# IEOR 222 Final Project

# Comparison of tree based models and SVMs on High-frequency LOB dynamics and feature importance study

**Shengying Wang, department of statistics, UC Berkeley**

**Dec 10 2014**

**Abstract**

In this project, we implement a machine learning framework to study the relationship between stock market dynamics and limit order books in terms of high frequency trading. By modeling features obtained in limit order book, we can utilize machine leaning techniques such as support vector machines, decision trees and random forests to predict price spread crossing as three classes, which are up, down and stationary. Then we test for prediction accuracy, features importance and models robustness among all these machine learning methods.

**Key words:** high-frequency limit order book; multi-class classifiers; decision trees; random forests; support vector machines(SVMs); machine learning

## 1 Introduction

In Kercheval & Zhang (2013), they propose a machine learning framework to capture the dynamics of high frequency limit order books (LOB) in financial equity markets and automate real-time prediction of metrics such as mid-price movement and price spread crossing. However, based on those ideas in Kercheval & Zhang (2013), we want to move forward to test for three things, prediction accuracy, features importance and models robustness among different machine learning methods. But now we are only focusing on tree based methods and SVMs.

First of all, in Kercheval & Zhang (2013), they implemented SVMs to build classifiers to predict the real-time stock price movement, and it worked well in a few examples. However, in financial markets, there are many different situations for each single stock. One cannot show that SVMs dominate all of other machine learning methods for all cases due to huge sampling errors and irreducible variance. So we propose tree bases methods to compete against SVMs in terms of prediction accuracy. Second, tree based methods have advantages in model interpretations. We

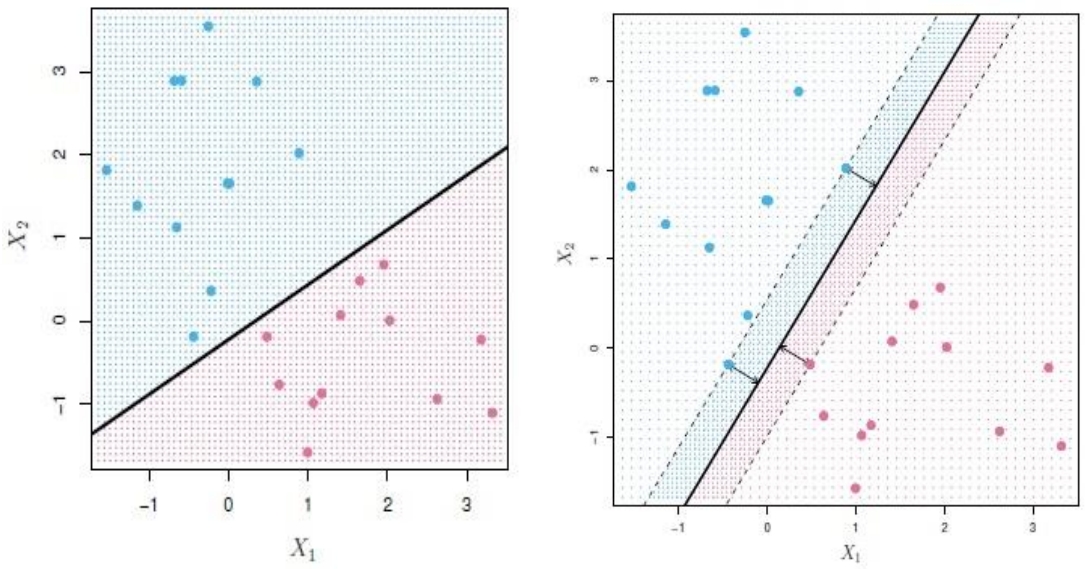
can discover the contribution of each feature and perform feature selection processes. It would also help us to identify those significant features which have closer relationship with real-time market dynamics. Also, we want to know the robustness of our models to see how frequent we have to retrain our data. If we are able to obtain robustness in some certain conditions, it indicates that it is better to have more observations in our training data sets. Otherwise, we have to update quickly with smaller size of the training data set obtained from LOB.

## 2 Support Vector Machines Framework

The basic support vector machine is a kind of binary classifier. It can find a hyperplane that separates the classes in feature space. A hyperplane in p dimensions is a flat affine subspace of dimension p − 1. In general the equation for a hyperplane has the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

In p = 2 dimensions a hyperplane is a line. If the left-hand side of the equation above is bigger than 0 for points in X, those points will be signed as one class. If the left-hand side of the equation above is smaller than 0 for points in X, those points will be signed as another class. The process is shown below as a two-dimensional feature space.




When we have more than two classes, we perform “one verse one” or “one verse all” to find the boundaries for all classes. However, it would spend a lot more time than a two-classes problem. If

we want to perform non-linear classification boundary, we can apply kernel functions. Two common kernels are polynomial (with degree d) and radium ones, shown as below respectively.

