ANALYSIS ON SHORT-TERM VOLATILITY PREDICTIONS

- STA2536 Data Science for Risk Modelling Project 3

Tianyu Ma (1003080337)
Tunan Jia (1002206463)
Qiandan Deng (1002928905)

Nov 28, 2020

Abstract: This project attempts to apply a Markov chain and Neural Network methods to forecast the behavior of the quadradic variation (QV) of the stocks in a day. The prediction of stock market behavior is very important for investors who are seekers for capital appreciation. As one measure of the volatility the Quadradic Variation provides a simple and straightforward way to tell the uncertainty of the asset in a particular time within a day. The application of Markov chain model for forecasting states is based on the strong feature of randomness of the studied objectives, in our project, is the quadratic variation of the price. On the other hand, the Neural Network method assumes for a trackable pattern to predict the objective according to some input features. This project aims to explore the short run behavior of the volatility in a single day, the expected number of appearances to a particular state. For the study, we extracted the data for all trading days in November 2018, which record the time, QV, mid-price, Number of Market Buys, Number Market Sells, Volume Market Buys and the Volume Market sells. We use milliseconds from midnights to track all feature movements starting from the trade start time to close time. For prediction purpose, we apply multilevel classifications based on the observed volatility.

Contents

I. Introduction

The decision on selecting the most beneficial options in the stock market is extremely depends on how well informed you are in the stock analysis. That is way it is most essential to come up with statistical models and their analysis. These models help to predict the share price movement of stocks. The random fluctuation of price of shares causes the uniform distribution of market information. This inherent stochastic behavior of stock market makes the prediction of possible states of the market more complicated.

There are many well-known forecasting methods to solve real-world problems and predict future trends; examples of forecasting methods include the Markov Chain, Artificial Neural Networks (ANN), Hidden Markov Chain, Autoregressive Moving Average Model (ARIMA). Every forecasting model has its own strengths and weaknesses. Selecting the best forecasting model to solve the complex real-world problem with key features is the main objective. In this project, we mainly focus on the Markov Chain process and Artificial Neural Networks to analyze the prediction performance.

For the study, we extracted the data for all trading days in November 2018, which record the time, QV, mid-price, Number of Market Buys, Number Market Sells, Volume Market Buys and the Volume Market sells. We use milliseconds from midnights to track all feature movements starting from the trade start time to close time to predict the short-term volatility in a time, Firstly, we will investigate the univariate distribution of each features. To analyze the features in a more consistent way, we will perform the normal transformation or lognormal transformation, depending on the inherent characteristic of each feature. We use KDE plots to visualize the results. Next, will assign labels to classify the different states of the Quadradic Variable, which will become the main prediction objectives through the whole project. The labels are generated by computing the quantiles of transformed QV, given in a particular time over all trading days. We will also visualize the distribution of those labels through a diurnal pattern.

After generating the labels to each point of time, we are able to move to the learning process. We train two categories of models: the Markov Chain and the Artificial Neural Network. Markov Chain is a special case of the stochastic process, which is one of the prominent tools that has been developed to predict trend for several decades. The method has been used to estimate the transition probability matrix from current state to next state based on certain amount of history. To compare different types of Markov Chain models, we use 1- step and 2- step ahead algorithms. The other main model is the Neural Network, which builds more complicated architectures to learn the pattern from the raw data. In this project, we will use the network to perform a multiple states classification to predict the labels of each QV given the observed features of each time. The label provides a first insight about the volatility level at a particular time. We will test different parameters in the ANN to build an optimized architecture.

Finally, we need to test the accuracy of the model. Since we have only one month of data, we will test the accuracy over the last day by using the left days information as the training source. To compare the prediction ability among different models, we use the Leave-One-Out-Cross-Validation (LOOCV).

II. Data Preparation

In order to provide a fine data source to support the modeling construction, we investigate the univariate distributions of each feature, and explore some simple distributional models to have a better understanding of the raw data set. By summarizing the raw data, we found that more than 300 abnormal zeros are observed on the data from Nov 23rd, then we do some research on what may cause these zeros. We found that Nov.23rd is Black Friday day in 2018. On Nov.23rd, a majority of people also have a day off; thus, the data on this day will be considered as holiday. As a result, we decided to delete the information involved in an outlier day. Next, we plot the kernel densities of different features and fined the estimated parameters to describe the distribution of the features. Meanwhile, we generate the histograms of either the normalized or log normalized features to scale the values, as a result, we will get a general idea of their distributions overall features consistently. It should be noted that neither normalizing nor log-normalizing the features would change the distributions that the original data follows, this data transformation only provides us a better visualization to investigate the univariate distributions of each features. The KDE plots of the original features will be attached in Appendix Part1. For better visualization, the KDE of normalized/ lognormalized data plots will be used to describe the distribution of those features.

