Cryptocurrency Price Forecasting with Tick-Level Order Book Data

Thesis report submitted in fulfillment of the requirements

for the degree of

Bachelor

by

LI Jialu, WU Xiang

(SID: 1155107895, 1155124573)

Under the supervision of

TAO Yufei

Computer Science
The Chinese University of Hong Kong
August 13, 2022

Contents

Abstract

This paper focuses on cryptocurrency price forecasting. Considering that the data deficiency and market sentiment are two significant challenges for machine learning models to predict cryptocurrency price, we extract features from the tick-level order book data of crypto market, which has a large scale and is supposed to depict traders' behaviors in the market to some extent. We construct the recurrent neural network with attention layer to improve the forecasting performance on the time-series mid-price data. Additionally, to provide our prediction model with more information of market sentiment, we proposed a "conditional training/prediction" method with the "Fear and Greed Index" in this paper. Finally, experiments and backtesting are designed and conducted to examine the performance of our models.

Keywords: machine learning, Recurrent Neural Network, time-series forecasting, tick-level order book data, cryptocurrency price, market sentiment, backtesting.

1 Introduction

The financial market has been attractive to people from all walks of life for a long time. Re-searchers, experts, entrepreneurs, and amateurs, have been endeavoring to explore methods of predicting the price trend of various assets in the financial market, which means "making money from money". Among these assets, the cryptocurrency has the attraction with its characteristics like security, convenience, and novelty. However, the volatility of cryptocurrency price can be attributed to a large number of factors, such as individual sentiment, traders' behaviors, political events, natural hazards, and so on. Under the circumstances, an amateur without sufficient professional knowledge, can hardly benefit from the complicated and changeable crypto market. Fortunately, the artificial intelligence can serve as a powerful assistant for participants in the financial market, regardless of professional knowledge level.

Nowadays, machine learning is one of the popular methodologies in financial market prediction, but there are many challenges in this field. First of all, in order to assess the performance of a trained machine learning model, the sample set should have as similar data distribution to that in real world as possible. This fundamental requirement brings two essential challenges for crypto market prediction via machine learning: one is the deficiency of data, the other is that we do not even know whether there exists such a regular distribution for the crypto market data. Additionally, there are also many technical challenges, such as metrics selection, data quantification and feature extraction. Therefore, it is hardly expected that machine learning can have better performance in crypto market prediction than financial experts.

On the other hand, the machine learning also has its undoubted advantages. In previous researches on the stock market prediction, the application of machine learning models in modeling high-frequency limit order book dynamics to forecast the stock prices has been proven to be feasible [1][2]. An order book records the high-frequency trading behaviors in the financial market, and some metrics extracted from the tick-level order book data can significantly reflect the price fluctuation of a security, e.g., mid-price movement and bid-ask spread. Compared with the daily data, the tick-level data has much larger scale, which alleviated the problem of data deficiency mentioned before. In addition, the intrinsic information of high-frequency data is hard to be captured by human, so we can expect machine learning to exploit its advantages in this field.

It is notable that the tick-level order book data is a kind of time-series data. Time-series data, just as its name implies, is a series of data points indexed in time order. Many previous researches shows that, the Recurrent Neural Network (RNN) models, especially Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) have a great performance on Time-series data forecasting. For example, [3] used LSTM model to conduct weather forecasting, [4] employed LSTM and GRU to forecast the new cases and new deaths rate for COVID-19 in a time-series manner.

In our previous project, we tried to reproduce and modify the method in [1], which concentrates on the stock market, to predict the crypto price trend, using the tick-level order book data. Specifically, given the tick-level order book data of some crypto currency on some exchange in a particular time period, we employed multi-class SVM classifier to generate a label indicating the direction of mid-price movement in some

time interval after the specific time period. Additionally, we proposed a "conditional training/prediction" method to improve the performance of the model.