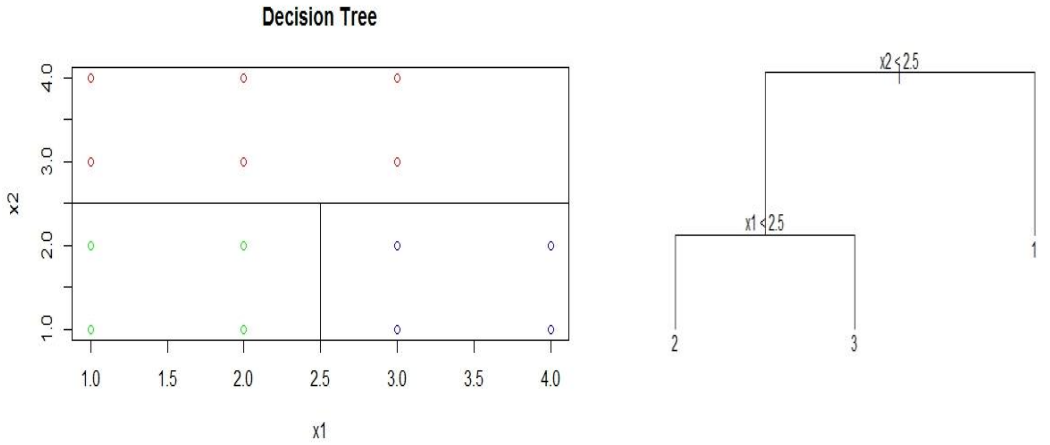
$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

The SVMs models have several advantages. First of all, it is very flexible and able to capture non-linear relationship between LOB features and market dynamics. Second, its prediction accuracy is competitive. Last but not least, it is computationally efficient. It can be down quickly so that we can update our trading decisions frequently.

## 3 Tree Based Methods Framework

In this project, we will perform two tree based models, Decision Trees and Random Forests. The idea of Decision Trees is that we divide the feature space, which is the set of possible values for X1, X2,…,Xp – into J distinct and non-overlapping regions (boxes), R, R2,…,RJ. For every observation that fall into the region Rj, we make the same prediction, which is the majority vote (major class) in the region Rj. We make splits to cut the whole feature space into those J boxes with minimizing the misclassification rate. A simple example with 3 classes and 2 features is shown below.




The left plot shows three regions (boxes) given by Decision Tree model which perfectly separate those points with three different classes by two splits. The right one shows the tree diagram. It will be applied later to interpret our LOB features and market dynamics.

The Random Forests model is to build a number of decision trees on Bootstrapped training samples. Then average all the tree functions to obtain the final one. But it has one assumption, which is that when a split in a tree is considered, a random selection of m predictors is chosen as candidates from the full set of p predictors. Each split is allowed to use only one of those m predictors.

Tree based methods have several advantages. First of all, it is very flexible. It is also a non-linear classification method. Second, tree based methods can report importance measure for each feature so that we can identify those key matrices. Also, Random Forests can be done in parallel to increase the speed. Comparing to SVMs, Random Forests might be competitive in terms of prediction accuracy.

## 4 Limit Order Book Dynamics and Feature Extraction

The idea to capture LOB dynamics as three categories which are up, down and stationary is based on the work done in Kercheval & Zhang (2013). When in a few steps in the feature, the best bid price is higher than the current best ask price, it is labeled as “up”. On the contrary, when in a few steps in the feature, the best ask price is lower than the current best bid price. It is labeled as “down”. If we don’t obtain any of these two, it is labeled as “stationary” (no spread-crossing). So in each time slot, we can obtain a class label from LOB if the number of steps

is well-defined.

In Kercheval & Zhang (2013), three sets of feature are used. They are basic set, time-insensitive set and time-sensitive set shown as below.

| *Basic Set* | Description($i$ = *level index*, $n = 10$) |
|---|---|
| $v_1 = \{P_i^{ask},\ V_i^{ask},\ P_i^{bid},\ V_i^{bid}\}_{i=1}^{n}$, | *price and volume (n levels)* |

| *Time-insensitive Set* | Description($i$ = *level index*) |
|---|---|
| $v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$, | *bid-ask spreads and mid-prices* |
| $v_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^{n}$, | *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* |

| *Time-sensitive Set* | Description($i$ = *level index*) |
|---|---|
| $v_6 = \{dP_i^{ask}/dt,\ dP_i^{bid}/dt,\ dV_i^{ask}/dt,\ dV_i^{bid}/dt\}_{i=1}^{n}$, | *price and volume derivatives* |
| $v_7 = \{\lambda_{\Delta t}^{la},\ \lambda_{\Delta t}^{lb},\ \lambda_{\Delta t}^{ma},\ \lambda_{\Delta t}^{mb},\ \lambda_{\Delta t}^{ca},\ \lambda_{\Delta t}^{cb}\ \}$ | *average intensity of each type* |
| $v_8 = \{\mathbf{1}_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\}$, | *relative intensity indicators* |
| $v_9 = \{d\lambda^{ma}/dt,\ d\lambda^{lb}/dt,\ d\lambda^{mb}/dt,\ d\lambda^{la}/dt\}$, | *accelerations(market/limit)* |

In the later analysis, we will perform SVMs and tree based methods to classify market dynamic (stock price movement) based on these three feature sets. However, there are a few key issues.