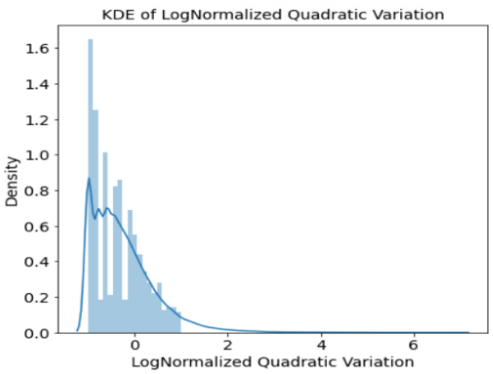

Figure 2.1 KDE of Quadratic Variation

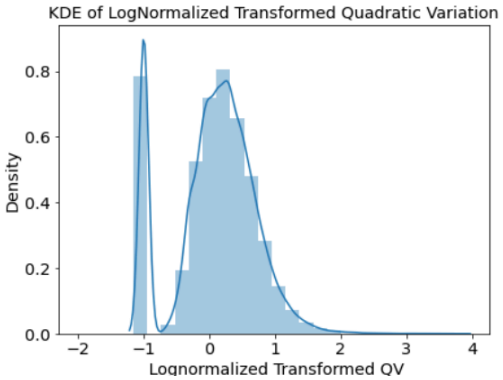

Figure 2.2 KDE of Lognormalized Quadratic Variation

Figure 2.1 shows the KDE plot of Quadratic Variation. Quadratic Variation (QV) measures QV over 10s periods. After deleting data on Nov.23rd, we still observe some points which have relatively higher value of QVs. By plotting QV, we notice that higher QV happens at the beginning of the day or the end of the day, which can be regarded as Outliers from our perspective. Based on the definition of QV, it should be very small for a short period.

Meanwhile, after removing those points, we are able to plot a better univariate distribution and do further investigation. Figure 2.2 shows the KDE plot of lognormalized quadratic variation. Based on the shape of the plot, it seems to follow a mixture model with skewed normal.

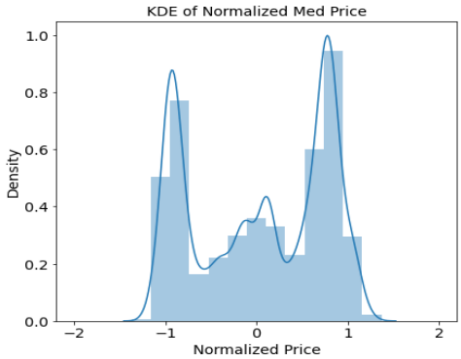

Figure 2.3 KDE of Normalized Med Price

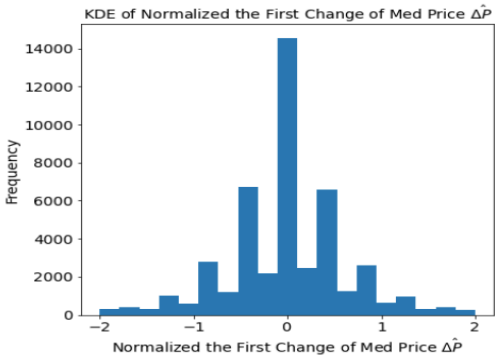

Figure 2.4 KDE of Normalized the First Change of Med Price

Mid-Price is likely to follow a Multinormal Distribution with three normal peaks which are around 27, 30.5 and 33 respectively in orginal data set Mid Price. From Figure 2.3, it gives us consistent results (3 peaks). Figure 2.4 shows the KDE of normalized the first change of med price, which seems to follow a Normal distribution centered at 0 and evenly spread out. This also means in most of time, the med price keeps the same as before.