In this paper, instead of keeping using the SVM classifier, we employed LSTM and GRU to build a neural network prediction model, and we continued to use the "conditional training/prediction" method in order to improve the prediction performance. Since now we are predicting the mid-price in a regression manner, we also need to modified our trading strategy to apply the model to real transaction.

The remainder of this report will be organized as follows: In section 2, we will introduce some preliminaries which helps you to better understand our project. In section 3, we will go through all the details of our system, including the GUI design, dataset description, model building, experiments analysis and trading method. And in section 4, we will have a brief summary for our project in this semester.

2 Preliminaries

2.1 SVM

A support vector machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis.
In SVM:
Input:

$$\{(x^1, y^1), (x^2, y^2), ..., (x^N, y^N)\}$$

$$\text{x} = [1, x_1, x_2, ..., x_d]^T, \text{ y} \in \{0,1\}$$

ω: parameters of the decision hyper plane vector

Support Vector Machine (SVM) is to minimize:

$$\min_{\omega} \sum_{t=1}^{N} y^t cost_1(\omega^T x^t) + (1 - y^t) cost_0(\omega^T x^t) + \frac{1}{2} \sum_{j=1}^{d} (\omega_j)^2$$

$$cost_1(\omega^T x) = log \frac{1}{1 + e^{\omega^T x}}$$

$$cost_0(\omega^T x) = log(1 - \frac{1}{1 + e^{\omega^T x}})$$

- Whenever $y^t = 1$, we can only consider $cost_1(\omega^T x^t)$, and we need to try to satisfy $\omega^T x^t \geq 1$
- Whenever $y^t = 0$, we can only consider $cost_0(\omega^T x^t)$, and we need to try to satisfy $\omega^T x^t \leq -1$

2.1.1 Multi-class SVM

SVMs are inherently two-class classifiers.

In particular, the most common technique in practice has been to build a one-versus-rest classifiers, and to choose the class which classifies the test datum with greatest margin. Another strategy is to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. A better implementation is provided by the construction of multi-class SVMs, where we build a two-class classifier over a feature vector (x, y) derived from the pair consisting of the input features and the class of the datum.

The Scikit-learn package has the built in multi-class method, which makes it easy for us to implement. For **sklearn.svm.SVC**, the multi-class support is handled according to a one-vs-one sheme, for **sklearn.svm.LinearSVC**, the multi-class support is handled according to one-vs-the-rest shee, whcih makes it faster to be trained.

2.1.2 Kernel

In SVM algorithm, we mainly care about the inner product of vector pairs. A kernel function is used to implicitly map the data from a low-dimensional feature space to a high-dimensional space and quickly compute the inner product, so that the SVM model can deal with the linearly inseparable data. Some commonly used kernel functions are as follows[5]:

- Linear Kernel

$$K(x, x_i) = x\dot{x}^T$$

- Polynomial Kernel

$$K(x, x_i) = (1 + x\dot{x}_i^T)^d$$

- Sigmoid Kernel

$$K(x, x_i) = tanh(\iota x_i^T x_j + r)$$

- Radial Basis Function

$$K(x, x_i) = e^{-\iota||x-x_i||^2} \text{ , kernel function parameter } \iota > 0$$

2.2 Recurrent Neural Network (RNN)

RNN is a type of artificial neural network that connections between nodes form a sequence. It allows temporal dynamic behavior for time sequence. In this project we choose to use these two types of RNN: long short term memory unit (LSTM) and gated recurrent unit (GRU), instead of the simple RNN model. And we carefully investigated in the attetnion layer, and cooperate this mechanism in our model.