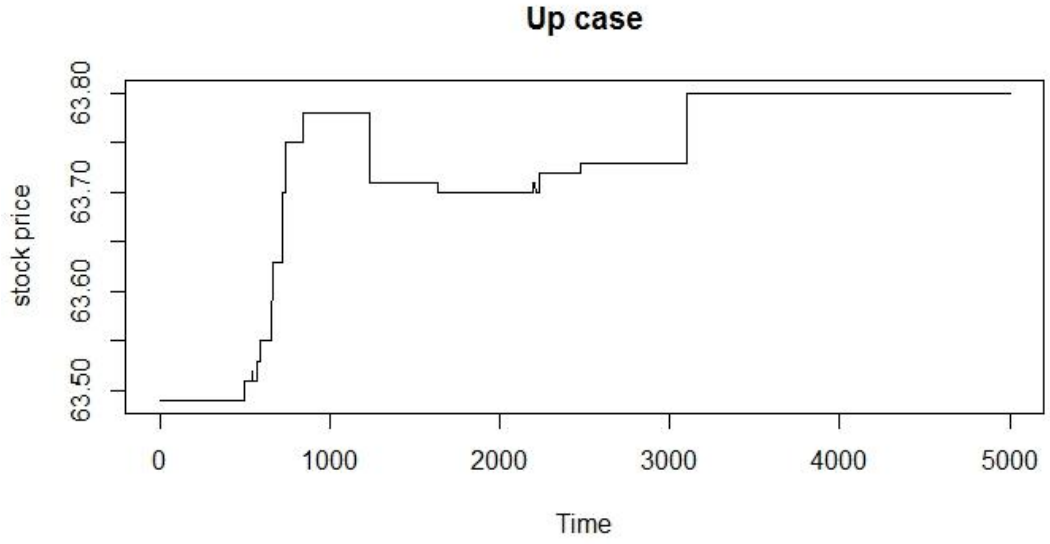
First, one can design different feature sets given LOB or message book or any other real trading records. In this project, we only have LOB with level 5. So our feature sets are not the same as the ones in Kercheval & Zhang (2013). And we don't have message book either. So we assume that cancellation of level 1 and market order are equally likely to happen.

Second, in Kercheval & Zhang (2013), authors choose to predict the spread-crossing in the next 5 steps (events). However, we think that the number of steps should depend on the real situations. For example, the stock prices for companies should move faster with more activities even in some certain conditions. But we don't obtain this for the stock prices all the time. In the sample data which we will use later, we predict the stock price movement in the next 2 seconds for DuPont, and 6 seconds for Tractor Supply Company and 10 seconds for Exxon Mobil. If we set the time period too small as 5 events, more than 95% of the labels in the training data sets are "stationary", which means that we don't have enough information to obtain the pattern of "up" class and "down" class. So now we are able to get more spread-crossings by setting the prediction time period a little longer as a couple seconds.

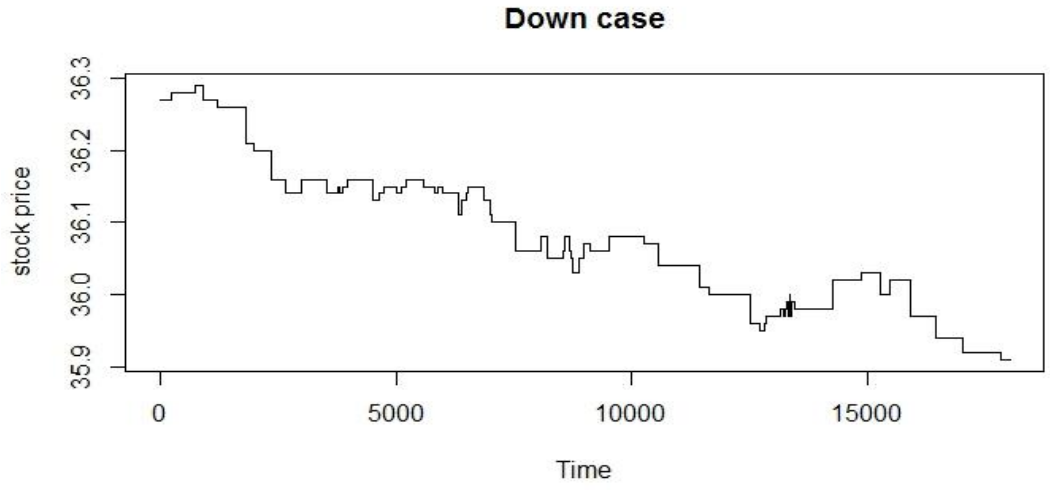
Third, in Kercheval & Zhang (2013), authors recommend to have each training data set with size roughly 1500 to 3000 observations. It would help to build real trading strategies in terms of time cost and prediction accuracy. However, in those three sampled stock data from three companies DuPont, Tractor Supply Company and Exxon Mobil, it is not easy observe a combination of three situations (up, down and stationary) in a single period with 3000 events all the time. We could also have situations with combinations of up and stationary only or combinations of down and stationary only. Therefore, in the later analysis, we will step through those three cases to see how our models perform for price going up, going down and going up and down. By merging them as two classes, we might lose some opportunities to make money. However, dealing with a two-class classification problems might improve the prediction accuracy and reduce the time costs.

## 5 Experiments and Results

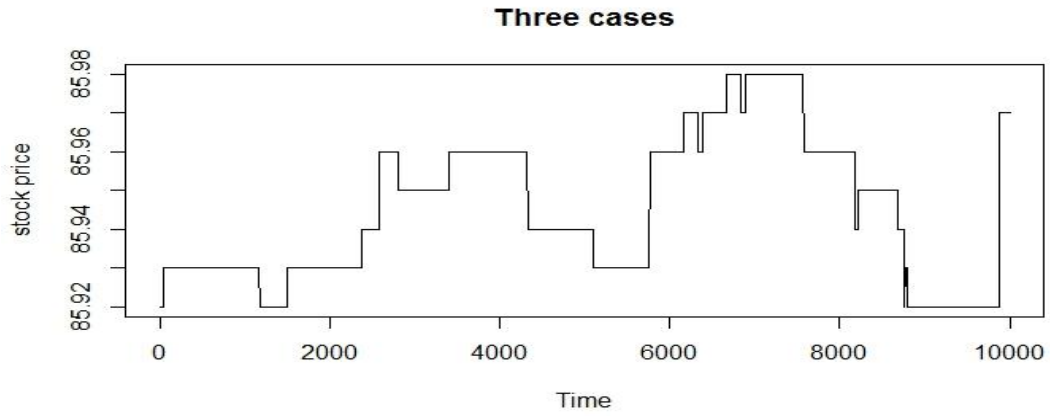
We get three stock samples from NASDAQ stock data sets. We will use those three sampled sets of data throughout this part of analysis. Sample 1 is from stock Tractor Supply Company, which is a case of price going up.




