Short-term Volatility Prediction

- STA2536 Final Project

Team Member: Yuling Wang, Zhichun Kang, Yuzhou Liu

Student Number: 1006715397, 1003051966, 100702486

Abstract

Stock market investment is just another synonym for ‘risk and return’, the accurate and timely prediction of market volatility remains one of the most challenging problems. In this project, we are given features for the evolution of the limit order book over one-year period for 6 stocks to explore machine learning methods in volatility classification problem. We develop an experimental framework to predict next volatility label. First, we explore the distribution of these features, as well as the diurnal pattern of the volatilities of each stock. Meanwhile, we will explore the effectiveness of five different models in predicting future volatility labels based on past information. Among all the models we built, XGBoost turns out to be the most effective one, which is selected for further prediction and test.

Contents

Abstract 2

Introduction 4

Methodology 5

2.1 Multi-Class Logistic Regression Model 5

2.2 K-Nearest-Neighbors Model 6

2.3 XGBoost 6

2.4 Neural Network 6

2.5 Long Short-Term Memory 8

2.6 Model Evaluation Metrics 8

Data Analysis 9

3.1 Data Cleaning 9

3.2 Diurnal Pattern of volatility labels 9

3.3 Summary Features 10

Modelling 13

4.1 Training data, validation data and test data: 13

4.2 Results: 13

Conclusion 20

Appendix 21

Introduction

In financial market, many investors are concerned with the volatility of a stock before they make their investment decisions. And the stock price volatilities are partially affected by past information in the market, including the evolution of the information in the limit order book. Therefore, investors are also interested in predicting the future volatilities using past information.

In this project, we are given features for the evolution of the limit order book over one-year period for 6 stocks. We will explore the distribution of these features, as well as the diurnal pattern of the volatilities of each stock. Meanwhile, we will explore the effectiveness of five different models in predicting future volatility labels based on past information.

Methodology

2.1 Multi-Class Logistic Regression Model

The basic setting of a multi-class logistic regression model is:

$$P(Y = y) = \frac{e^{\omega_y^T x}}{\sum_{y' \epsilon y} e^{\omega_{y'}^T x}}$$

where $\omega_1 = 0$ so that the model is identifiable. And the likelihood function can be given by:

$$L\big((\omega_c)_{c\epsilon y}\big) = P\big((Y_i = y_i)_{i\epsilon N} \big| (\omega_c)_{c\epsilon y}, (X_i = x_i)_{i\epsilon N}\big)$$

$$= \prod_{i\epsilon N} P\big(Y_i = y_i \big| (\omega_c)_{c\epsilon y}, X_i = x_i\big)$$

$$= \left(\prod_{i\epsilon N, y_i=1} \frac{e^{\omega_1^T x_i}}{\sum_{c\epsilon y} e^{\omega_c^T x_i}}\right)\left(\prod_{i\epsilon N, y_i=2} \frac{e^{\omega_1^T x_i}}{\sum_{c\epsilon y} e^{\omega_c^T x_i}}\right) \dots \left(\prod_{i\epsilon N, y_i=n} \frac{e^{\omega_n^T x_i}}{\sum_{c\epsilon y} e^{\omega_c^T x_i}}\right)$$

Moreover, we can simplify the expression above by introducing vectors W and y_i as follows:

$$W = \begin{bmatrix} \omega_1^T \\ \omega_2^T \\ \dots \\ \omega_n^T \end{bmatrix} \quad y_i = \begin{bmatrix} \mathrm{I}_{\{y_i=1\}} \\ \mathrm{I}_{\{y_i=2\}} \\ \dots \\ \mathrm{I}_{\{y_i=n\}} \end{bmatrix} \quad i\epsilon N$$

The likelihood and log-likelihood function can be written with these notations as follows:

$$L(W) = \prod_{i\epsilon N} \frac{e^{y_i^T W x_i}}{\sum_{c\epsilon y} e^{\omega_c^T x_i}}$$

$$l(W) = \sum_{i\epsilon N} \{y_i^T W x_i - log \sum_{c\epsilon y} e^{\omega_c^T x_i}\}$$

To obtain an analytical formula of the MLE, we use the Newton-Raphson method to get a sequence of improved estimates of the model. To achieve this goal, we define the Score Function $S(W)$ and Fisher Information $I(W)$ as follows:

$$[S(W)]_{pq} = \partial_{W_{pq}} l(\omega) \quad [I(W)]_{pqrs} = -\partial_{W_{pq}W_{rs}} l(\omega)$$

We further define another (n*1)-vector valued function:

$$[\mu_i(W)]_p = \frac{e^{\omega_c^T x_i}}{\sum_{c\epsilon y} e^{\omega_c^T x_i}}, \quad p\epsilon y$$

$S(W)$ and $I(W)$ can be expressed as follows:

$$[S(W)]_{pq} = \sum_{i \epsilon N} \left([y_i]_p - [\mu_i(W)]_p\right)[x_i]_q$$

$$[I(W)]_{pqrs} = \sum_{i \epsilon N} \left(\delta_{pr} - [\mu_i(W)]_p\right)[\mu_i(W)]_r [x_i]_s [x_i]_q$$

At the same time, the estimation of the log-likelihood $W^{(k)}$ can be updated by the following rule:

$$\widehat{W}^{(k+1)} = \widehat{W}^{(k)} + \left[I\left(\widehat{W}^{(k)}\right)\right]^{-1} S(\widehat{W}^{(k)})$$

When the difference between $\widehat{W}^{(k+1)}$ and its previous value falls in the pre-determined tolerance, the algorithm will stop. And we usually set the tolerance to be 10^{-6}.