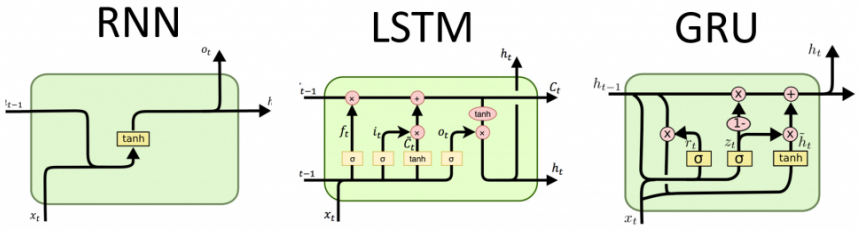


Figure 1: The comparison between core structures of RNN, LSTM and GRU

2.2.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM), it is also proposed to solve problems such as long-term memory and gradients in back propagation. It has the same input and output structure with other ordinary RNNs. But

LSTM has better performance than RNN, especially vanishing gradient problem.
In LSTM:

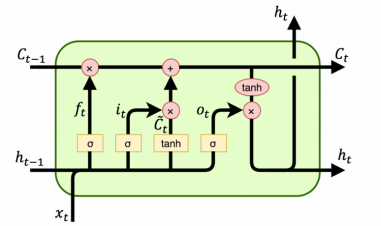


Figure 2: The core structure of LSTM

h_t, C_t : hidden layer vectors. x_t : input vector. b_f, b_i, b_c, b_o: bias vector. W_f, W_i, W_c, W_o: parameter matrices. $\sigma, \tanh$: activation functions. The feed forward process is below:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

And the back propagation is:

$$\frac{\partial C_{t+1}}{\partial h_t} = \frac{\partial C_{t+1}}{\partial \tilde{C}_{t+1}} \frac{\partial \tilde{C}_{t+1}}{\partial h_t} + \frac{\partial C_{t+1}}{\partial f_{t+1}} \frac{\partial f_{t+1}}{\partial h_t} + \frac{\partial C_{t+1}}{\partial t_{t+1}} \frac{\partial i_{t+1}}{\partial h_t}$$

$$\frac{\partial C_{t+1}}{\partial C_t}$$

$$\frac{\partial h_{t+1}}{\partial C_t} = \frac{\partial h_{t+1}}{\partial C_{t+1}} \frac{\partial C_{t+1}}{\partial C_t}$$

$$\frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial h_{t+1}}{\partial C_{t+1}} \frac{\partial C_{t+1}}{\partial h_t} + \frac{\partial h_{t+1}}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial h_t}$$

$$\Pi_t = \frac{\partial E_t}{\partial h_t} + \frac{\partial h_{t+1}}{\partial h_t} \Pi_{t+1} + \frac{\partial C_{t+1}}{\partial h_t} \mathcal{T}_{t+1}$$

$$\mathcal{T}_t = \frac{\partial E_t}{\partial h_t} \frac{\partial E_t}{\partial C_t} + \frac{\partial h_{t+1}}{\partial C_t} \Pi_{t+1} + \frac{\partial C_{t+1}}{\partial C_t} \mathcal{T}_{t+1}$$

$$\beta_t^f = \beta_{t+1}^f + \frac{\partial C_t}{\partial f_t} \frac{\partial f_t}{\partial W_t^f} (\frac{\partial h_t}{\partial C_t} \Pi_t + \mathcal{T}_t)$$

$$\beta_t^i = \beta_{t+1}^i + \frac{\partial C_t}{\partial i_t} \frac{\partial i_t}{\partial W_t^i} (\frac{\partial h_t}{\partial C_t} \Pi_t + \mathcal{T}_t)$$

$$\beta_t^c = \beta_{t+1}^c + \frac{\partial C_t}{\partial \tilde{C}_t} \frac{\partial \tilde{C}_t}{\partial W_t^c} (\frac{\partial h_t}{\partial C_t} \Pi_t + \mathcal{T}_t)$$

$$\beta_t^o = \beta_{t+1}^o + \frac{\partial h_t}{\partial o_t} \frac{\partial o_t}{\partial W_t^o} (\Pi_t)$$

2.2.2 Gate Recurrent Unit (GRU)

GRU is a type of RNN (Recurrent Neural Network) as well. GRU and LSTM have similar inner structure, but GRU does not have the output gate, so it requires simpler frame work, less parameters, less computing power and also easier for us to implement. And GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. In GRU:
h_t : hidden layer vectors. x_t : input vector. b_z, b_r, b_h : bias vector. W_z, W_r, W_h : parameter matrices. $\sigma, \tanh$: activation functions The feed forward process is below:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