Sample 2 is from DuPont, which is a case of price going down.

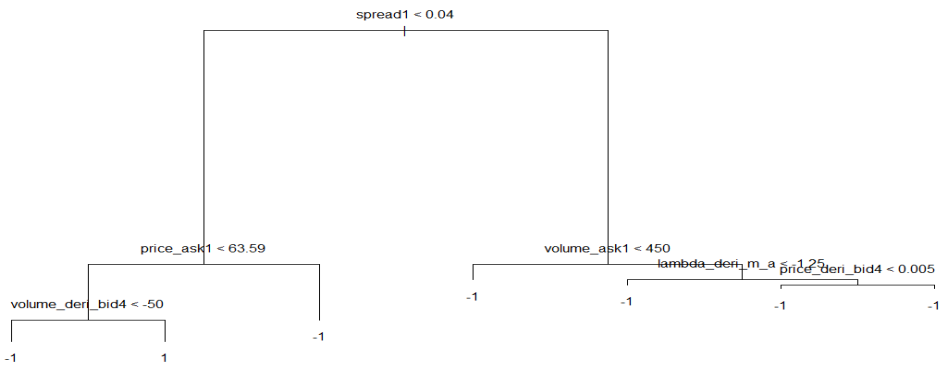



Sample 3 is from stock Exxon Mobil, which is a regular case of a combination of three situations.



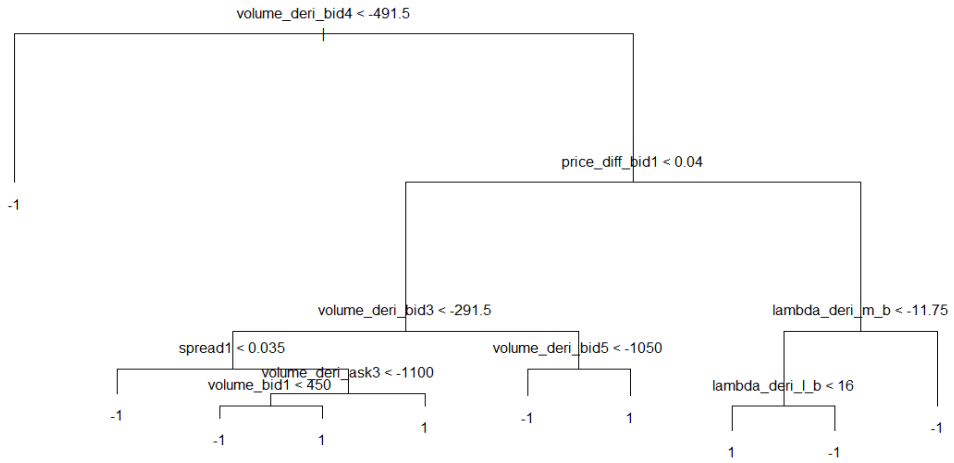

## 5.1 Interpretation of a Single Decision Tree Diagram

First we build a single tree model for the "up" case with 3000 observations. The tree diagram is shown as below. (up = 1, stationary = -1)



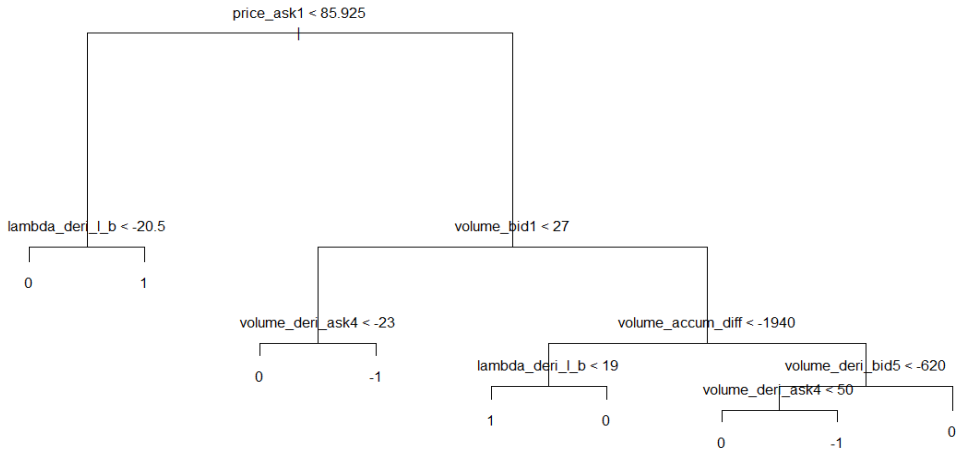
It is clear that spread is playing a major role in classifying up and stationary movements. If the "spread1" is not smaller than 0.04, it is classified as stationary. Otherwise, we need to take a look at "price_ask1" and "volume_deri_bid4". The only way to be classified as up is to have "spead1" less than 0.04 and "price_ask1" less than 63.59 and then "volume_deri_bid4" greater than or equal to 50.