2.2 K-Nearest-Neighbors Model

The principle behind the nearest-neighbors method is to find a predetermined number of points in feature space that are closest to the given point in distance. After finding these points, we simply select the label of the majority of the nearest-neighbors and assign it to the given point.

The k-nearest-neighbors model is mainly dependent on two parameters. The first parameter is k, which is the number of nearest neighbors taken into consideration. And the second parameter is the counting method, which can be uniform or distance. Setting the parameter to be uniform means each nearest neighbor is counted equally, regardless of the distance between the neighbor and the given point. And the distance counting method calculate the weighted count of the labels of the nearest neighbors using the distance as weight.

2.3 XGBoost

The XGBoost (eXtreme Gradient Boosting) is an ensemble algorithm based on gradient boosted decision trees. It combines predictions of weak classifiers (such as naïve tree model) to achieve a strong classifier via a serial training process. The XGBoost algorithm introduces a regular term in the base learner loss function. Also, it not only applies the first derivative to calculate the pseudo-residue, but also calculates the second derivative to approximate the first pruning. The objective loss function can be defined as:

$$L = \sum_{i=1}^{n} l(yi, yı) + \sum_{K=1}^{K} \Omega(f_k)$$

2.4 Neural Network

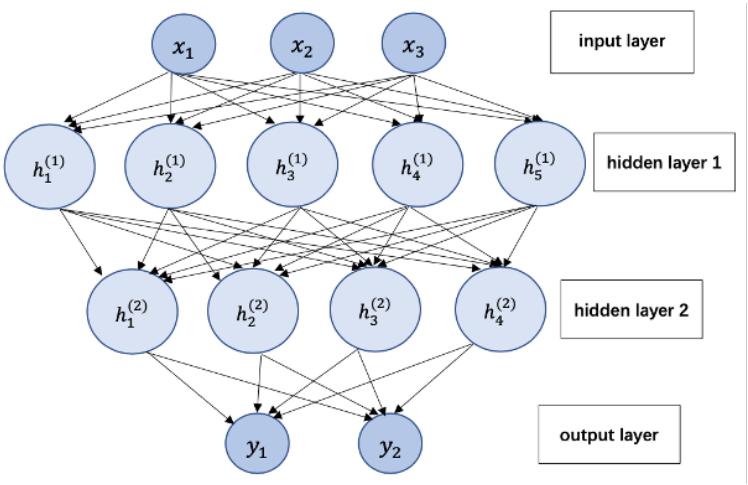
Artificial Neural Nets are the computing systems that comprise of a collection of

connected nodes. They deterministically map: `

$$F: \mathcal{X} \to \mathcal{Y},$$

from an input space X into an output space Y.

In our classification problem, we have labeled training examples (x, y). Our fully connected neural network is constructed as follow:



Where:

- Propagation function defined as an affine transformation:

$$h^{(1)} = g\{(w^{(1)})^T x + b^{(1)}\}$$
$$h^{(2)} = g\{(w^{(2)})^T h + b^{(2)}\}$$

- Leaky ReLU (Leaky Rectified Linear Unit) activation function defined as:

$$g(x) = \begin{cases} x, & x > 0 \\ \epsilon x, & x \leq 0 \end{cases}, \text{ where } \epsilon \ll 1.$$

- The output function is the hidden-to-output linear function itself:

$$y = (w^{(3)})^T h^{(2)} + b^{(3)}$$

- The Loss function is given by the Cross-entropy Loss in the optimization:

$$L(X, Y) = -\sum_{i \,\epsilon\, data} log \frac{e^{f_{y_i}(x_i)}}{\sum_k e^{f_k(x_i)}}$$

During our training part, Stochastic Gradient Descent (SGD) is used to compute gradient. In each time, a random minibatch of N training examples was sampled and update all parameters in the net system through backpropagation:

$\Theta \leftarrow \Theta - \eta \nabla_\theta L$, where η refers to the learning rate

After iteration, we could obtain the optimal parameters and further the learnt model. In addition, we used a simple grid search to determine the appropriate number of lag and node and minibatch size.

2.5 Long Short-Term Memory

Long Short-Term Memory model is a Recurrent Neural Network (RNN) with memory functions and forget gates so that overall meaningful information is extracted by the model from noisy data. An LSTM model contains many LSTM units, each of which contains an input gate i, an output gate *o*, a forgetting gate *f* and a memory unit *c*.
At time t, given the input vector x_t and the hidden state h_{t-1} of the previous moment, the LSTM unit calculates the implicit state h_t of the current moment by internal looping and updating:

$$i_t = \sigma(U^{i^T} x_i + W^{i^T} h_{t-1} + b^i)$$

$$f_t = \sigma(U^{f^T} x_i + W^{f^T} h_{t-1} + b^f)$$

$$o_t = \sigma(U^{o^T} x_i + W^{o^T} h_{t-1} + b^o)$$

$$c_t = f_t^T * c_{t-1} + i_t^T * \phi(U^c x_i + W^c h_{t-1} + b^c)$$

$$h_t = \left(\phi(c_t)\right)^T * o_t$$

Where the parameter set $\{U^i, U^f, U^o, U^c, W^i, W^f, W^o, W^c\}$ corresponds to the weight matrix of different gates, $\{b^i, b^f, b^o, b^c\}$ denotes the corresponding offset term, and σ and ϕ are respectively sigmoid. And the activation function is ReLU.