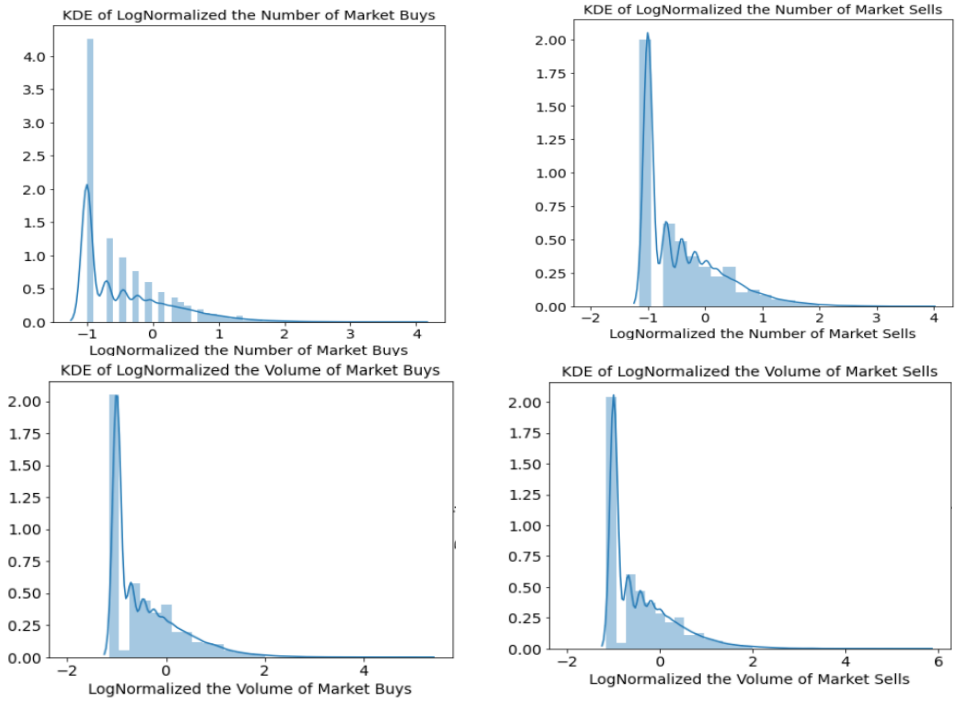

Figure 2.5 KDE of Lognormalized the Number of Market Buys and Sells and the Volume Buys and Sells

Figure 2.5 shows the KDE of lognormalized the number of buys, the number of sells, the volume of buys, and the volume of sells, respectively. The first impression from the above plots is that these features may follow the same distribution with different parameters. Since

four variables from the original dataset are discrete and positive counting variables, they seem to follow a Poisson distribution. However, the range of the original dataset is too wide, which performs bad on visualization. Thus, we decide to take log normalize transformation on the features for a better illustration. For example, the maximum number of market buys is 96, which is quite large. After taking lognormalized transformation, 96 turns to 3.93. In this case, the plot includes some extreme values and would not lose information from the original dataset. Moreover, data transformation would not change the distribution that the original data follows.

The shape of those plots seems to follow Poisson distribution and be shifted to the left by 1 unit. We also observe a large number of data is at -1, which corresponds to the large number of zeros in the raw dataset. The parameter estimates of λ is calculated by averaging the lognormalized data and adding 1. The table 3.6 below summarizes the estimates under Poisson distribution:

	Number of Buys	Number of Sells	Volume of Buys	Volume of Sells
λ Estimates	0.578	0.586	0.563	0.572

Table 2.6 Poisson Distribution Estimates

Properly applying data transformation would be a useful tool to explore data distribution and its features.

III. Diurnal Pattern Computation

In this section, we generate labels based on quadratic variation on the same time for all 20 days. In other words, we first fix a time, then we generate labels based on all quadratic variation observed at this time for all 20 days. We notice there are many zeros at a given time for 20 days. Since the minimum value of quadratic variation is 0, the code given in class cannot consistently assign label 0 to all zero values; thus, we change the method about how to measure the 1/5, 2/5, 3/5, 4/5 quantiles for each 10s bucket of the day and how to label QV at each point in the day by the corresponding quantiles.

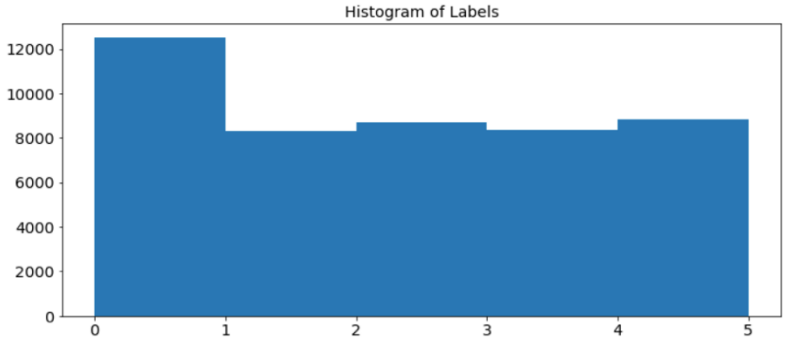

Figure 3.1: Histogram of Labels based on Quadratic Variation

Figure 3.1 shows the histogram of labels based on quadratic variation. From the QV plot which will be given in Appendix Part1, it is obviously most of QV is 0 or close to 0. The plot above gives some conclusion that the maximum number of labels is 0 (over 12000). Next, Diurnal pattern refers to the changes throughout a period, which would visualize the changes of quantiles. Diurnal pattern of quadratic variation provides a better tool to illustrate how the quantiles fluctuate over time. Figure 3.2 below shows the diurnal pattern of quadratic variation at each time during the 20 days' trading hours.