And the back propagation is:

$$\Pi_t = \frac{\partial E_t}{\partial h_t} + \frac{\partial h_{t+1}}{\partial h_t}\Pi_{t+1}$$

$$\beta_t^{\tilde{z}} = \beta_{t+1}^{\tilde{z}} + \frac{\partial h_t}{\partial z_t}\frac{\partial z_t}{\partial W_t^{\tilde{z}}}\Pi_t$$

$$\beta_t^{r} = \beta_{t+1}^{r} + \frac{\partial h_t}{\partial \tilde{h}_t}\frac{\partial \tilde{h}_t}{\partial r_t}\frac{\partial r_t}{\partial W_t^{r}}\Pi_t$$

$$\beta_t^{h} = \beta_{t+1}^{h} + \frac{\partial h_t}{\partial \tilde{h}_t}\frac{\partial \tilde{h}_t}{\partial W_t^{h}}\Pi_t$$

2.2.3 RNN with Attention Layers(AT-LSTM)

One of the major drawbacks of a basic version of LSTM is that, current node has access to information for all the input so far, e.g. the final node of LSTM will hold all information of the entire input sequence. In the learning process of LSTM, it has to encode and "compress" the sequence into a single vector, which lead to partial information lost in this process. This problem can't be ignored especially in our project, where we cooperate with data sets at the level of million. So we add one more attention layer which is first mentioned in this paper [6], and the idea of "attention" serve as a breakthrough in the field of finance.
Attention mechanism helps a neural network in memorizing the large sequence of data.
The first paper of the Attention model is Bahdanau's paper [6]. Here a bidirectional LSTM is used for natural language processing. However, we can't directly use this bidirectional model in this finance field, we can only use one direction to predict.
But we still adopt and follow the structure discussed in Bahdanau's paper. And the core structure is as below, the context vector c_i for the output y_i is generated using the weighted sum:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

The weights α are computed by a softmax function given by the following equation:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

In this way, α is a vector that stores the weights corresponding to each element in the long input. Therefore, we cooperate this mechanism into our LSTM and GRU model to do experiments.

3 System design

3.1 GUI Description

Compared with other existing trading systems supported by machine learning techniques, we developed a graphical user interface to allows users to use our system through a visual interface, instead of text-based command prompt interface. This could largely decrease the difficulty of using our system.
The first page that our system will prompt out to the user is below:

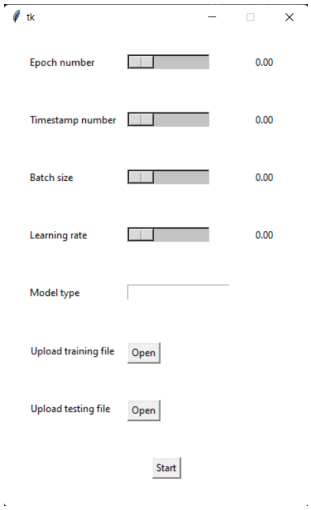


Figure 3: The starting page of our system

Users can input their specific requirements of the number of timestamps that they want to predict, the number of epochs that the model will be trained, the number of batch size and model type. Here the model type is required to be "LSTM", "GRU" or "SVM". And they can also upload their data from their local computers, two datasets for training and testing respectively. After all parameters are decided and files uploaded, users can click the "Start" button to start training and testing on our system.

There will be two prompt boxes to ask if the user hope to continue the validation and back testing process. If the click "no", then our system will stop at this stage.