2.6 Model Evaluation Metrics

2.6.1 Accuracy

Accuracy score represents how correct the model can predict. It is given by:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

2.6.2 Recall

Recall rate shows how many are cases have been classified correctly in the whole positive cases. It can be considered as a measure of how complete a model can predict. A low recall indicates there are many false negatives. Recall can be thought of as a measure of a classifier completeness. A low recall indicates many False Negatives. It is defined as:

$$Recall = \frac{True\ Positive}{True\ Positive + \ False\ Negative}$$

2.6.3 F1-score

F1-score conveys the balance between model's precision and recall. It is defined as:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where,

$$Precision = \frac{True\ Positive}{rue\ Positive + \ False\ Positive}$$

Data Analysis

3.1 Data Cleaning

In the dataset, we are provided with limit order book information of 6 stocks on 1334 days in total. Exploring the structure of the data, we found that the information on 2018-07-03, 2018-11-23, 2018-12-24 are all incomplete, which was caused by the early close of the market. Therefore, we deleted the data on these days to make the size of data on each day comparable, which left us with data of 1320 days. Meanwhile, the last second of each day contains all zero or NA values, which means they are missing data, so we also delete the data in the last second from each day.

3.2 Diurnal Pattern of volatility labels

For a given time period, different types of measures can be used to represent the volatility in its price movement. In this part, we will discuss two methods of volatility measures and plot the diurnal pattern of the volatilities of different measures.

3.2.1 Volatility Measures

When measuring the volatility of the price of a stock, we focus on the changes in its mid-price, which is the average of the best bid and best ask price. In each 5-minute period, we decide to use the standard deviation and range of the mid-price as the volatility measures since they all depict the fluctuation in stock prices.

3.2.2 Diurnal Pattern

Everyday each stock has 78 5-minute periods in the trading hours. Firstly, we calculate the 1/5, 2/5, 3/5, 4/5 quantile of each stock at each time period of day. To smooth the quantile curve for each stock, we adopt the polynomial regression using time scaled between -1 and 1 as independent variables and each quantile as dependent variables. The smoothed diurnal pattern of standard deviation of all the 6 stocks are shown in figure 3.2.2. The diurnal patterns of the volatility of AAL, FB, GOOG and INTC are similar, of which the volatilities are higher in first 10 5-minute periods and remain stable afterwards. At the same time, the differences between each quantile are also similar across all quantiles.

We can also notice that the 1/5 and 2/5 quantile of SIRI overlap between the 30^{th} and the 70^{th} period, which is caused by the stable price in some days of these periods. And for VOD, two significant descending in its volatility can be found in the plot. The diurnal pattern of the ranges of all stocks can be found in the appendix.

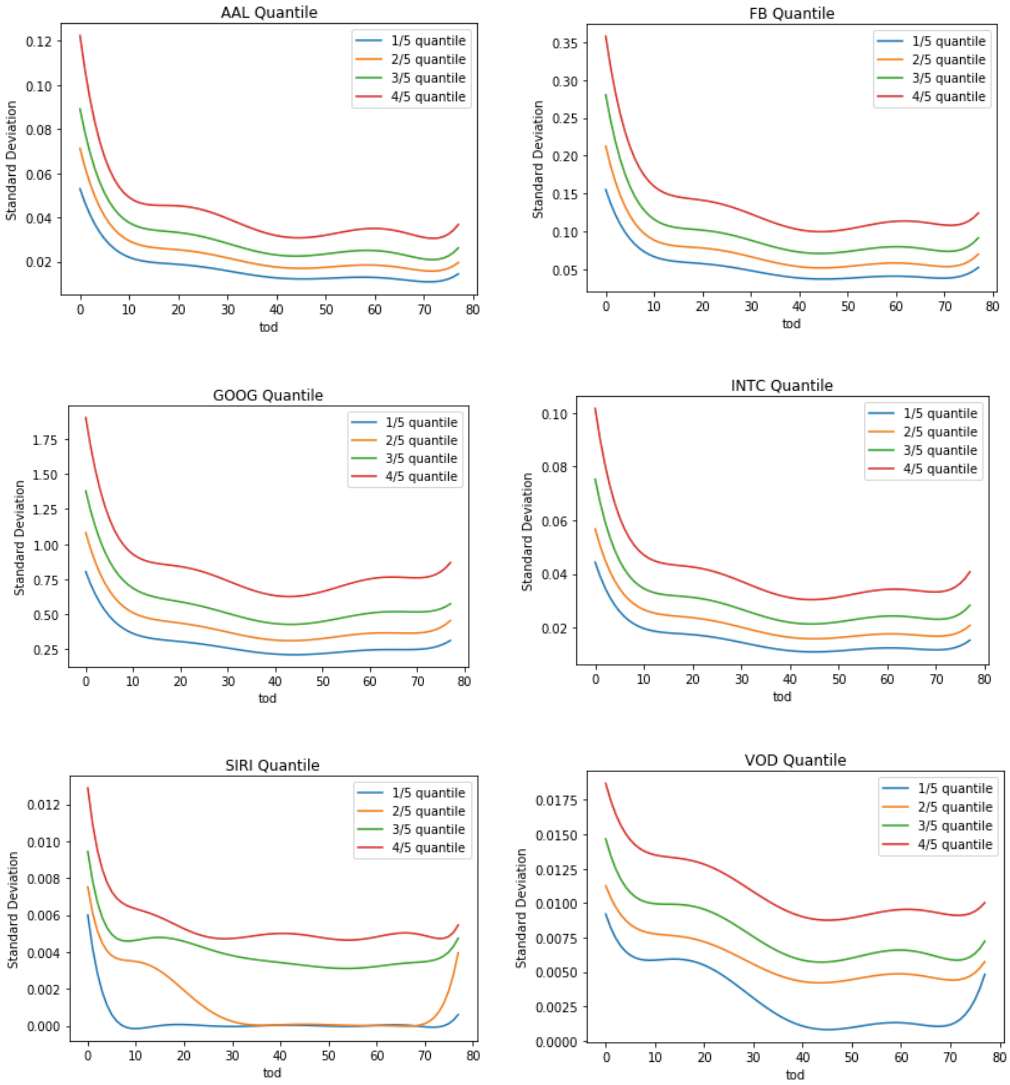


Figure 3.2.2 Smoothed diurnal pattern of standard deviation of all the 6 stocks

3.3 Summary Features

In this part, we use the standard deviation of mid-price in 5 minutes as the volatility in this time period. To predict the labels of future volatilities using past information, we create some summary features based on features provided in the original dataset. The summary features are described as follows:

- **Event Time:**
  The time of day, ranging from 0 to 77.
- **Sum Limit Buy Price:**
  The sum of all the 20*300 limit order buy prices in 5 minutes.
- **Sum Limit Buy Volume:**
  The sum of all the 20*300 limit order buy prices in 5 minutes.
- **Sum Limit Sell Price:**

The sum of all the 20*300 limit order sell volumes in 5 minutes.

