Machine Learning Project on Financial Data
Stat 222, Spring 2016

Chenzhi Li, Zhuangdi Li, Ye Zhi
Github: https://github.com/lichenzhi/STAT222_Financial_Data

March 30, 2016

Abstract

We apply the machine learning algorithms support vector machines(SVM), random forests and gradient boosting on the high-frequency limit order book (LOB) data. Using the approach suggested by Kercheval and Zhang[4], we also acquire predication of mid-price movements and bid-ask spread crossings to implement trading strategies.

1 Introduction

High-frequency trading relies on the extraordinarily high-speed and advanced computer programs for producing, routing and carrying out orders. It has very short time-frames for establishing and liquidating positions and generates numerous trades on daily basis[2]. In this research project, we explore the LOB data and capture stock price movements at the millisecond scale. Our main task is to predict the mid-price movement and bid-ask spread crossings using machine learning techniques, including support vector machines, random forests and gradient boosting. After acquiring useful features for prediction and decide the suitable time horizon in terms of number of time-stamped trade events, we split the data set into two parts: 9:30 to 11:00 and 11:00 to 12:00 of the specific trading day. The data set from 9:30 to 11:00 is used for model fitting and model assessment, which is further split into two parts: training set (75%) and testing set (25%). We cross validate and fit the SVM, random forests and gradient boosting models after tuning the parameters for each machine learning algorithm with the training set. Then we test our models with the testing set and report the model performance using recall, precision and F_1-measure. We also come up with a couple of summary statistics beneficial for predicting the price movement. The summary statistics is used to fit a logistic regression and compare with the results acquired from the machine learning approaches. Finally, based on the machine learning models, we implement simple trading strategies on the 11:00-12:00 data and calculate the total profit.

The paper is structured as follows: Section 2: Data Preprocessing; Section 3: Model Fitting; Section 4: Model Assessment; Section 5: Interpretation; Section 6: Trading Strategies; Section 7: Conclusion.

2 Data Preprocessing

2.1 Creation of Feature Vector Sets

Inspired by SVM multi-class classifier by Kercheval and Zhang[4], we divide the predictors into three categories: the basic set, the time-insensitive set and the time-sensitive set. Table 1 summaries all the feature vectors we created using the LOB data. The basic set comprises 10 levels of prices and volumes of bid and ask. The best bid/ask price/volume are listed first, and then the next best second, etc. Attributes in the time-insensitive set are easily computed from the basic set at a single point in time. Of this, bid-ask spread and mid-price, price ranges, as well as average price and volume at different price levels are calculated in feature sets v2, v3, and v5, respectively; while v5 is designed to track the accumulated differences of price and volume between ask and bid sides. For the time sensitive set, we measure the time in terms of the number of time-stamped trade events. We can easily pass any number of time-stamped trade events as a parameter to the function. As for the feature vector v_6, we use $\triangle$t = 30 to compute the derivatives.

Table 1: Feature Vector Sets

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

Time-insensitive Set	Description(i=level index, n=10)
$v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^n$	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
$v_4 = \{\frac{1}{n}\sum_{i=1}^n P_i^{ask}, \frac{1}{n}\sum_{i=1}^n P_i^{bid}, \frac{1}{n}\sum_{i=1}^n V_i^{ask}, \frac{1}{n}\sum_{i=1}^n V_i^{bid}\}_{i=1}^n$	mean prices and volumes
$v_5 = \{\sum_{i=1}^n (P_i^{ask} - P_i^{bid}), \sum_{i=1}^n (V_i^{ask} - V_i^{bid})\}_{i=1}^n$	mean prices and volumes

Time-sensitive Set	Description(i=level index, n=10)
$v_6 = \{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\}_{i=1}^n$	price and volume derivatives

2.2 Creation of Labeled Responses

To study both mid-price movements and bid-ask spread crossings, we create the true label sets for every data point. The mid-price at time t is defined as the mean of the best ask price at time t and best bid price at time t, i.e. $P_t^{mid} = \frac{1}{2}(P_t^{ask} + P_t^{bid})$. The three possibility for mid-price movements are upward, downward and stationary. For example, compared to time t, the mid-price at time t+$\triangle$t increases, then it's an upward mid-price movements. This indicator provides potential trading opportunities to make profits to some extent.

