of the features. Since we do not consider the full features set in each split, it makes the bias of the model increase. However, the number of trees will help us reduce the variance for the model. The effect of reducing variance will bring us some gains. Therefore, we need to choose an ideal pair of these two parameters.

For the number of features, we use two methods. The first one is $\log_2 m$ and the second is $\sqrt{m}$ where m is the total number of features we have. On the other hand, for the number of trees, we try the numbers $50, 100, 200$. In general, the model performances remain the same after 200 trees, and the only cost for fitting many trees is computation. We do the grid search like SVM. Fitting the random forest seem to be not so time consuming as fitting SVM. We use the full training set for the random forest. For mid-price movement, the optimal number of features is $\sqrt{m}$ and the number of trees is 100. It turns out that the optimal parameters for spreading crossing are the same as the one for mid-price movement.

4 Model Assessment

To provide a complete picture of the models, we use precision, recall and F1-score to evaluate them. Table 2 shows the confusion matrix and how to calculate those measures. The precision is the proportion of correct prediction when we predict positive, in our case: up, stationary or down. The recall is the correct prediction when the true label is positive. F1-score is an aggregate index combining measures. If all of them are close to 1, the model is great. Note that because our data have three classes: up, stationary, down, we transform the class into binary classes when we calculate the measures. That is, we will use the confusion matrix of up and not up when we calculate the precision for class up.

Table 2: Confusion Matrix for Binary Classification

		True	
		Positive	Negative
Predict	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- Precision $= \frac{\text{True Positive}}{\text{True Positive + False Positive}}$
- Recall $= \frac{\text{True Positive}}{\text{True Positive + False Negative}}$
- F1-score $= \frac{2 \text{ Precision} \times \text{Recall}}{\text{Precision + Recall}}$

4.1 SVM

For SVM, instead of refitting a model using all the train and validation data, we directly use the best model we found during the process of tuning parameters. That is, we only use 50,000 samples when we fit the model. We show the three measures of SVM model for mid-price movement in Table 3 and spread crossing in Table 4. In fact, the model is not that much generalizable, and does not work well with the test data. We further check our confusion matrix and find out that for both cases, the SVM tends to make the prediction mainly on one class. That is why we observe the high recall in up class in each model.

Table 3: Model Assessment for SVM: Mid-price Movement

	Precision	Recall	F1-score
Up	30.3%	33.8%	31.9%
Stationary	38.6%	0.164%	23.1%
Down	32.7%	51.3%	39.9%

Table 4: Model Assessment for SVM: Spread Crossing

	Precision	Recall	F1-score
Up	33.3%	20.2%	25.2%
Stationary	38.7%	13.1%	19.6%
Down	34.9%	71.0%	46.8%

4.2 Random Forest

For the random forest, we refit the model on the train and validation data using the same parameters and make the prediction on the test data set. Table 5 shows the result for mid-price movement, and result for spread crossing is presented in Table 6. It seems that the random forest can predict three classes but it does not predict very well with the new dataset.

Table 5: Model Assessment for Random Forest: Mid-price Movement

	Precision	Recall	F1-score
Up	40.7%	46.5%	43.4%
Stationary	47.4%	15.8%	23.8%
Down	37.7%	62.2%	46.9%

Table 6: Model Assessment for Random Forest: Spread Crossing

	Precision	Recall	F1-score
Up	38.1%	28.5%	32.6%
Stationary	31.8%	22.2%	26.1%
Down	33.7%	52.1%	40.9%

We try to figure out why our model perform so worse on the test dataset. As shown in the Figure 3. Actually, the movement for the data in the training set does not share the same trend as the data in the testing set. Besides, we do not include the time sensitive feature in our model. This is probably the reason why our model does not perform so well on the testing data.