- **Sum Limit sell Volume:**
  The sum of all the 20*300 limit order sells volumes in 5 minutes.
- **Max Limit Sell Price:**
  The maximum limit order sell price in 5 minutes.
- **Min Limit Buy Price:** T
  The minimum limit order buy price in 5 minutes.
- **Sum Market Buy Count:**
  The sum of all the 300 markets order buys counts in 5 minutes.
- **Sum Market Sell Count:**
  The sum of all the 300 markets order sell counts in 5 minutes.
- **Sum Market Buy Volume:**
  The sum of all the 300 markets order buys volumes in 5 minutes.
- **Sum Market Sell Volume:**
- The sum of all the 300 markets order sells volumes in 5 minutes.
- **Market Buy Volume std:**
  The standard deviation of the market buys volumes in 5 minutes.
- **Market Sell Volume std:**
  The standard deviation of the market sells volumes in 5 minutes.
- **Average Spread:**
  Average of the difference between the best bid and best ask prices in 5 minutes.
- **Limit Order Volume Difference:**
  Firstly, we sum the difference between limit order sell volume and buy volume at each second. Then we take the absolute value of the difference at each second and sum them to calculate the limit order volume difference.
- **Previous Label:**
  The label in the previous 5-minute period.
- **Stock Type:**
  A categorical random variable representing the type of the stock, ranging from 0 to 5.

The Figure 3.3 below represents the kernel density of empirical distribution of each feature. We can firstly notice that features except spread and volume difference for all stocks have very similar distributions. Also, approximate distributions also exist between features of buy and sell side, and between limit and market. These can provide important information of the market.

Based on these features, we explore the effectiveness of different models in predicting future volatility labels. For each model, we aim at using the information in the past 5-minute periods to predict the volatility label of next 5 minutes.