There are also three possibility bid-ask spread crossings: upward, downward and stationary. An upward price spread crossing occurs if the best bid price at time t+$\triangle$t is greater than the best ask price price at time t, i.e. $P_{t+\triangle t}^{bid} > P_t^{ask}$. A downward price spread crossing take places if the best ask price at t+$\triangle$t does not exceed the best bid price at time t,i.e. $P_{t+\triangle t}^{ask} < P_t^{bid}$ A stationary spread crossing happens when $P_{t+\triangle t}^{ask} \geq P_t^{bid}$ and $P_{t+\triangle t}^{bid} \leq P_t^{ask}$

After obtaining the true label of each data point, we combine the feature vector sets and the label sets to create a nicely-formatted data frame, which will be used for fitting the machine learning models. For now, we have 332,673 rows and 128 columns, with 126 features and 2 labeled responses.

2.3 Splitting and Sampling Data

Based on the project requirement, we split the merged data set into modeling set and trading strategy set. We first extract the time variable out and divide the data set based on the hour. If the time is between 9:30 and 11:00, these transaction information is used for modeling, including training, validation and testing; if the time is from 11:00 to 12:00, these transaction information is used for trading strategy. The modeling set has 203,349 rows and the strategy set has 76,307 rows. To better fit, validate and test our machine learning models, we split our modeling set to 75% training and validation set and 25% testing set. Note in this project we use cross-validation technique to tune parameters. There are 152,511 rows for training and validating and 50838 rows for testing.

For computational simplicity, we write the sampling functions to take samples from the training data. There are two main ways to draw samples, sampling by population and sampling by proportion. The first one is based on mid-price movement and second one is based on another response variable: bid-ask spread crossing. The method of sampling by population method calculates the proportions of three values upward, downward and stationary in the training set and stratified sample them based on their original proportions correspondingly. Therefore, the proportions of upward, downward, stationary in our sample will be consistent with those in the training set. For the sampling by proportion method, we have a default proportion (1,1,2), which will sample different amounts of y response with values upward, downward, stationary correspondingly. The method also takes consideration of the cases when the number of y values in the training set is smaller than the desired amount; if this situation happens, we will use oversampling and set the replace argument to True. The oversampling is mainly used for the spread crossing, where the length of stationary response is quite different from the lengths of upward and downward. In this project, our sample size will be 30,000.

After data preprocessing, we will fit and tune different machine learning models in our training and validation set and compare the results based on testing set. Furthermore, we remove data points with NA values, resulted from derivative and responses.

3 Model Fitting

3.1 Overview

We use K-fold cross validation and grid search to tune parameters for each model. Cross validation is a statistical method of evaluating learning algorithms by splitting the data set into two segments. In k-fold cross validation, the data set is equally divided into k folds and k iterations of training and validation are performed. For each iteration, a different fold of data is held-out for validation and the remaining k - 1 folds are used for training. It can be applied in three contexts: performance estimation, model selection and tuning model parameters[3]. In this project, we use the 10-fold cross validation for tuning the machine learning model parameters, which can assist us in achieving the best learning and predicting results for each model.

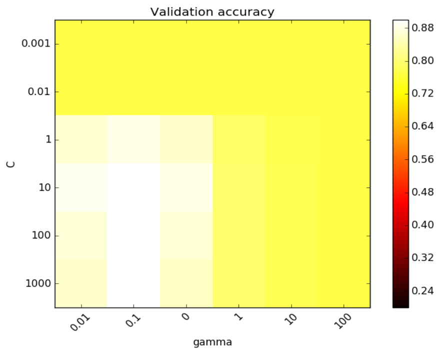
To obtain the best parameters for each machine learning model, we use cross-validated grid search to exhaustively consider all parameter combinations for the model. We provide the parameter grid which contains possible values for the parameters such that the cross-validated grid search can search all the possible combinations in the parameter grid, evaluate the performance of each combination via cross validation, and finally return the best parameter for the model learning models.

3.2 Support Vector Machines

Support vector Machines(SVM) are set of related supervised learning methods for classification and regression, which minimize the empirical classification error and maximize the geometric margin. SVM map the input vector into a higher dimensional space where the maximal separating hyper plane is constructed. Maximizing the distance between different parallel hyper planes, SVM come up with the classification of the input vector[4]. To achieve a better result of model fitting, several parameters should be tuned in the skicit learn packages.