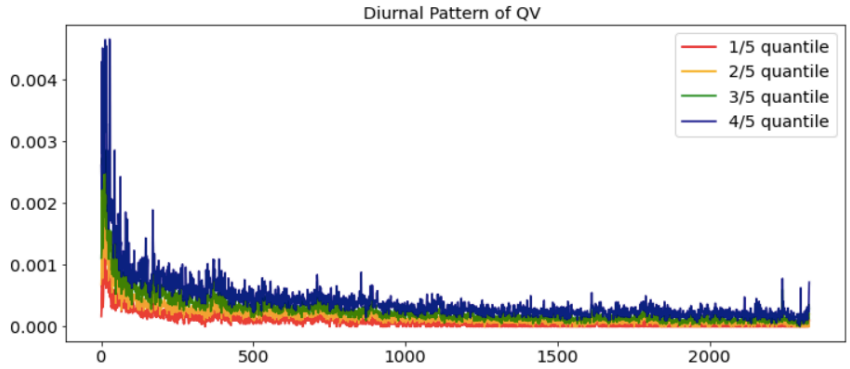

Figure 3.2: Diurnal Pattern of Quadratic Variation Quantiles

From the plot, quantiles have a downward trend with violent fluctuations. At the beginning of the plot, the quadratic variations corresponding to the 1/5 quantile are almost greater than 0. This means the quadratic variations are greater at the first few days, and the diurnal plot gives the same information. From time 1000 onwards, the 1/5 quantiles start to be zero. From time 1500 onwards, both 1/5 quantiles and 2/5 quantiles start to be zero. This tells us the more than half of the quadratic variations from the 20 weekdays are zero.

IV. Markov Chain Prediction

A **Markov chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. The stock market has a significant contribution in the swiftly growing world economy. The fluctuation in stock market can have a profound influence on individuals and the entire economy as well. Stochastic processes can be distinguished in different types depending upon the state space, index parameter and the dependence relations among the random variables through the specification of the joint distribution function. The sequence of $\{X_n, n \geq 0\}$ is said to be a Markov Chain if:

$$Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = Pr(X_{n+1} = x \mid X_n = x_n)$$

This means regardless of the history prior to time n, the probability of moving to the next stage only depends on the present state and not the previous states. In this project, we consider 1-step and 2-step ahead Markov Chain model for the transitions; thus, we will show the transition probability matrix separately:

- 1- Step: $\widehat{P_{ij}} = \Pr(X_t = j | X_{t-1} = i) = \frac{n_{ij}}{\sum_j n_{ij}}$
- 2- Step: $\widehat{P_{ij,z}} = \Pr(X_t = z | X_{t-1} = i, X_{t-2} = j) = \frac{n_{ij,z}}{\sum_z n_{ij,z}}$

where nij denotes the counted number of times label goes from i to label j,
and $n_{ij,z}$ denotes the counted number of times label was in i at t-1, and j at t-2, and then move to label z.
The Markov transition probability model begins with the probability based on our training data, and then we will use the test dataset to check the accuracy.

i. Step Markov Chain Model:

The estimated transition matrix is estimated by using the probability equation above and all dataset. It is simply the counted number of times label from i to j divided by the total number of transitions. Meanwhile, we apply the Leave-one-out Cross Validation (LOOCV) to test the accuracy of the model. LOOCV is an extreme case of K-fold cross-validation. It requires the model to be trained using (n-1) observations and tested using the one that is 'left out'. Since the method needs to loop over the entire dataset, the computational cost is very high. Since we have only 20 samples, we can benefit from LOOCV by a more robust estimate of model performance. The procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data.

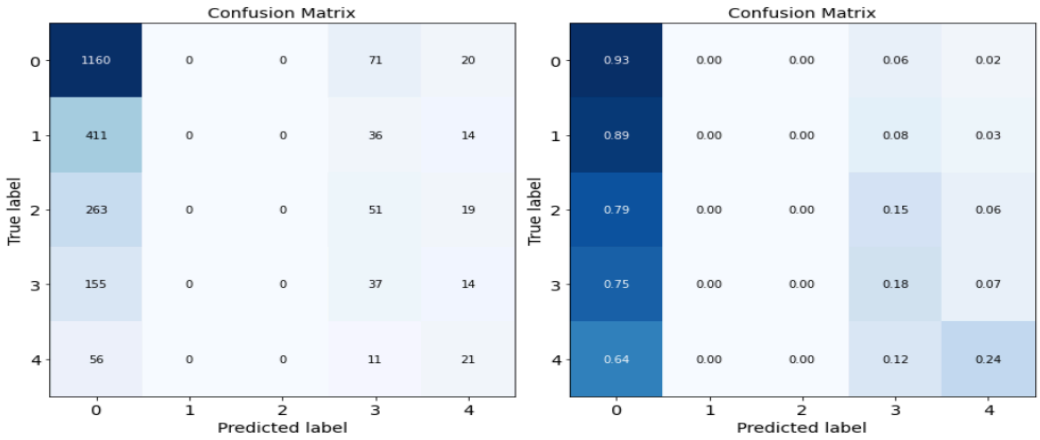

Figure 4.1: Confusion Matrix of Last Day Prediction under 1-Step ahead Markov Chain