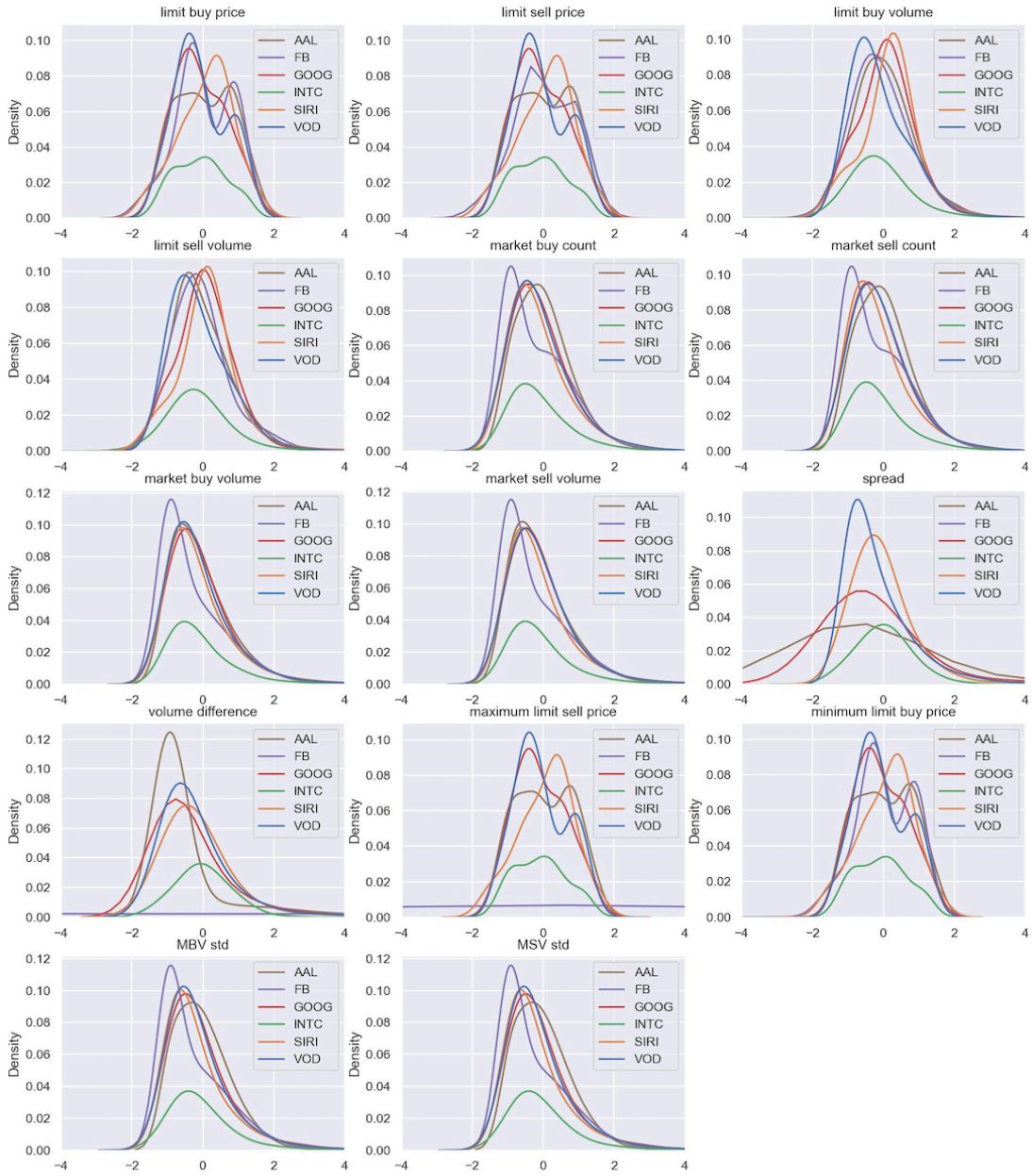


Figure 3.3 Kernel Density of empirical distribution of each feature

Modelling

4.1 Training data, validation data and test data:

We are provided with 1320 days of data after cleaning the original dataset. To make the model training process effective, we select 10% of the days from each stock as test data, which means 133 days are selected and all the data in these days are considered test data. The rest of the data are split into 5 folds randomly using the same method, making sure each stock is evenly distributed across all folds for the effectiveness of cross validation.

4.2 Results:

- Multi-Class Logistic Regression Model:

To build a 5-class logistic regression model, an essential part is to find the optimal regularization strength parameter C that makes model prediction accuracy the highest. Firstly, we focus on the model that includes all features. Applying 5-fold cross validation, we calculate the average accuracy score across all folds for models with C ranging from 0.05 to 0.95, which is shown in figure 4.2.1. It's obvious that the model has the highest average accuracy when C equals 0.35, where the average accuracy score is 30.908%. Utilizing this model to predict the labels on test data, the accuracy score, recall score and f1 score are 31.02%, 29.83% and 23.88% respectively. At the same time, the confusion matrix is shown in figure 4.2.2. We can see that most data are classified into label 0 or 4, which makes the prediction accuracy rather low.

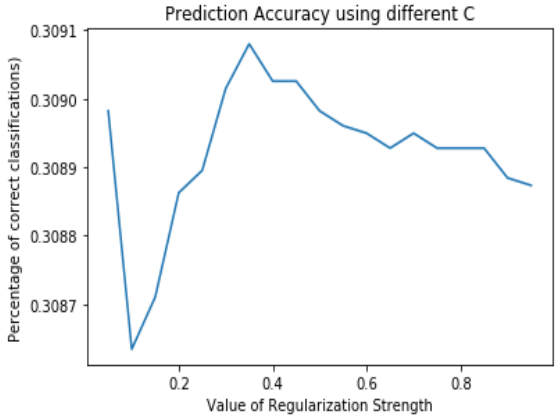


Figure 4.2.1 Prediction Accuracy using different C

Logistic	Accuracy	Recall	F1
Fold 1	31.02%	30.01%	24.01%
Fold 2	30.78%	29.81%	23.85%
Fold 3	30.98%	29.95%	23.95%
Fold 4	30.65%	29.75%	23.81%
Fold 5	30.92%	29.93%	23.93%
Test	31.02%	29.83%	23.88%

Table 4.2.1 Logistic Regression Prediction metric across all folds

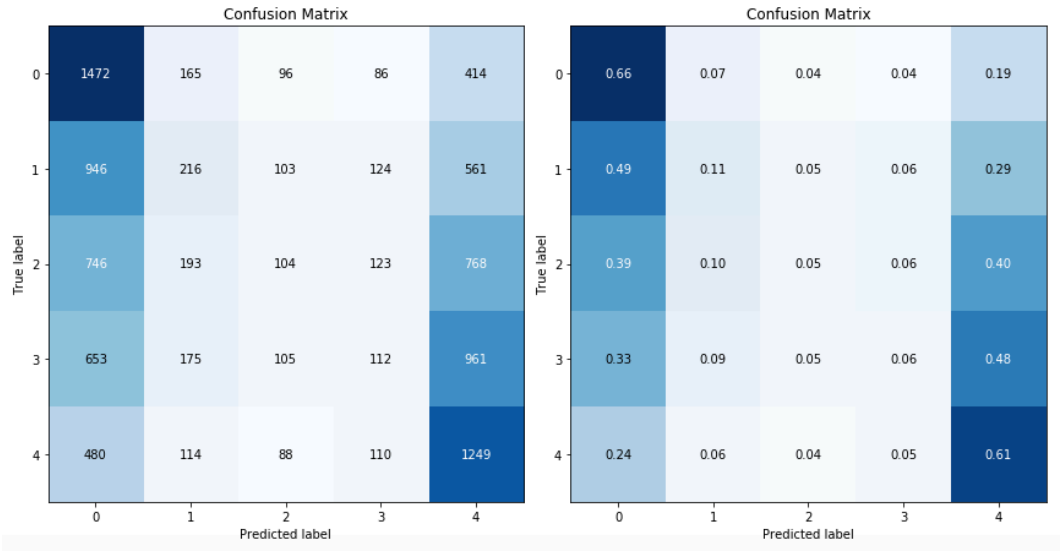


Figure 4.2.2 Confusion Matrices for Multi-Class Logistic Regression Model

Besides, we also explore the prediction accuracy of models with different combinations of different features by recursively eliminating the number of features in the model and calculate the 5-fold cross-validated prediction accuracy. The prediction accuracies of optimal models with different numbers of features are shown in figure 4.2.3. We can find that the model with all 16 features has the highest prediction accuracy. Therefore, we select this model as our preferred Multi-Logistic Regression Model.