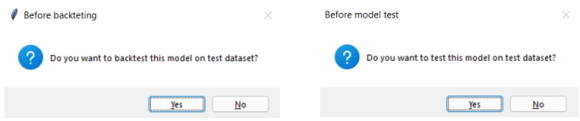


Figure 4: The prompt boxes from our system

If the user choose to continue, then the training process of each epoch and the transactions submitted during backtesting will be print out in the command prompt, as shown below:

After the validation and backtesting process finished, the results including figures and metrics will be shown to the user.

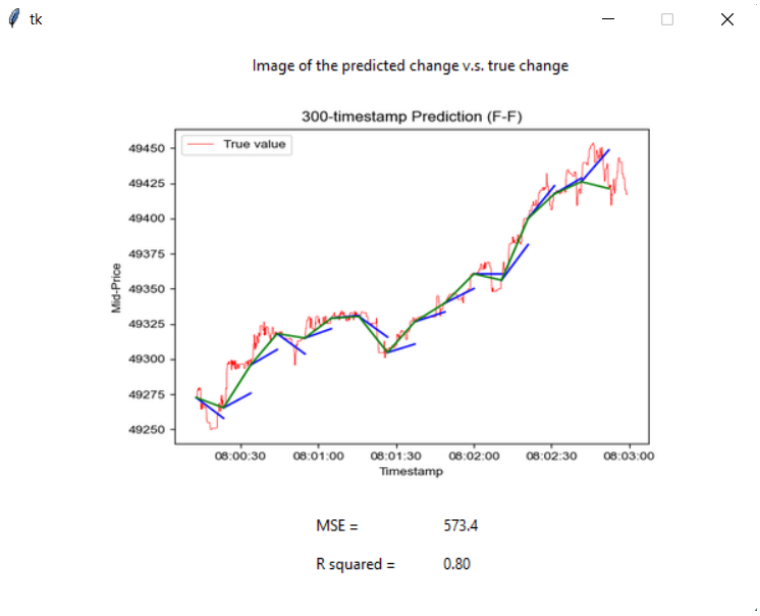


Figure 5: The testing result page of our system

Then the backtesting result will be prompted to users. This page will not only show the trading actions and the return graph, but also show statistics like: the maximum drawdown this whole period, the accuracy of trading activities, and the trading accuracy for buying and selling respectively.

Figure 6: The backtesting result page of our system

3.2 Dataset Description

3.2.1 Tick-Level Order Book of BTCUSDT

We downloaded the tick-level order book snapshots of **BTCUSDT** in *.csv.gz* files from the cryptocurrency exchange Binance from July 30th, 2021 to November 29th, 2021, via the API provided by *Tardis.dev*. Each row of the order book data (i.e., each snapshot) contains:

- t: a message arrival timestamp in microseconds since epoch.
- $P_a[0\ldots4]$: top 5 asks prices in ascending order.

- $V_a[0...4]$: top 5 asks amount in ascending order.
- $P_b[0...4]$: top 5 bids prices in ascending order.
- $V_b[0...4]$: top 5 bids amount in ascending order.

Roughly speaking, the terms "bid" and "ask" represent the two sides in security trading at a timepoint. A bid price is claimed by a buyer, indicating that he or she is willing to pay for the security at this price; while an ask price is claimed by a seller, indicating that he or she is willing to sell the security at this price. On the order book, the bid prices are sorted in descending order and the ask prices are sorted in ascending order, so that the best bid price refers to the highest bid price, and the best ask price refers to the lowest ask prices at this moment. Once a trading event is cancelled or executed, the corresponding bid/ask price and amount will be removed from the order book snapshots.

A mid-price is the average between the best bid price and the best ask price at the moment. Although the mid-price is not the actual transaction price of the security, it can reflect the price trend to some extent.

Notice that, the order book of one trading day contains about 2.5 million snapshots. This indicates that the interval of thirty continuous timestamps in the order book is about one second.