- C, the penalty parameter of the error term. It trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly.
- kernel, the kernel type to be used in the algorithm. It must be one of linear, poly, rbf, sigmoid, precomputed or a callable. We decide to choose the commonly used (Gaussian) radial basis function ('rbf') kernel.
- gamma, the Kernel coefficient for rbf, poly and sigmoid'. It defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close.

Figure 1: Validation accuracy for C,gamma



From the heat map, the validation accuracy is encoded as colors varying from dark red to bright yellow. The highest accuracy concentrates at area of gamma=0.1, C=[10,1000]. After testing, we choose the best parameter C=10, gamma=0.1 and kernal='rbf' to fit and evaluate the model.

3.3 Random Forests

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) of the individual trees. The method combines Breiman's "bagging" idea and the random selection of features, correcting for decision trees' habit of overfitting to their training set[6]. For model fitting, there are several parameters we might need to tune in skicit learn packages.

- n_estimators, the number of trees in the forest. The larger, the more accurate. However, it takes considerable amount of computational time when increasing forest size. We choose 500 trees here to balance accuracy and computation simplicity.
- max_features, the number of features to consider when looking for the best split. This the parameter we tune from $\sqrt{n\_features}$, $log_2(n\_features)$, $\frac{1}{2}n\_features$, $\frac{1}{3}n\_features$, $\frac{1}{4}n\_features$, $\frac{1}{5}n\_features$, $\frac{1}{6}n\_features$, $\frac{2}{3}n\_features$,$\frac{3}{4}n\_features$, n_features. n_feature is 126 for our data set.
- max_depth, the maximum depth of the tree. The larger, the more accurate. However, it takes considerable amount of computational time when increasing the depth. To achieve a higher accuracy, we decide to use default value so that nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

To sum up, the only parameter we tune using cross validation is max_features. The best parameter is 11, which is $\sqrt{n\_features}$. To validate again, we apply our tuned random forest classifier with 500 n_estimators, 11 max_features and the largest max_depth to the whole training and validation data set with 152511 rows and compare the prediction labels and true labels. The overall precision and recall are both approach to 1 in terms of mid-price movement and bid-ask spread crossing.

3.4 Gradient Boosting

Gradient boosting is another machine learning algorithm for classification. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Gradient boosting fits an additive model in a forward stage-wise manner. In each stage, it introduces a weak learner to compensate the shortcomings of existing weak learners, which allows optimization of an arbitrary differentiable loss function[7]. The same as what we did for the random forest algorithm, we need to tune the parameters first when using the gradient boosting classifier function in the skicit learn package.

- n_estimators, the number of boosting stages to perform. Gradient boosting is fairly robust to overfitting, therefore, a larger number represents more performing stages, usually leading to better performance. We choose n_estimators to be 500, which will give us a relatively high accuracy and reasonable running time.
- learning_rate, learning rate will shrink the contribution of each tree by the value of learning_rate. There is a trade-off between learning_rate and n_estimators. After we set the number of trees, i.e, n_estimators, we use GridSearch to tune the learning_rate in order to find the best estimator.
- max_depth, the maximum depth of the individual regression estimators, which limits the number of nodes in the tree. We tune this parameter for better performance.

For gradient boosting, we mainly tune the learning_rate. We first set ten equally spaced values from 0 to 1 to find which one will lead to best performance. According to the best_estimator function, we find that the parameter is near 0.5555... Then we shrink our set range to (0.5,0.6) and run the gridsearch function again and find that the best learning_rate is 0.56 and the corresponding value for max_depth is 3. Therefore, we find that the best learning_rate and max_depth and apply them to the gradient boosting.

4 Model Assessment

4.1 overview

After tuning our parameters, we apply our Random Forest model to testing set with 126 features and compare the prediction labels with the true labels. There are mainly two ways to measure the quality of the prediction process, one is a confusion matrix and the other is percentage indicators including precision, recall, and F-1 measure. A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).[5] One important reason to introduce confusion matrix here is that we would like to see whether the model predicts oppositely or just to stationary when it fails to make the accurate prediction. For example, it will be more harmful for trading strategy if a model predicts upward than stationary when it actually acts downward.