Then we build a single tree model for the "down" case shown as below. (down = -1, stationary = 1)



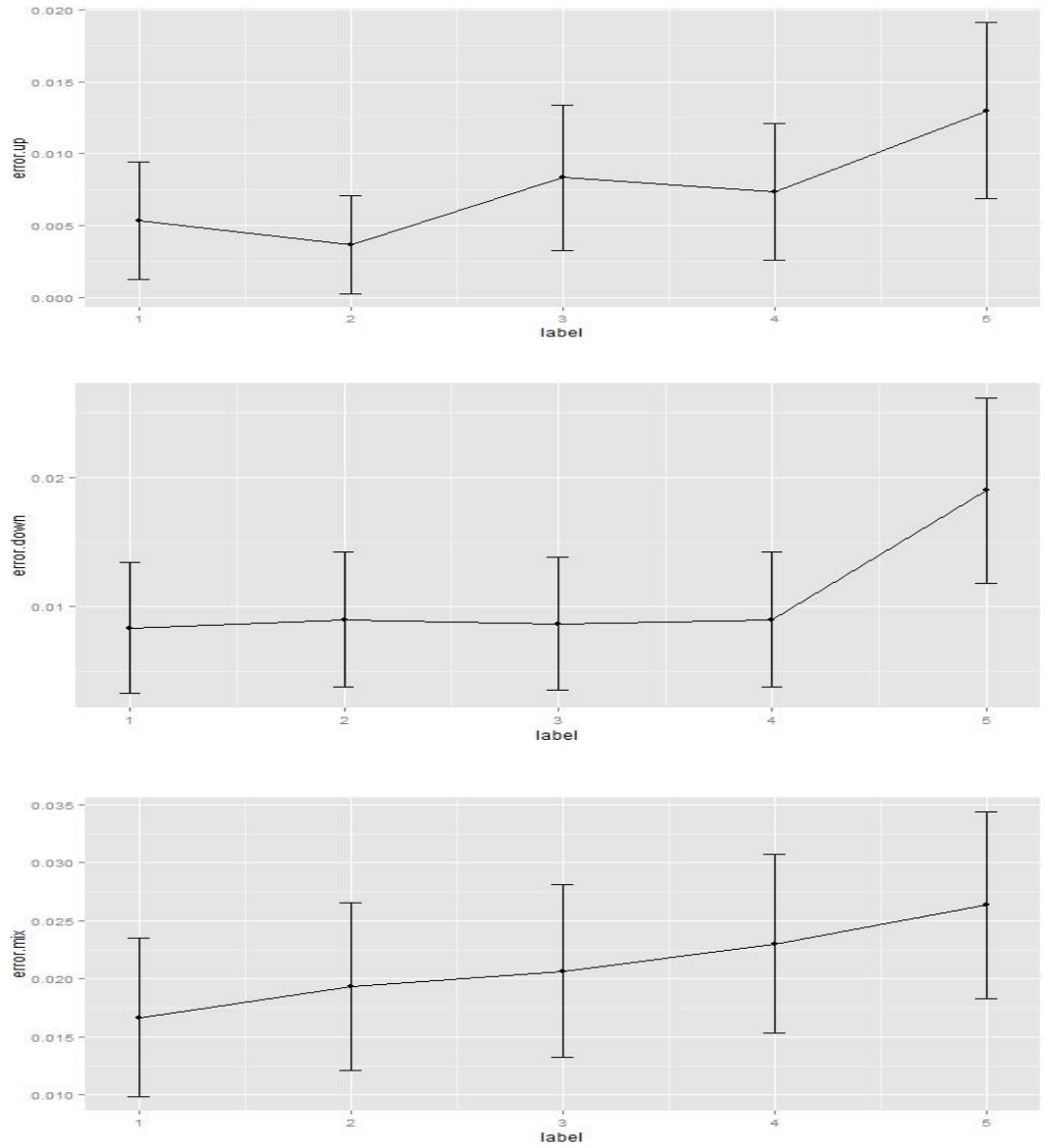

This time, the tree is a little complicated than the previous one with 10 nodes. Also, we have chosen volume and price features to make splits.

Then we build a single tree model for the "regular" case shown as below. (up = 1, stationary = 0, down=-1)



## 5.2 Comparison of Prediction Accuracy

Then we want to compare Random Forests with SVMs for prediction accuracy obtained by 10-fold cross validation error rate. Given 3000 observations for all three cases, we randomly divide 3000 observation as 10 subsets. Predict on one of those 10 subsets by fitting models with the rest 9 subsets of data. Then repeat 10 times so that each of them has been chosen as the test set once. Compute the number of times of misclassifications. The tuning parameters for each random forests model or SVMs model have been set as the optimal one obtained by cross validation error rate.