3.2.2 Crypto Fear and Greed Index

As mentioned before, the market sentiment is one of the significant factors influencing the price fluctuation of cryptocurrency. The Fear Greed Index is a daily indicator used to measure the investor sentiments in the crypto market. The index is scored from 0 to 100, where the market sentiment is considered as fear if the index is less than 50, greed if the index is larger than 50. Since we derived the order book data from July 30th, 2021 to November 29th, 2021, we just query the Fear Greed Indexes of Bitcoins for these 4 months via the API provided by *Alternative.me*, as shown in the Table 1:

Date	Fear and Greed Index	Fear/Greed Classification
2021/07/30	53	Neutral
2021/07/31	60	Greed
2021/08/01	60	Greed
	...	
2021/11/27	21	Extreme fear
2021/11/28	27	Fear
2021/11/29	33	Fear

Table 1: Fear and Greed Index

3.3 Model Build

Figure 7 shows the general flow of our conditional model. Firstly, the tick-level order book data need to be preprocessed. Specifically, we need to complete feature extraction, normalization and labeling in this step. Then the preprocessed data is input into either the fear model or greed model according to the Fear Greed Index of the crypto market on the day of the data source. In the following section, more details of each part will be illustrated.

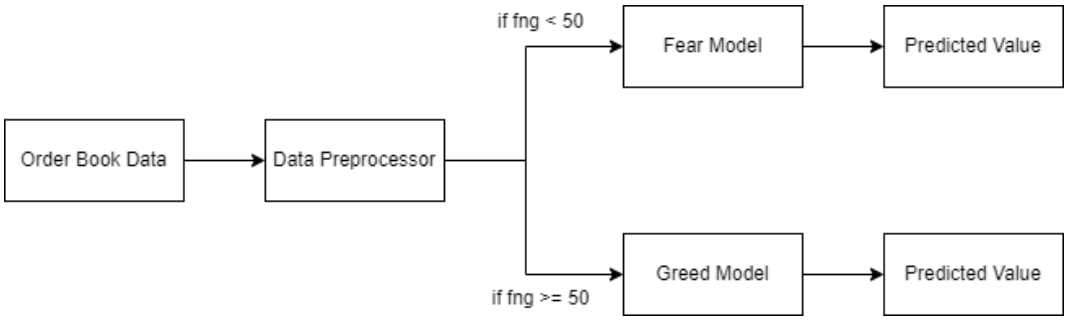


Figure 7: Overview of the Conditional Prediction Model

3.3.1 Data Pre-Processing

The description of the raw tick-level order book data we used is mentioned in section 3.2.1, and the raw data cannot be directly used as the input of the model. The target of the preprocessing is to convert the raw order book data into some proper format, so that it can be fed into the model.

One of the challenge of preprocessing is feature extraction. Since the tick-level order book of the cryptocurrency is similar to the limit order book of the stock price, we referred to article [1], which has already provided us with some reasonable attribute sets extracted from the high-frequency limit order book as shown in figure 8.

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)*

Figure 8: Feature Sets (source: [1])

Unfortunately, we could not extract all of these 9 feature vector sets, because the feature vector sets v_7,v_8,v_9 are dependent on the message book, but we only have the order book data. Therefore, we firstly extracted the feature vector sets 1 to 6 in the above table. Since we were using the sliding window for our multi-SVM classification model, each feature vector set is produced by the first snapshot and the last snapshot of each snapshots window.

We also designed 2 additional feature vector sets for the sliding window classification:

- v_7: mean and standard deviation of ask prices, ask amounts, bid prices, bid amounts in a snapshot

window.

- v_8: total time duration of the snapshot window, which is the difference between the timestamps of last snapshot and the first snapshot of a window. Since the number of snapshots of each window is fixed, and the insertion of a new snapshot in the order book means the cancellation or execution of some trading events, this value can indicate the trading intensity: the shorter the duration is, the more frequently the trading executes.

While for the LSTM/GRU model, we only used the first 5 feature sets because the input of the LSTM/GRU model is an $\frac{N}{S} \times S \times M$ matrix, where N is the number of samples, S is the time steps, and M is the number of sample attributes. Here, each sample is extracted from a single snapshot but not a snapshot window, thus we cannot extract the time-sensitive feature sets.