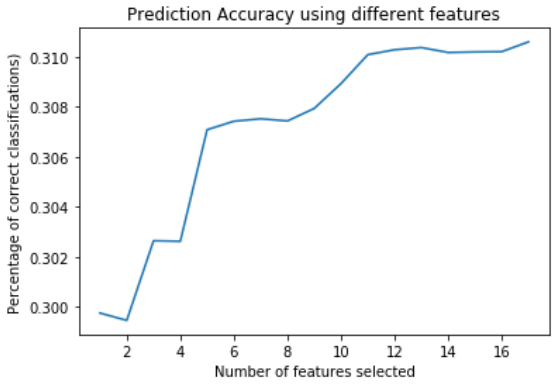


Figure 4.2.3 Prediction Accuracy using different Features

- K-Nearest-Neighbors Model:

The distance counting method in k-nearest-neighbors model makes more sense in this case because it takes the distance between the neighbors and the given point into account and gives more weights to nearer neighbors. Using this method, we conduct 5-fold cross-validation for models with different numbers of nearest neighbors and calculate the average prediction accuracy, recall scores and f1 scores across all folds for each model, which are shown in figure 4.2.4 and 4.2.5. The plots show that accuracy scores and recall scores keep increasing as number of nearest neighbors rise and show no trend when the number is bigger than 100. While the f1 score plot is like a bell shape, where the score reaches the highest when number of nearest neighbors is between 50 and 100. Considering these metrics and simplicity of the model, we select the model using 100 nearest neighbors. The accuracy score, recall score and f1 score of this model on training data are 32.41%, 31.74%, 30.07% respectively. And these metrics on test data are 33.73%, 31.17%, 31.11%. From the test data confusion matrix in figure 4.2.7, we can also find that most datapoints are classified into right labels.

KNN	Accuracy	Recall	F1
Fold 1	31.76%	31.14%	0.2964
Fold 2	30.38%	29.81%	0.2860
Fold 3	32.65%	31.95%	0.3004
Fold 4	32.49%	31.79%	0.2983
Fold 5	32.04%	31.40%	0.2976
Test	31.86%	31.17%	0.3111

Table 4.2.2 KNN Prediction metric across all folds

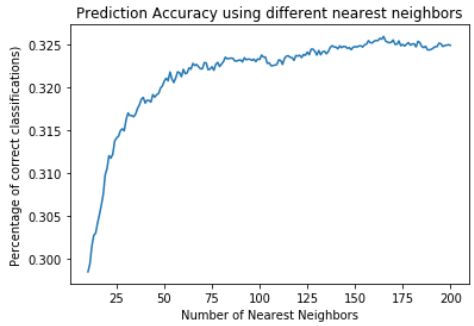


Figure 4.2.4 Prediction Accuracy using different nearest neighbors

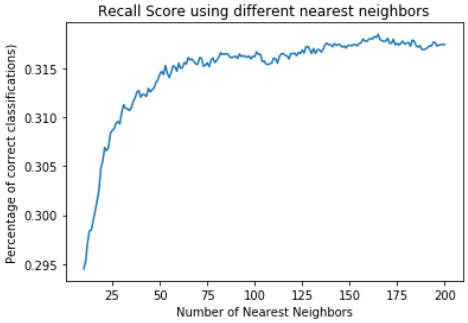


Figure 4.2.5 Recall Score using different nearest neighbors

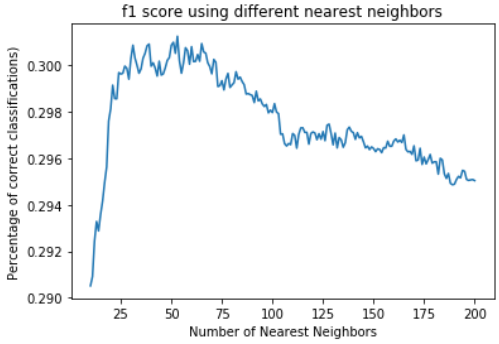


Figure 4.2.6 f1 score using different nearest neighbors

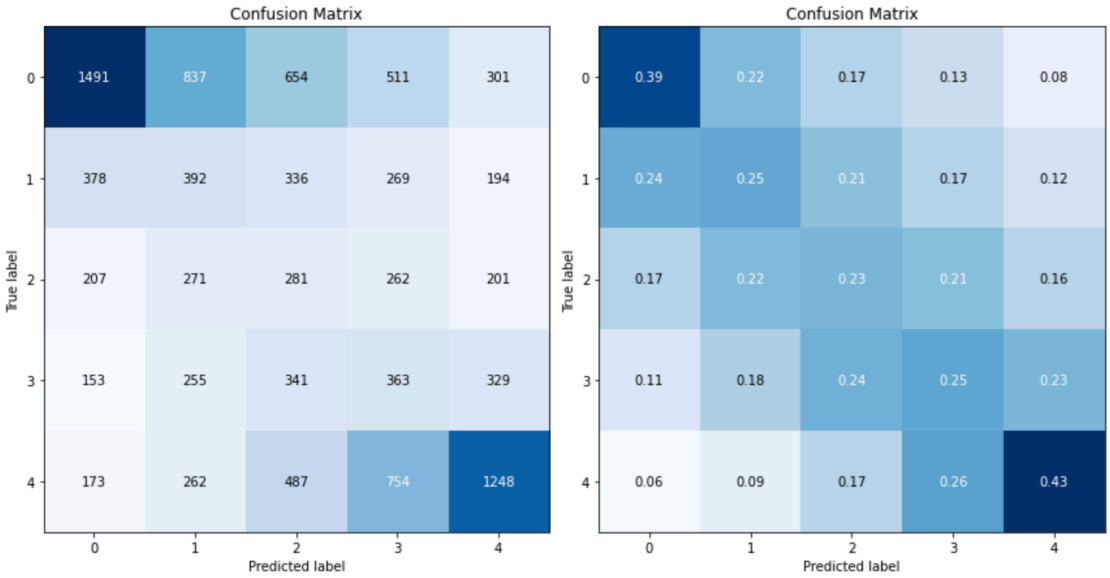


Figure 4.2.7 Confusion Matrices for K-Nearest-Neighbors Model

- XG Boost Model:

We firstly used grid search with 5-fold cross validation to tune hyperparameters. Important parameters involved in this report are as follows:

■ 'max_depth': the maximum depth of the base tree model, with higher values for more complicated base-tree models;
■ 'n_estimators': the number of base tree models, with higher values for more iterations;
■ 'min_child_weight': the minimum sum of child node weights, with higher values for more conservative models.