Figure 4.1 shows the confusion matrix of last day prediction under 1-step ahead Markov Chain model. We notice most of the predicted labels are 0 since the last day (test) data contains a lot QV= 0. Under our method of generating labels, if more than half of the dataset is zero, there probably will be no predicted label 1 and label 2. This is the trade off by improving the method on how to assign label 0 to QV= 0. The accuracy on the last day is 0.5207. Moreover, the average training accuracy of the model is 0.3339. The one step

Markov Chain model performs better than random guessing (1/5). We are wondering how to increase the accuracy. Thus, we consider doing more investigations by using the 2-step Markov Chain.

ii. 2- Step Markov Chain Model:

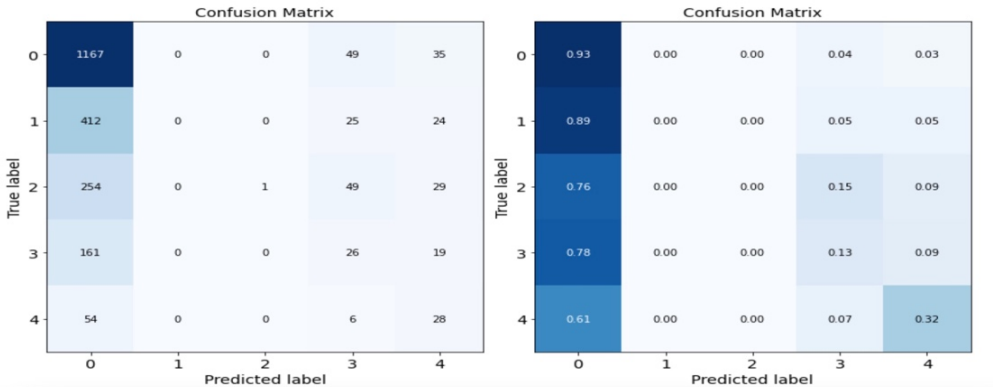

Figure 4.2: Confusion Matrix of Last Day Prediction under 2-Step ahead Markov Chain

Figure 4.2 shows the confusion matrix of last day prediction under 2-step ahead Markov Chain model. The average training accuracy is only 0.3414. The two step Markov Chain model performs better than one step Markov Chain model. Moreover, the accuracy of the model on the last day is 0.5224, which is slightly better than 2-step ahead Markov Chain model. However, the result is still not desirable; thus, we consider doing more investigations by introducing Neural Nets.

V. Artificial Neural Nets Prediction

Neural network forms the base of deep learning, where the structure is inspired by the human brain. It is made up of neurons. These neurons are key processing units of the network. The output of each neuron is defined by a weight and an activation function. Artificial Neural Nets (ANN) are essentially mapping $F: \mathcal{X} \to \mathcal{Y}$ from input space $\mathcal{X}$ into an output space $\mathcal{Y}$ through some intermediate maps, called hidden layers. The architecture of the neural net tells us how the neurons are connected to each other. Figure 5.1 shows a simple example of neural net architecture. The specific architecture, number of hidden layers, number of nodes in each layer, activation function are all hyper-parameters. We tune those parameters to make our model to achieve the best performance.

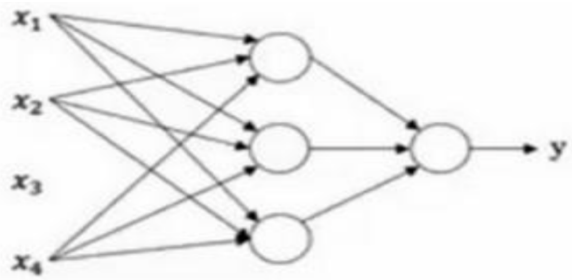

Figure 5.1: Example of Artificial Neural Nets Diagram

In this project, we use ANN to perform classification for QV labels. The basic idea behind this is similar to multi-class logistic regression model. The neural net model is as follows:

$$\mathbb{P}(Y = y \mid X = x) = \frac{e^{F_y(x;\Theta)}}{\sum_{y' \in Y} e^{F_{y'}(x;\Theta)}}$$

in which $F_y(x;\Theta)$ is the output of a neural network consisting of all possible labels. And Θ represents all parameters in the model, such as the weight and biases. We use cross entropy as our loss function. Each predicted class is compared to the actual class, then the loss score is calculated based on how far away it is from the actual value. Cross-Entropy loss is used to adjust the model weight and biases. Cross-Entropy is defined as the following:

$$L = -\sum_{i=1}^{n} y_i \log(p_i)$$

in which n is the number of classes, y_i is the true label and p_i is the probability for the i^{th} class.