Precision, recall and F-1 measure are defined below:

- precision(P) = $\frac{\#label\ y\ \ predicted\ correctly}{\#label\ y\ predicted}$
- recall(R) = $\frac{\#label\ y\ \ predicted\ correctly}{\#label\ y\ true}$
- F-1 = $\frac{2*P*R}{P+R}$

Thus, precision for each label is the corresponding diagonal value divided by row total in the confusion matrix and recall is the diagonal value divided by column total.

4.2 Support Vector Machines

For mid-price movement, the precision, recall and F-1 measure for each label, downward, upward and stationary are mostly above 75%, except for the recall and F-1 measure for stationary label. From the confusion matrix, we can see that the true label of stationary has lower proportion than the other two. This is due to the fact that we select 30 time stamped trading events. SVM does not have enough data to learn in the training set, resulting a low recall and therefore low F-1 measure. However, the performance of predicting downward and upward mid-price movement is pretty good.

Regarding the bid-ask spread crossings, most of the true label is stationary due to the fact that upward and downward bid-ask spread crossings are less frequent, however assure a profit if correctly identified. The SVM performance is quite well, with a very high prediction precision, recall and F-1 measure for all the three levels.

Table 2: SVM: Mid-Price Movement Confusion Matrix

True / Prediction	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	20726	1640	4923	27289
Stationary	297	2229	308	2834
Upward	2804	1048	16858	20710
ColTotal	23827	4917	22089	50833

Table 3: SVM: Mid-Price Movement Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Downward	0.759500	0.869854	0.810940
Stationary	0.786521	0.453325	0.575152
Upward	0.814003	0.763185	0.787775

Table 4: SVM: Bid-Ask Spread Crossing Confusion Matrix

True / Prediction	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	443	165	0	608
Stationary	73	49616	55	49744
Upward	0	132	347	479
ColTotal	516	49913	402	50831

Table 5: SVM: Bid-Ask Spread Crossing Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Downward	0.728618	0.858527	0.788256
Stationary	0.997427	0.994050	0.995735
Upward	0.724426	0.863184	0.787741

4.3 Random Forests

For mid-price movement, the precision, recall and F-1 measure for each label, downward, stationary and upward are mostly around 80%, except the recall rate for stationary. The problem is that random forest only predicts 2259 stationary events but actually that there are 4917. It seems that random forest is pretty aggressive that it tends to predict more downward and upward. The 80% sounds ideal so we have confidence to apply random forest to high frequency data. One concern is that when it misclassifies, it predicts oppositely in a considerable amount of time. For example, it predicts 3732 times downward when it is actually moving downward. So we would lose money when trading under this circumstance.

For bid-ask spread cross, the overall performance looks not as good as mid-price movement since the rate is mostly around 60% to 70%. However, a good sign here is that they almost never predict oppositely. Though the precision for downward is only 66%, but it only predicts 1 upward; the precision for upward is only 78%, but it never predicts downward. The reason of relatively low accuracy is that the model tends to predict stationary here, which will not cause losing money when trading without transaction fee.

Overall, random forests performs well when predicting mid-price movement and spread crossing.

Table 6: Random Forest: Mid-Price Movement Confusion Matrix

True / Prediction	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	20128	1613	3732	25473
Stationary	164	1936	159	2259
Upward	3535	1368	18198	23101
ColTotal	23827	4917	22089	50833

Table 7: Random Forest: Mid-Price Movement Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Downward	0.790170	0.844756	0.816552
Stationary	0.857016	0.393736	0.539576
Upward	0.787758	0.823849	0.805399

Table 8: Random Forest: Bid-Ask Spread Crossing Confusion Matrix

True / Prediction	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	366	187	1	554
Stationary	150	49669	196	50015
Upward	0	57	205	262
ColTotal	516	49913	402	50831

Table 9: Random Forest: Bid-Ask Spread Crossing Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Downward	0.660650	0.709302	0.684112
Stationary	0.993082	0.995111	0.994096
Upward	0.782443	0.509950	0.617470

4.4 Gradient Boosting

For mid-price movement, the precision, recall and F-1 measure for label downward and upward are around 70% while the precision for stationary is around 60%, recall and F-1 measure is even lower, around 30% to 40%. The reason is that the amount of stationary in our data is relatively small, compared to downward and upward. Gradient boosting predicts 2335 stationary events while there are 4917; the same as random forest, it tends to predict more downward and upward events. From the table, we can see that gradient boosting has good performance on correctly predicting the downward and upward events.