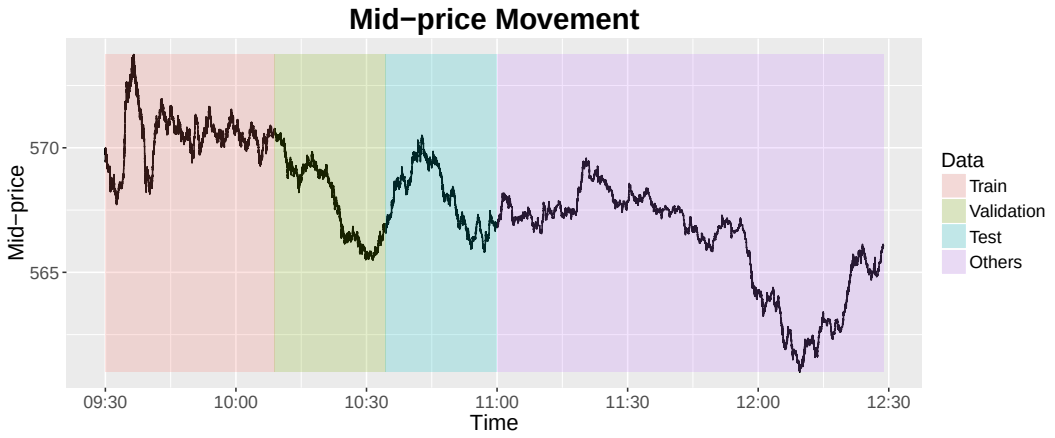

Figure 2: Mid-Price Movement

4.3 Feature Importances in Random Forest

In this section, we show the feature importances in the random forest model for spread crossing. We can calculate the contribution of each feature based on the split made by that particular feature. In classification, we can use the reduced impurity as a measure to evaluate the importance or contribution of each feature. In regression, we can use the reduced variance to measure the feature importance. We show the features with top 10 high importances in Figure 3. In fact, most of the features contribute almost the same amount which means that they are equally important.

5 Interpretation

We come up with a new variable v_7 that can improve our prediction. The ratio of the total volume of top five ask-side orders and the total volume of top five bid-side orders can be an indication of the imbalance of supply and demand.

Table 7: New Feature Set

Feature set	Description (ı=level index, n=10)
v_7=$\{\sum_{i=1}^{5} V_i^{bid} / \sum_{i=1}^{5} V_i^{ask}\}$	volume ratio

We apply logistic regressions to predict the spread crossings. The equation is:

$$\begin{aligned} y_{it} &\sim Binary(p_{it}) \\ logit(p_{it}) &= \vec{\beta}^{\top} \vec{x_{it}} \end{aligned} \tag{1}$$

, where the dependent variables y_t indicate dummy variables: (1) whether the spread crossing is up or not (i=1), (2) whether it is stationary or not (i=2), or (3) whether it is down or not (i=3) at time t. x_{it} includes the six feature vectors in Table 1 as well as the new feature in Table 7.

We use the same model assessment measurements to compare with those obtained from the sophisticated model. By comparing Table 1 with Tables 2-6, we find out that the prediction accuracy of the logistic regression model is not lower than that of the more sophisticated machine learning algorithms. The logistic model performs better than the other models in predicting the class "stationary" (Precision: 57.8% for the logistic model, 38.6% for SVM, 31.8% for Random Forest).

Table 8: Model Assessment for Logistic Model: Spread Crossing

	Precision	Recall	F1-score
Up	8.7%	25.4%	12.9%
Stationary	57.8%	35.9%	44.3%
Down	34.9%	32.8%	33.8%

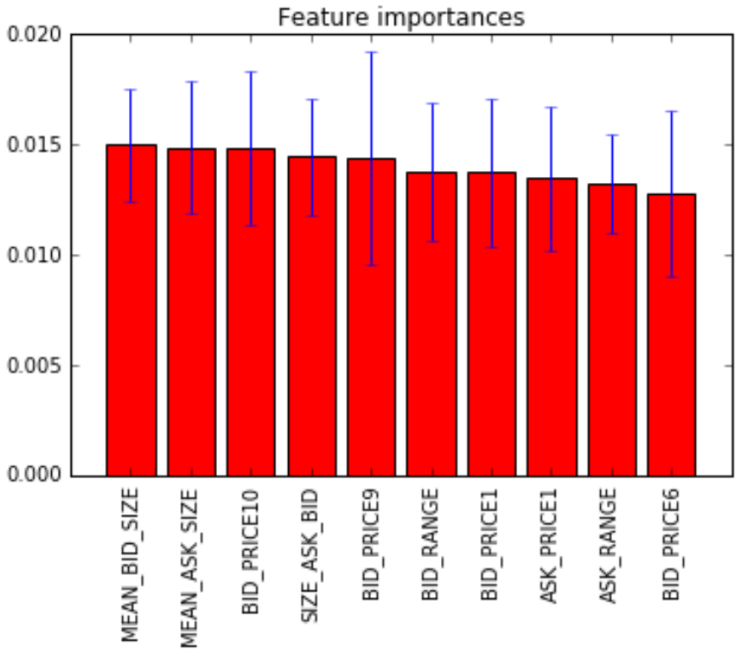

Figure 3: Feature Importances for Random Forest

The logistic model also helps the interpretation of the results compared to the sophisticated machine learning algorithms with a strong black box flavor. Table 9 introduces the three variables that were estimated to have largest impacts on the spread crossings. The estimates of those variables were reported in parentheses. For example, the estimate of dV_1^{ask}/dt is the largest with -0.0051 when we set dependent variable as a dummy variable whether the spread crossing is up or not. This result can be interpreted as: (1) dV_1^{ask}/dt is the most important in predicting whether a spread crossing is up, (2) for a one-unit increase in dV_1^{ask}/dt, the expected change in log odds of having a up spread crossing is -0.0051. As we should interpret the coefficients using log odds, the interpretation may not be much intuitive compared to linear model. Still, the interpretation using the logistic model is much easier than that of the SVM and random forest.