i. Optimization

Once a loss function is provided, the goal is to minimize the loss. The gradient descent (GD) method uses the gradient of the loss function to update an estimate of the optimal parameters Θ. The update rule is as follows:

$$\Theta \leftarrow \Theta - \eta \nabla \mathrm{L}(\mathrm{X}, \mathrm{Y}; \Theta)$$

where

$$\nabla L(X, Y; \Theta) \coloneqq \begin{pmatrix} \partial_{\theta_1} L(X, Y; \Theta) \\ \partial_{\theta_2} L(X, Y; \Theta) \\ \vdots \\ \partial_{\theta_M} L(X, Y; \Theta) \end{pmatrix}$$

Denote the vector of derivatives of the loss function L. The parameter η is called the learning rate, which controls the speed of the movement. Traditional gradient descent run through the whole training data to update a parameter in a single iteration. This will take a long time to train a model and has large computational costs.

Therefore, we introduce another optimization method, called Stochastic Gradient Descent (SGD). Instead of using the whole training set in one iteration, SGD uses one or subset of data to update parameters in one iteration. If you use a subset of data, it is called Mini-Batch SGD. SGD will be much faster than GD, but the results may not be as accurate as GD. Since we need to perform LOOCV, using SGD would be a better choice than GD.

Adam is an extension to SGD. It is an effective algorithm, and it achieves good results fast. In this project, we try different optimization methods and choose to use Adam. The algorithm is as follows:

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 5.2: Adam Algorithm for Stochastic Optimization[1]

ii. Back Propagation

In the GD method described in the previous section, as well as other optimization methods, a key objective is to find the gradient $\nabla L(\Theta)$. For simple model, we can derive analytical closed form of formula. However, it is too difficult in ANN. There is an efficient numerical scheme to compute this task, called back-propagation. Suppose $\Theta = (\theta_1, \theta_2)$ and the loss is given by the sequence of maps

$$(\theta_1, \theta_2) \to z := f(\theta_1, \theta_2) \to L(z)$$

We can compute the gradient using simple chain rule as follows:

$$\frac{\partial L(\Theta)}{\partial \theta_i} = \frac{\partial L(z)}{\partial z} \times \frac{\partial f(\Theta)}{\partial \theta_i}$$

The backward pass (Figure 5.3) maps, from right to left, the sensitivity of the loss to changes in z into the sensitivities of the loss to changes in each parameter. The top left link corresponds to the sensitivity of the loss to θ_1. The bottom left link corresponds to the sensitivity of the loss to θ_2.

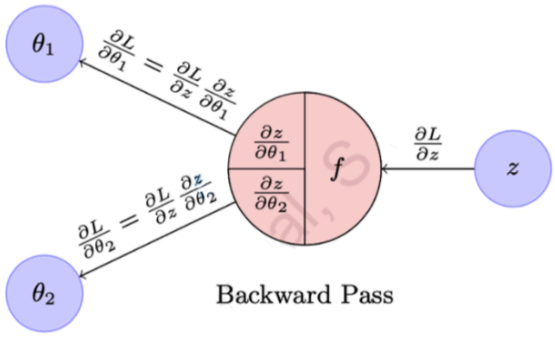

Figure 5.3: A Basic Computation Graph with Two Input Parameters showing Backward Passes

iii. The Choice of the Activated Functions

The activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network and determines whether it should be activated or not based on whether each neuron's input is relevant for the model's prediction. Since our objective in this project is to perform the logistic classification, the activation function at the output layer must help normalize the output of each neuron to a range between 0 and 1.

Another aspect of activation function is that they must be computationally efficient and. Because we use the backpropagation to train the model, the chosen activated function must ensure that the network can perform backpropagation even the inputs approach zero or are negative.

In a neural network, numeric data points are fed into the neurons in the input layer, the right choice of number of layers and the activated function in each layer ensures that learning process can be performed effectively and efficiently. We tested multiple activated functions in this project:

Activated function	Advantage	Disadvantage
Sigmoid	• Output values bound between 0 and 1, normalizing the output of each neuron	• Vanishing gradient-for very high or very low values of X, there is almost no change to the prediction, which is the case in this project, causing the network refuses to learn further • Computationally expensive
ReLu	• Computationally efficient • Allows for backpropagation	• When inputs approach zero or are negative, the gradient vanishes, and the network can't perform backpropagation anymore.
Leaky ReLU	• Prevent the dying ReLU problem	• Results are not consistent