For bid-ask spread crossing, the overall performance flips around as that of the mid-price movement. We can see that although the recall rate for downward and upward is greater than 70%, their prediction precision is really low, only around 20%, i.e, only $\frac{1}{5}$ of our predictions for up and down is correct. On the contrary, the prediction accuracy for stationary event is quite high, which is up to 99%. Therefore, we can see that for spread crossing, the gradient boosting is conservative and it tends to predict more stationary events rather than upward or downward. When the time stamp is relatively small, gradient boosting will have good performance on predicting the bid-ask spread crossing.

Table 10: Gradient Boosting: Mid-Price Movement Confusion Matrix

Prediction \ True	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	17262	1675	4625	23562
Stationary	490	1387	458	2335
Upward	6075	1855	17006	24936
ColTotal	223827	4917	22089	50833

Table 11: Gradient Boosting: Mid-Price Movement Measures

Prediction \ True	**Precision**	**Recall**	**F-1 Measure**
Downward	0.732620	0.724472	0.728523
Stationary	0.594004	0.282083	0.382515
Upward	0.681986	0.769885	0.723275

Table 12: Gradient Boosting: Bid-Ask Spread Crossing Confusion Matrix

Prediction \ True	**Downward**	**Stationary**	**Upward**	**RowTotal**
Downward	402	2040	10	2452
Stationary	99	46803	69	46971
Upward	15	1070	323	1408
ColTotal	516	49913	402	50831

Table 13: Gradient Boosting: Bid-Ask Spread Crossing Measures

Prediction \ True	**Precision**	**Recall**	**F-1 Measure**
Downward	0.163948	0.779070	0.270889
Stationary	0.996423	0.937692	0.966166
Upward	0.229403	0.803483	0.356906

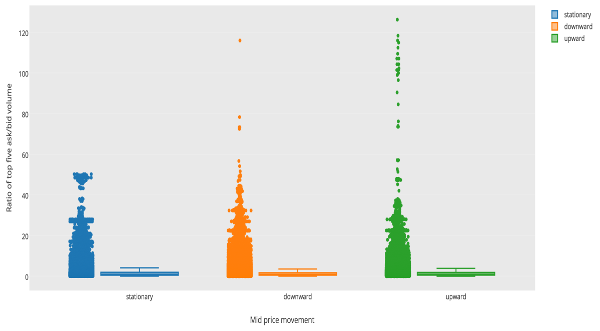
4.5 Comparison

Based on our previous model assessment, we can see that for mid-price movement, SVM and random forest has similar performance. Random forest predicts downward and stationary events more accurately while SVM predicts upward event a little bit more accurate than random forest. For bid-price spread crossing, we can see that all three algorithms are good at predicting stationary events; among these three, gradient boosting is the least conservative one. The values on for upward and downward on the anti-diagonal are relatively large compared with the other two algorithm; therefore, we will not use gradient boosting to predict the spread crossing in strategy data set.

5 Interpretations

The key concept that underlies the logistic regression is the logit, i.e. the natural logarithm of an odds ratio. In this section, we encode the responses in labeled set into binary variables and predict the price movement based on the predictors. One set of the predictors we use is the 126 features we created for machine learning models plus the top five volumn ratio and the other set is the 10 predictors we acquire from the principal component analysis (PCA) of the 126 features. The prediction results from both sets of predictors are not as good as those of machine learning models, as shown in the tables. We also examine the odd ratios for both sets of predictors. Almost all the odd ratios are close 1, except that the odd ratio of difference between best ask price and best bid price to predict stationary/non-stationary spread crossing, which is around 2. It indicates that for a one-unit increase in the difference between best ask price and best bid price, we expect to see about 100% increase in the odds that spread crossing stays stationary. This conclusion matches what we get from the random forest algorithm where difference between best ask price and best bid has a relatively high importance.

Figure 2: Boxplot for top five volume ratio



The ratio is computed by the total volumes of top five ask-side orders divided by the total volumes of top five bid-side orders.The boxplot is colored by different labeled responses for mid-price movement. We can see that only the upward and downward mid-price movement has ratio exceeding 60, indicating that the imbalance of supply and demand will cause price movements.