We also considered the number of lags in the data by looping over a set of potential lag numbers. Subject to computational capacity, other hyperparameters of the model used the default values. We finally obtained the optimal parameters as max_depth = 5, n_estimators = 125, and min_child_weight = 1.

The following tables show the statistical metrics of XGBoost on the cross validation and test set using optimal parameters. The confusion matrix is also shown in the picture. We can see that the model tends to predict label 0 or 4. The model fails to catch the minute differences between labels.

XGBoost	Accuracy	Recall	F1
Fold 1	35.66%	34.99%	0.3334
Fold 2	33.87%	33.26%	0.3209
Fold 3	35.69%	34.88%	0.3253
Fold 4	35.71%	34.91%	0.3270
Fold 5	35.15%	34.41%	0.3237
Test	37.14%	33.74%	0.3366

Table 4.2.3 XGBoost Prediction metric across all folds

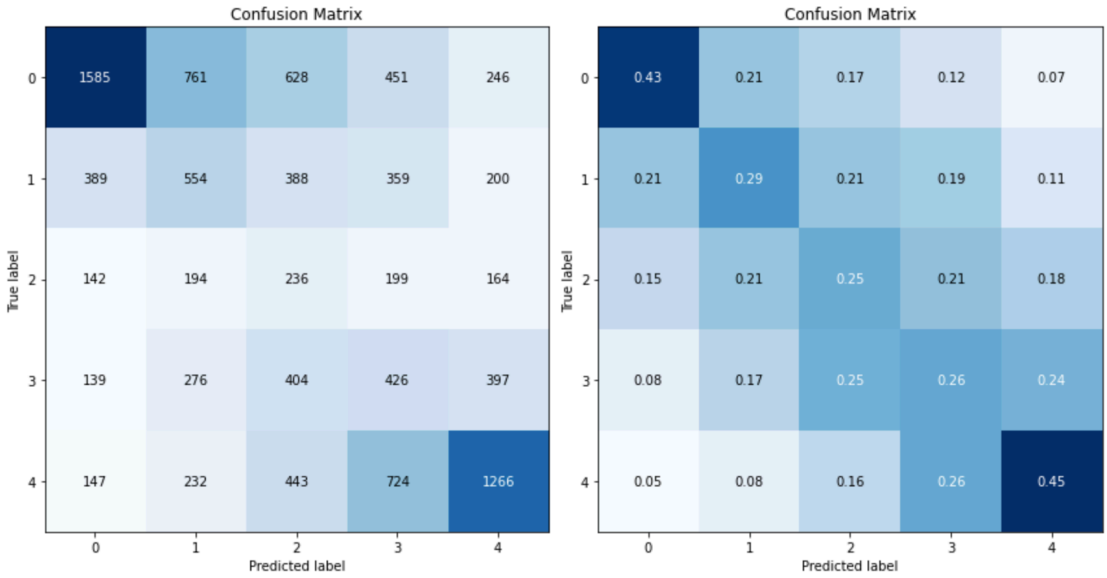


Figure 4.2.8 Confusion Matrices for XG Boost Model

- Neural Network Model:

We firstly performed grid search for different combinations of parameters to select the features and pick the suitable values of hyperparameters. From the model of best performance, the number of lags has been set to 5, size of minibatch set to 32, and the epochs set to 100. All features have been included throughout the whole process. The confusion matrix of test set has shown a similar predicting situation as in XGBoost. But neural network would have a lower recall and F_1, indicating more cases have been incorrectly classified as wrong classes.

Neural Network	Accuracy	Recall	F1
Fold 1	31.92%	31.08%	0.2734
Fold 2	30.89%	30.87%	0.2659
Fold 3	31.93%	30.65%	0.2649
Fold 4	32.57%	31.32%	0.2860
Fold 5	32.27%	31.19%	0.2824
Test	33.74%	29.77%	0.2912