Figure 5.4: Advantages and Disadvantages of Activated Functions

When choosing the number of layers, we started from 2 layers and increased until 5 layers. However, considering we have a small data size, the impact of layer numbers did not show the significance in terms of the prediction performance on the test. So, we stayed at the starting set of 2 layers. With the perspective of activated functions, we implemented sigmoid from the inputs to the first layer, to make sure the output arrived at the layer neurons are normalized and bond between 0 and 1. Then we implement ReLU for all the hidden layers and the final output layer. This architecture is built on multiple testing results.

iv. NN Results

	Features	Lag Number	Nodes Numbers	LOOCV	Test Accuracy on the last day
Model 1	Normalized: Price change; Lognormalized: NB, NS, VB, VS, Transformed QV	1	20	0.3657	0.4063
Model 2	Normalized: Price change; Lognormalized: NB, NS, VB, VS, Transformed QV	2	20	0.3667	0.4170
Model 3	Normalized: Price change; Lognormalized: NB, NS, VB, VS, Transformed QV	2	50	0.3637	0.4045
Model 4	Normalized: Price change; Lognormalized: NB, NS, VB, VS, QV	1	20	0.3728	0.4359
Model 5	Normalized: Price change; Lognormalized: NB, NS, VB, VS, QV	2	20	0.3680	0.4465
Model 6	Normalized: Price change; Lognormalized: NB, NS, V B, VS, QV	1	50	0.3706	0.4628

Figure 5.5: Accuracy Tables under Different Models

Figure 5.5 shows the LOOCV accuracy and test accuracy under different models. We observe that our best fitted model with the highest LOOCV accuracy, which is suggested by model 4. Now, we fit the best model on our test dataset, the accuracy on the last day prediction is 0.4359. The best fitted model contains feature normalized price change, log-normalized number of buys, number of sells, volumes of buys, volumes of sells, and quadratic variations of the price in each 10 seconds. The confusion matrix of our best model is given below, which shows the total number of predicted labels. The prediction performance under ANNs overall beats the Markov Chains. To furtherly improve the model performance, there are many other models to consider such as Hidden Markov Model, and more data sources are required.

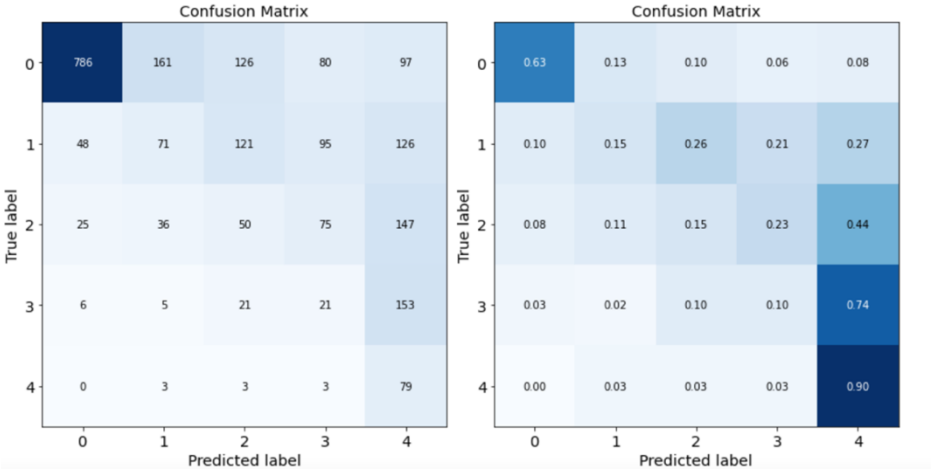

Figure 5.6: Confusion Matrix of the Best Model

VI. Conclusion

In the first part of this project, we perform exploratory data analysis on the given datasets. We investigate the univariate distribution of each of the features. The summary of the distribution is as the following:

- QV is likely to follow a Mixture Model with skewed Normal
- Mid-Price is likely to follow a Multinomial Distribution with three normal peaks
- Number of buys, number of sells, volume of buys, volume of sells is likely to follow Poisson Distribution with different parameter, with λ= 0.578, 0.586, 0.563, 0.572 respectively

In the second part, we compute the diurnal pattern of QV during the day by measuring the 1/5, 2/5...,4/5 quantiles for each 10s bucket of the day, and label QV at each point in the day by the quantile which it falls within.

After data preparation, we test several modeling approaches for predicting the QV labels. Including one-step and two-step Markov Chain Model and several Neural Networks with different hyperparameters. In all cases, we test the accuracy of the model on the last day and perform Leave-One-Out Cross Validation to see the performance of these models. Since we include more features in Neural Networks than in a Markov Chain Model, the accuracy in predicting labels is higher in the Neural Networks.