Table 14: Logistic Regression on 126 features:Mid-price Movement Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Upward	0.577906	0.309982	0.403520
Non-upward	0.610171	0.826700	0.702121
Stationary	0.553763	0.021059	0.040575
Non-stationary	0.905465	0.998193	0.949571
Downward	0.589744	0.713973	0.645939
Non-downward	0.577455	0.440405	0.499703

Table 15: Logistic Regression on 10 PCA components:Spread Crossing Measures

True / Prediction	**Precision**	**Recall**	**F-1 Measure**
Upward	0.007909	1	0.015693
Non-upward	0.992091	1	0.996030
Stationary	0.981232	1	0.990527
Non-stationary	0.018768	1	0.036845
Downward	0.01086	1	0.021486
Non-downward	0.98914	1	0.994540

6 Trading Strategies

6.1 Data and Trading Strategy

For trading strategy, we use data from 11:00 - 12:00 and 126 features from sophisticated machine learning techniques. Recall the definition of bid-ask spread crossing. If there is an upward spread cross, the best bid price at $t + \triangle t$ will be higher than the best ask price at time t. Thus we will obtain profit if we buy at best ask price at time t and sell at best bid price at time $t + \triangle t$ when we observe an upward spread cross at time t. Similarly, if there is a downward spread cross, the best ask price at $t + \triangle t$ will be higher than the best bid price at time t. Thus we will obtain profit if we short at best bid price at time t and buy back at best ask price at time $t + \triangle t$. Mid-price movement cannot guarantee profit.

Using this trading strategy, our main task is to predict spread crossing based on 126 features using machine learning techniques. We decide to try Support Vector Machines with C=10 and Gamma = 0.1 and Random Forests with 500 estimators and 11 max_features based on the previous modeling fitting and assessment since these two models perform better than gradient boosting at 9:30-11:00. However, in 30 time stamped event, neither SVM and Random Forests predict upward and downward spread crossing. To enlarge $\triangle t$ reasonably, we decide to use 50 time stamped event. If the time frame is too large, the prediction accuracy decreases dramatically. However, SVM still predicts only 2 downward movement and 0 upward movement. Therefore, we later only use Random Forests to predict spread cross.

6.2 Profit

The first step is to check the profit using this trading strategy if every spread crossing label is predicted correctly. The profit from longing stocks is 20.8 and the profit from shorting stocks is 25.11. So the maximum profit in 50 time stamped event is 45.91.

we use predicted label to determine our trading behavior, however, The profit from longing stocks is -36.07 and the profit from shorting stocks is -60.19 and the total profit is -96.26. Therefore, we will lose 96.26 dollars if we trade based on the Random Forests model we trained in 9:30-11:00.

6.3 Discussion

The negative profit results from the low accuracy of machine learning models in strategy set(11:00 - 12:00). The precision of downward is only 0.1 and upward is 0.002 even though the total precision is 0.97 since the majority is stationary, which is easy to predict. Though models perform well in testing set in 9:30-11:00, all about 70% - 80% in precision, they do not produce satisfactory results in 11:00-12:00, our strategy time frame. Therefore it is very difficult to achieve high back testing accuracy using the machine learning techniques described and feature sets selected in this paper. In the future, we might need to include more features, for instance, time-sensitive sets and consider time-series techniques, since the result from back testing convinces us that we need to capture stock performance in different time frame.

Furthermore, assume that we can improve our model accuracy in strategy data set in the future, we still need to consider transaction fee in real life. Our thinking is to reclassify the spread cross label. Only if the spread is above the transaction fee, instead of just zero, we will determine that it is an upward or downward spread cross. Therefore, our revenue obtained from longing or shorting stocks will cover transaction fee so that we will make positive profit.

7 Conclusion

Among the three machine learning algorithms we applied, random forest has the best performance on our testing dataset. However, its prediction for spread crossing on the trading strategy set is not as good as we expect. Without certain time sensitive features, it might be difficult to predict spread crossing changes over time and make profits from it.

References

[1] Kercheval, Alec N., and Zhang, Yuan. *Modeling high-frequency limit rder book dynamics with support vector machines.*

[2] Jones, Charles M. *What do we know about high-frequency trading?.*

[3] Refaeilzadeh, Payam, Tang, Lei and Liu, Huan. *Cross-Validation*

[4] Srivastava, Durgesh K. and Bhambhu, Lekha. *Data Classification using support vector machine*

[5] Powers, David M W. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation*

[6] Ho, Tin Kam. *Random Decision Forests.*

[7] Li, Cheng. *A General Introduction to Gradient Boosting.*