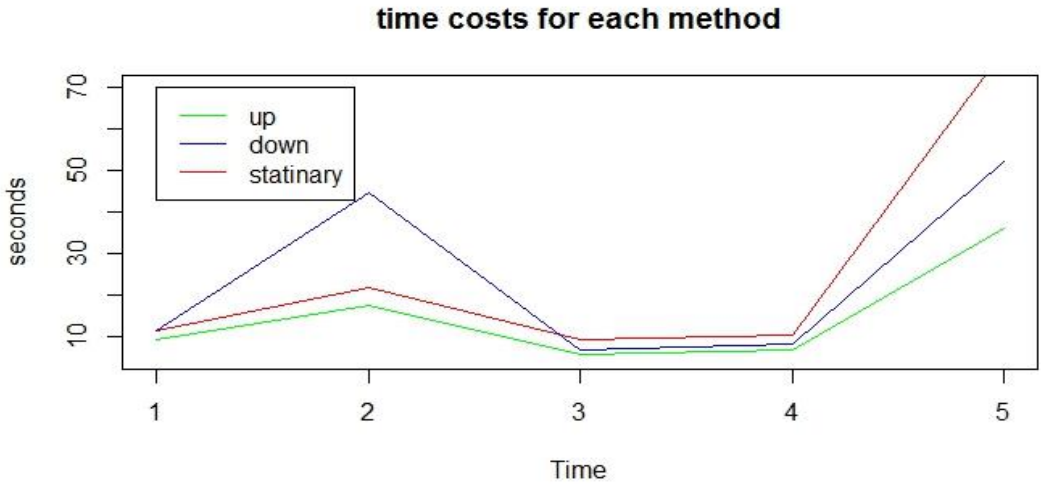



Label: **1**: random forests with 50 trees **2**: random forests with 200 trees **3**: SVM with linear kernel **4**: SVM with cubic kernel **5**: SVM with radial kernel

For “up” and “regular” case, random forests perform better than SVMs generally. For “down” case, random forests and SVMs are almost the same. However, SVM with radial kernel perform worse than any other method in all three cases. In general, random forests are competitive and SVM with radial kernel should be a bad choice. When considering a right number of trees in random forests, it is a tradeoff. It could be over-fitting if we grow too many trees. For SVMs, we don’t have evidence to see cubic kernel is better than linear kernel. There are over-fitting issues as well.

### 5.2 Comparison of Time Costs

Then we test for time costs. The results are shown as below.




Label: **1**: random forests with 50 trees **2**: random forests with 200 trees **3**: SVM with linear kernel **4**: SVM with cubic kernel **5**: SVM with radial kernel

From the plot above, we can see that generally “regular” case takes more time to run since in this situation, three classes are considered. Also, SVM with polynomial kernels are a little better than random forests. However, if we pick a right random forest model with an appropriate level of complexity, it would be competitive with SVM. Even if we pick a complex random forest model, it can be done in parallel to decrease the time cost. SVM with radial kernel is too slow.

### 5.3 Tests for Feature Importance

Next, we build 100 random forests in a 3000 observations in all three cases with three

different sampled stocks to test feature importance. Then print out the top 10 features which contribute most in terms of decreasing misclassification rates.

| spread1 | spread3 | price_accum_diff | spread4 | price_ask3 |
|---|---|---|---|---|
| 5880.120 | 4430.281 | 4101.554 | 3531.477 | 2745.262 |
| price_ask1 | price_mean_ask | price_ask5 | price_ask2 | spread5 |
| 2741.408 | 2619.677 | 2603.736 | 2573.764 | 2524.812 |

“up” case

| volume_deri_bid4 | price_diff_bid1 | volume_deri_bid3 | volume_ask2 | lambda_m_bid |
|---|---|---|---|---|
| 9205.707 | 5399.919 | 5192.244 | 5137.731 | 4045.056 |
| volume_bid1 | volume_deri_bid5 | lambda_l_bid | volume_ask4 | spread1 |
| 3189.179 | 3025.360 | 2633.816 | 2374.771 | 2353.565 |

“down” case

| lambda_deri_l_b | lambda_c_ask | volume_deri_ask4 | volume_accum_diff | volume_bid1 |
|---|---|---|---|---|
| 557.9530 | 471.4013 | 454.5249 | 449.6854 | 446.8271 |
| volume_deri_bid2 | volume_deri_bid5 | lambda_deri_m_b | price_mean_ask | lambda_l_bid |
| 426.0406 | 425.8615 | 416.4036 | 410.6990 | 409.1581 |

“regular” case

In “up” case, spread is the most important feature. However, in “down” case, volume is more significant. In “regular” case, both lambda (arrival rate) and volume dominate the others. In the “regular” case, top ten features are close in terms of their contributions. However, in “down” case, “volume_deri_bid4” is much significant than any other features. It suggests that in some particular situation for one stock, one particular feature can be very important but not for all other cases for different stocks in different time periods. Also, we should think about that whether there are strong correlations among those features, such as “spread1”, “spread3”, “spread4” and “spread5” in “up” case. When considering feature selection, we want to de-correlate the overall correlation among all the features to get the real key matrices.

We want to compare the results with another supervised feature selection method Lasso. By fitting logistics regression with Lasso penalty, we get top five features (depend on the value of estimated coefficients) for the same observations in “up” and “down” cases.

| price_diff_ask5 | price_diff_bid5 | spread1 | price_ask1 | price_deri_bid1 |
|---|---|---|---|---|
| -7.642773e+01 | -3.728444e+01 | -3.503420e+01 | -2.052818e+01 | -1.152060e+01 |