Table 4.2.4 NN Prediction metric across all folds

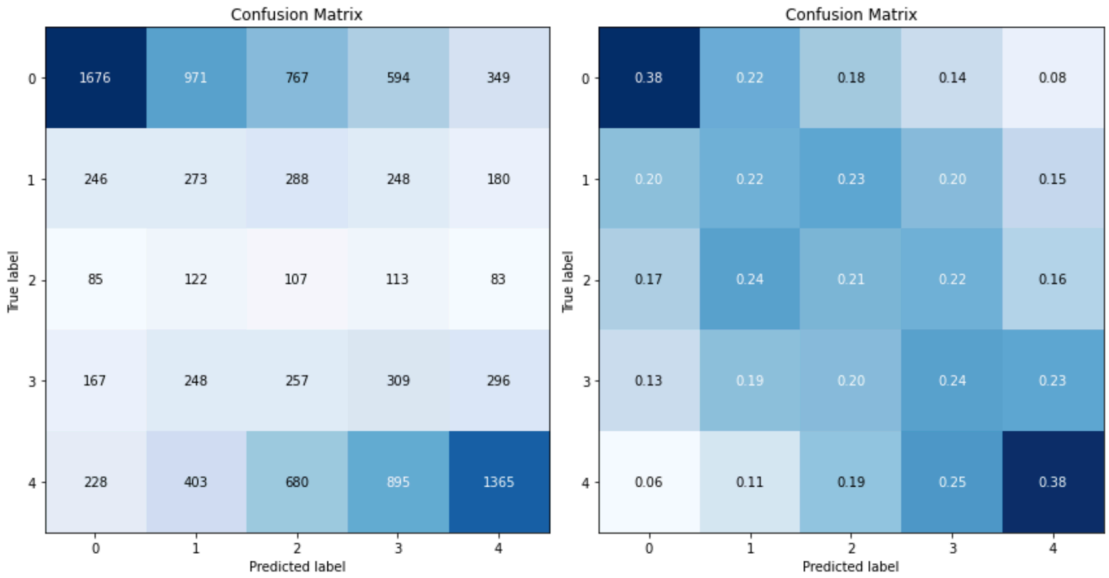


Figure 4.2.9 Confusion Matrices

- Long Short-Term Model:

Through model hyperparameters, the number of hidden nodes has been set to 128, For the loss function, we use binary cross entropy, and set the learning rate equals to 1e-5. Through grid search, the optimal batch size has been set to 64, epochs 50, and lags 5. The LSTM an average accuracy score of 35.19% on the test set. Compared with the Neural Network, the LSTM model has a slight improvement. This may be due to that the memory function of this recurrent neural network enables analysis of time series with order dependence between items in a sequence.

LSTM	Accuracy	Recall	F1
Fold 1	33.91%	33.12%	0.3035
Fold 2	31.97%	31.90%	0.2909
Fold 3	33.78%	32.65%	0.2920
Fold 4	35.21%	33.81%	0.3071
Fold 5	33.76%	32.73%	0.2985
Test	35.19%	30.87%	0.3025

Table 4.2.5 LSTM Prediction metric across all folds

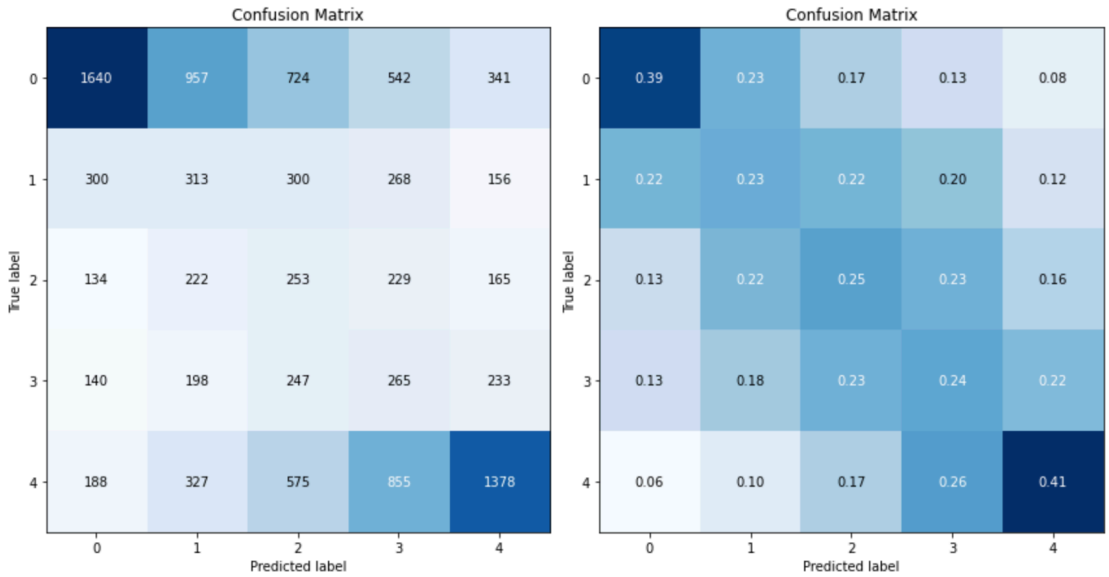


Figure 4.2.10 Confusion Matrices for Long Short-term Model

Combining the accuracy metrics through cross validation and on test data of all models, we create a table to compare them more directly. It turns out that XG Boost Model has the best performance in both cross validation and test data examination, which will be our preferred model for further use.

Models	Items	Accuracy	Recall	F1
Multiclass Logistic	CV	30.91%	29.89%	23.92%
	Test	31.02%	29.83%	23.89%
KNN	CV	32.40%	31.74%	30.07%
	Test	33.73%	31.17%	31.11%
XGBoost	CV	35.21%	34.49%	32.61%
	Test	37.14%	33.74%	33.66%
Neural Network	CV	32.27%	31.31%	27.13%
	Test	33.74%	29.77%	29.12%
LSTM	CV	33.67%	32.77%	27.13%
	Test	34.62%	29.66%	29.41%

Table 4.2.6 Prediction Accuracy Metrics through cross validation and on test data

Conclusion

In this report, we explored the diurnal patterns of the volatility of six stocks based on the 1-year evolution in the limit order book. Besides, we created various summary features based on the limit order book and used them to construct five models to predict the volatility labels using past information. Among all the models, we think XGBoost Model is the most effective one since it has the highest cross-validated accuracy score, recall score and f1 score. At the same time, it also performs well on the test data.

However, there is still improvement space for our model. Hyperparameters optimization using reinforcement learning could be tested over servers with higher computational capacity. Besides, the overall prediction accuracy of our models is still relatively low. Their performances are not discriminant. This could be improved by incorporating more relevant and informative features in our more modelling. We hope to improve on this aspect in future research.

Appendix

Appendix 1 Diurnal Pattern of the range of stock prices