One more interesting finding is that our new feature (v_7) works well in the logistic models for the stationary and down spread crossings. This feature is the second most important variable for those models. For example, the estimated v_7=$\sum_{i=1}^{5} V_i^{bid}/\sum_{i=1}^{5} V_i^{ask}$ is -0.0105 for the logistic regression model for up spread crossing. This means that the change in log odds of having downward spread crossing is expected to -0.0105, for a one-unit increase in v_7=$\sum_{i=1}^{5} V_i^{bid}/\sum_{i=1}^{5} V_i^{ask}$.

Table 9: Three Variables with the Largest Impacts on the Spread Crossing

	First	Second	Third
Up	dV_1^{ask}/dt (-0.0051)	dV_2^{ask}/dt (-0.0046)	dV_5^{ask}/dt (-0.0045)
Stationary	$\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid})$ (0.1871)	$\sum_{i=1}^{5} V_i^{bid}/\sum_{i=1}^{5} V_i^{ask}$ (0.0527)	P_{10}^{ask}-P_{10}^{bid} (0.0231)
Down	$\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid})$ (-0.0741)	$\sum_{i=1}^{5} V_i^{bid}/\sum_{i=1}^{5} V_i^{ask}$ (-0.0105)	P_{10}^{ask}-P_{10}^{bid} (-0.0099)

6 Trading Strategy

To simplify the problem, we make the following assumptions:

(a) There is no transaction cost.

(b) We can place only market orders, and assume that they can be executed immediately at the best bid/ask.

(c) The position can only be long/short at most one share.

(d) We can only hold one share during the whole time period.

We implement the same strategy for both SVM and random forest. And the strategy is:

Table 10: Trading Strategy

	Meaning	Action	Profit
Up	${P_{t+\Delta t}}^{bid} > {P_t}^{ask}$	sell at $t+\Delta t$	$Bid\_price1_{t+\Delta t} - Ask\_price1_t$
Stationary	${P_{t+\Delta t}}^{bid} = {P_t}^{ask}$	Do nothing	0
Down	${P_t}^{bid} > {P_{t+\Delta t}}^{ask}$	borrow a share at t, buy it back at $t+\Delta t$	$Bid\_price1_t - Ask\_price1_{t+\Delta t}$

Here is how we implement the strategy. We will start our transaction right at 11:00 am, and predict the label for the very first observation and then take some action according to the table above. Because we can only have one share at each timestamp, we have to sell/buy the share at $t+\Delta t$. From time t to time $t+\Delta t$, we could not take any action, so we have to jump to $t+\Delta t$ and predict the label for $t+\Delta t$. Then we move to $t+\Delta t$, so and so forth. In total, we would only have about 129 actions with 129,324 observations in total and Δt to be 1000. For SVM, our profit is -$20; for random forest, our profit is -$24; for logistic regression, our profit is -$12.

We have negative profit because the models didn't do well on the data after 11:00 am, we could see from the model assessment that all the classifiers are basically "guessing" the label. The logistic regression performs slightly better maybe because it performs better on predicting the "stationary" label with precision for class "stationary" being 57.8%.

And one of the reasons why the models don't perform well on the data after 11:00am is because the pattern for the price after 11:00am is not similar at all to the data before 11:00am, we could see from Figure 2 that after 11:00am, the price starts decreasing a lot. And because we didn't have many time-sensitive features in our dataset, we couldn't catch the downward trend. We could do some further improvements on that. In conclusion, using the data before 11:00am to build the model and predicting the labels after 11:00am is not a good idea.

7 Conclusion

In this project, we use the limit order book of Apple on May 22nd, 2012, and try to predict the mid-price movement and spread crossing. We choose the features that could help predict the two dependent variables. We first apply the sophisticated machine learning algorithms (e.g. SVM and Random Forest), evaluate model performances (e.g. precision, recall, F1-score), and measure the profits by implementing our strategies. We also fit the simple algorithm (e.g. logistic model) and compare the model performances as well as the profits with the sophisticated models. In fact, We can improve the interpretability without losing prediction accuracy or profits in a significant amount, by using a simple model.

Given the limitation of time and data, we do not improve the performance of the models much other than tuning the parameters. Moreover, because we do not have the sufficient knowledge about the data, the time-sensitive features are hard to implement. However, the time information of the stock is crucial for analysis. We can add more relevant features such as the lag of bid or ask price in the future to catch the time trend for the price movements, which may also be a potential way to improve our profit.

Besides, as said before, the data before 11:00 am is not a good indicator for the data after 11:00 am. One way to improve the profit is to combine different classifiers, make predictions and let them vote for the final predicted label. This method may increase our accuracy in the future and give us higher profit. Alternatively, we can use similar updating model strategy as did in Kercheval and Zhang (2015) [2] We keep update our model when we have a new observation. By doing so, we will capture the time trend of the data.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Alec N Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. In *Quantitative Finance,*. 15(8):1315-1329, 2015. http://www.math.fsu.edu/~kercheva/papers/multi-svm.pdf.