“up” case

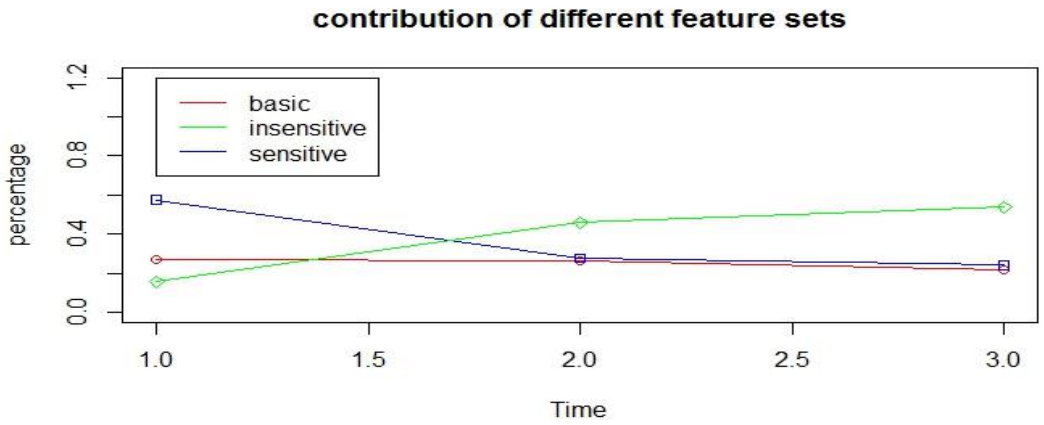
| price_diff_ask1 | price_diff_ask3 | price_diff_bid3 | lambda_m_ask | price_bid1 |
|---|---|---|---|---|
| 2.782643e+14 | 1.070470e+14 | 1.513864e+12 | 1.245938e+03 | 1.615552e+01 |

“down” case

The results from Lasso surprisingly are not quite the same as the ones obtained from random forests models. I think there are two main reasons. First, the random forests model is one of

those non-linear methods. It is able to build non-linear relationship between spread-crossing labels and LOB features. On the other hand, logistic regression with Lasso penalty is building linear relationship. It might not be sufficient to interpret by linear models only. However, non-linear modes might over-fit data as well. Secondly, when implementing random forests, we don't have to scale training data. However, using Lasso penalty, we have to scale training data in terms of L1 norm. So the feature importance results for those two methods may differ a lot.

Then we calculate the proportion of total contribution for each set of features by random forests with 200 trees.

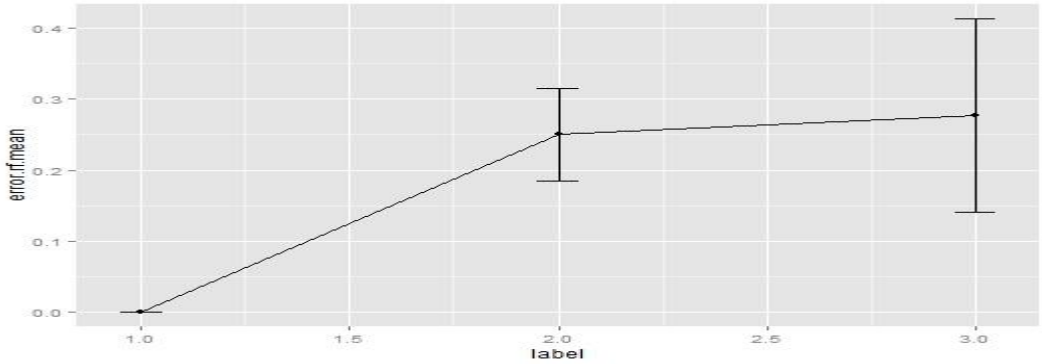



x label: **1**: "up" case **2**: "down" case **3**: "regular" case

The plot above shows that in terms of mean misclassification decreasing rate, all of three sets, Basic, Time-insensitive and Time-sensitive have contributions for each of those three situations, "up", "down" and "regular". The performance of basic sets is stable across each situation. However, the performances of the other two sets differ from different situations. But all of those three sets of features have significant contributions. But it is still necessary to do feature selection to reduce the dimension of the feature space for all of those three sets to filter out those noises, which would increase the speed and accuracy when implementing machine learning models on training data.

### 5.4 Robustness Study

Then we test for prediction robustness. We fit random forests model in "up" case with 2000 observations. Then predict for the spread crossing in next 1,5 and 10 seconds. Then repeat for 10 random samples to get mean error rates and standard deviations.




x label: **1**: error rate in 1 sec **2**: error rate in 5 sec **3**: error rate in 10 sec

The results show that when predicting for the spread-crossing in a longer period, we are getting larger prediction errors. In this case, the prediction accuracy for the next 1 second is surprisingly perfect. But for the next 3 seconds and 5 seconds, the mean prediction accuracies are very close. However, the standard deviations are getting larger when predicting in a longer period. So in general, it suggests that predicting for a shorter period is better in terms of both mean and variance of prediction error rates. Also we can see that overall error rates are higher than the 10-fold cross validation error rates obtained in the previous step. It suggests that the traditional cross validation methods would not work well in this type of problem since we are dealing with time series here. The traditional machine learning algorithms assume that all variables are independent. When implementing the traditional cross validation methods to get the optimal model, we have some unexpected bias coming from the dependence of those predictors and response variables. So there is no guarantee that the cross validation methods would give us the "best" model.