VII. Reference

1. *Kingma, D and Ba, J . (2015) Adam: A method for Stochastic Optimization. Available at: https://arxiv.org/pdf/1412.6980.pdf*

VIII. Appendix

i. Part1

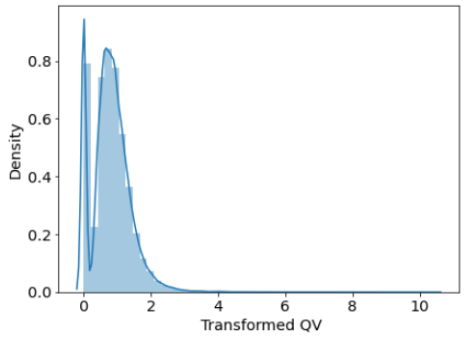

Figure 8.1 KDE of Transformed Quadratic Variation

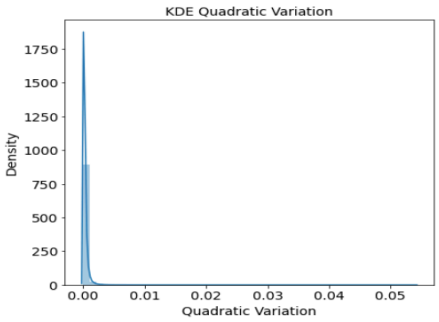

Figure 8.1 KDE of Transformed Quadratic Variation

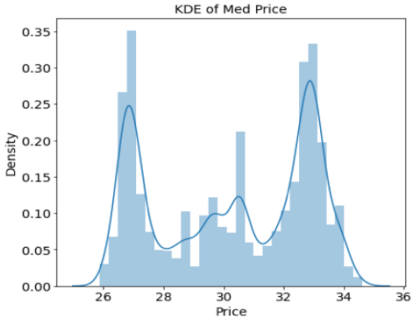

Figure 8.3 KDE of Med Price

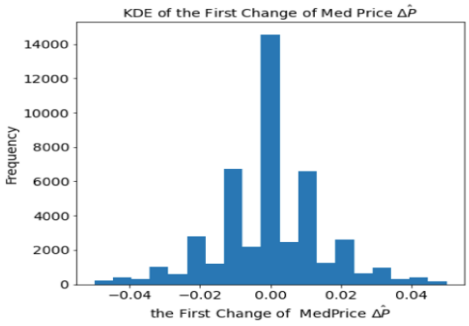

Figure 8.4 KDE of the First Change of Med Price

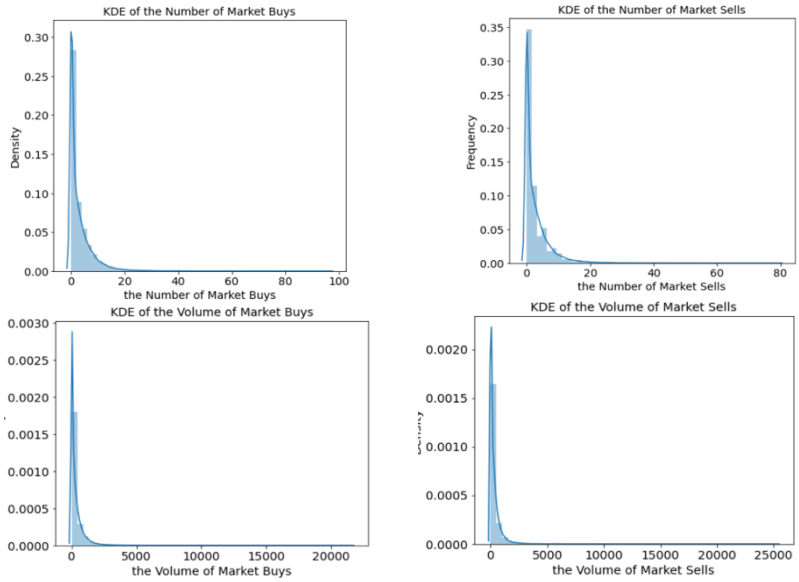

Figure 8.5 KDE of the Number of Market Buys and Sells and the Volume Buys and Sells

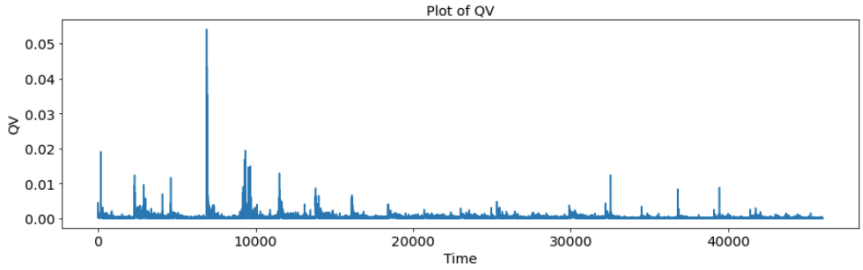

Figure 8.6: Plot of Quadratic Variation