Then we test for robustness of feature importance. We divide 3000 observations within 3 minutes in the "up" case into 4 subsets in the order of time. Then we fit random forests model to each subset with 750 observations to get top ten features shown as below.

| price_mean_ask | mid_price2 | mid_price4 | mid_price5 | price_ask4 | price_ask1 |
|---|---|---|---|---|---|
| 44.90965 | 41.72142 | 38.21547 | 37.23550 | 36.44554 | 36.10279 |
| mid_price1 | price_ask2 | mid_price3 | price_ask3 | | |
| 33.97308 | 29.39380 | 25.73684 | 21.93168 | | |

first 750 observations

| lambda_deri_m_a | price_deri_bid5 | price_deri_bid4 | volume_ask1 | price_ask1 |
|---|---|---|---|---|
| 4.085210 | 2.680605 | 2.492422 | 2.388728 | 2.086369 |
| price_ask5 | price_ask2 | mid_price5 | mid_price1 | price_mean_ask |
| 1.757893 | 1.631791 | 1.513661 | 1.476331 | 1.183117 |

second 750 observations

| lambda_deri_m_a | price_deri_bid5 | price_deri_bid4 | volume_ask1 | volume_mean_ask |
|---|---|---|---|---|
| 4.29959326 | 2.84549321 | 2.65443566 | 2.50183664 | 0.90594729 |
| lambda_l_ask | lambda_deri_l_a | volume_deri_ask1 | lambda_c_ask | lambda_l_bid |
| 0.50689173 | 0.50637678 | 0.16392072 | 0.03472007 | 0.03397282 |

third 750 observations

| lambda_deri_m_a | price_deri_bid4 | price_deri_bid5 | volume_ask1 | volume_mean_ask |
|---|---|---|---|---|
| 4.39174218 | 2.92531263 | 2.69412199 | 1.75392579 | 1.68804661 |
| lambda_l_ask | lambda_deri_l_a | volume_deri_ask1 | lambda_l_bid | lambda_c_ask |
| 0.51965882 | 0.33839811 | 0.14381480 | 0.04452700 | 0.03642339 |

fourth 750 observations

For the first subset, it is more about price features. But for the second to the fourth, arrival rates and volume features are more significant. In general, in the period of 3000 events, we do obtain some robustness since most of the top ten features are about price, volume and arrival rates. Therefore, in this case, these three sets of features are key matrices. However, the orders of these features in different time slots are not table. Also, for different stock in different time periods, the top feature importance matrices should be different.

The results in testing robustness suggest that it is better to train our data quickly to get better test accuracy and small variance. Also we should do feature selection with a couple minutes to get those key feature matrices.

## 6 Conclusions

From our analysis focusing on SVMs and random forests, we finally get four main conclusions. First, fitting a single decision tree gives clear interpretation, and the tree size should be relatively small in general.

Second, when comparing the test accuracy for random forests against SVMs in terms of 10-fold cross validation, it shows that the results are close. It means that we can utilize both methods in building real trading strategies. However, in a certain condition, we have to decide the optimal parameters in random forests models and SVMs to get the best machine learning model. There are tradeoffs and we don’t believe that one method dominate the others all the time. The prediction results and time costs obtained by random forests are competitive. However, we still need to decide how complex of a random forests model we want. Even if we may have a very complicated random forests model, we still can build in in parallel to decrease the time. But in some cases, a simple random forests model would work well.

Third, all of those three sets of features (basic, time-intensive and time-sensitive) make contributions. Basic sets are more stable. The reason is that the other two sets features are calculated by different time periods. Since we build models in a very short period, some of them may have a little variability in a second and have huge jumps in the next second. That makes them shifting from noises to signals and back and forth all the time. Also, by comparing the feature importance results by random forests against logistic regression with Lasso penalty, we get different top features. Therefore, we have to decide to believe linear, scaled relationship or non-linear, unscaled relationship.

Last, we don't observe too much robustness in neither random forests nor SVMs in terms of prediction. When we build our model and test for spread crossing in the next couple seconds, we get relatively high error rates and variance. It suggests that it is better to train our data frequently to make better prediction accuracy. Since the variance is high, prediction performance would be vary a lot in different situations. Also, the feature importance results show that in different time slots with a couple minutes in one stock, the features contribution may vary a little. But the order of top features is changing all the time. So it is better to do features selection frequently as well.

## 7 Problems and Feature Work

In doing the analysis of studying the market dynamics by LOB features and machine learning methods, we come across some problems and parts which require some feature work.

First, we should think about the optimal number of time steps to be predicted. If we set the number of steps to small, it may give us better prediction accuracies. However, in some particular cases, we don't obtain so many spread crossings in one short period. So we may lose those market dynamic signals. We should somehow set the number of steps larger enough to get those signals but in the meantime should not do much harm to the prediction performance.

In this project, we have got a chance to do feature selections. However, from the results in feature importance study, we do see that most of features don't have much contribution in terms of decreasing the misclassification rate. So we should implement feature selection methods to filter out those features before building SVMs or random forests models. It will help to increase the prediction accuracy and decrease the time costs. But different feature selection methods may

give us different results like random forests and Lasso. So we have to decide which one to apply in terms of model selection and intuition.

Third, in terms of feature selections and interpretations, we have to deal with correlations among features. Those features extracted from limit order book may have strong correlations. When doing feature selections, we should come up a way to de-correlate those features.

Last, the response variable y (the spread crossing metric as up, down and stationary) may not be independent. In the basic concepts of machine learning, we assume that the response variables should be independent. However, in this case, we build models for time series. The dependence may introduce a lot bias in terms of model selection processes. For example, in 10-fold cross validation, we randomly pick training data in a certain period. So we predict he response variable by data in the feature. It will not happen in the real-world. So this kind of bias leads to have relatively larger test errors when we train data in the past and predict spread-crossings in the feature, because there is no guarantee that the best model obtained from traditional cross validation methods is the “real” best one. In the other words, even if we get perfect cross validation rate, and get the best model by our training data, the model may not work well in predicting the unseen data in the feature, because the spread crossing rate is dependent with respect to time. That is the most crucial part which should be solved if people want to build real trading strategies.

## 8 References

A.Kercheva and Y.Zhang. Modeling high-frequency limit order book dynamics with support vector machines. October 2013.