After the feature extraction, we also need to carry out the normalization, data labeling and train-test split during the preprocessing phase. The whole procedures of the preprocessing for the LSTM/GRU prediction model are as follows:

- Step 1: Extract the feature sets v_1 to v_5 as described above, and simply concatenate all the features, which results in an $N \times 46$ matrix (20 columns for v_1, 10 columns for v_2, 10 columns for v_3, 4 columns for v_4, and 2 columns for v_5), where N is the number of order book snapshots fed into the preprocessor.

- Step 2: Normalize the matrix obtained from Step 1 with the standardization approach, i.e. $z = \frac{x_i - \mu}{\delta}$, where μ and δ is the mean and standard deviation of the corresponding attributes $(i = 1, ..., 46)$ respectively. The purpose of the normalization is mainly to avoid the consequence that the attributes of different magnitude order may cause the unbalanced effect on the model training.

- Step 3: For the multi-class SVM classifier, we label each sample acording to the movement direction of the mid-price (e.g., $Y = \{downward, stationary, downward\}$), while for the LSTM/GRU prediction model, which is essentially a regression model, we label each sample just with the mid-price value after n timestamps, where n is the number of timestamps after which we want to predict the mid-price. Finally we can get a labeled sample set of size $N - n$, because the label of each datapoint depends on the mid-price after n datapoints.
- Step 4: Normalize Y with the min-max normalization approach to scale the output value into the range [-1, 1].
- Step 5: Reshape the 2-Dimensional X matrix into a 3-Dimensional matrix of size $\frac{N}{S} \times S \times M$, as mentioned before.

3.3.2 LSTM/GRU Models Build

In our previous work, we tried to use multi-class SVM as the classifier, but the prediction result is much lower than our expectation. Therefore, we wonder whether the SVM model is not suitable for the high-frequency time-series data classification. For comparison, we employed LSTM and GRU, which is known for its great performance on the serialized data regression. There is another advantage of regression compared with using a multi-class SVM classifier: unlike classification, which can only get a label indicating the movement direction of the mid-price, regression can directly predict the exact value of the mid-price, which may be more helpful and visualized in the realistic trading.
We firstly built a pure LSTM model and a pure GRU model, which contain only one LSTM/GRU layer and one Dense layer, as shown in figure 9 and 10.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 30, 46)]	0
lstm (LSTM)	(None, 30, 50)	19400
dense (Dense)	(None, 30, 1)	51

Total params: 19,451
Trainable params: 19,451
Non-trainable params: 0

Figure 9: Pure LSTM model summary

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 30, 46)]	0
gru (GRU)	(None, 30, 50)	14700
dense (Dense)	(None, 30, 1)	51

Total params: 14,751
Trainable params: 14,751
Non-trainable params: 0

Figure 10: Pure GRU model summary

As can be seen, the parameters number of GRU model is about three quarters of the LSTM model's parameters number. In order to improve the performance of the naive model, we constructed another LSTM model with one more attention layer, as mentioned in section 2.2.1, the summary of the AT-LSTM model is shown in the figure 11.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 30, 46)]	0
lstm (LSTM)	(None, 30, 50)	19400
attention (attention)	(None, 50)	80
dense (Dense)	(None, 1)	51

Total params: 19,531
Trainable params: 19,531
Non-trainable params: 0

Figure 11: AT-LSTM model summary

3.3.3 Conditional Training and Prediction

In our project, we proposed the method called "conditional training/prediction". Roughly speaking, we use different data to train different models, and employ different trained models to carry out prediction based on different dataset, according to some extra conditions. Actually, we cannot expect our models to have a very accurate prediction performance with merely historical order book data, because the fluctuation of the financial market is affected by too many other factors. Obviously, we do not want our model to keep seeking some "illusory rules" blindly from the vast amounts of rambling order book data, and we need to consider these factors. One of the most significant factors is market sentiment. As mentioned in section

3.2.2, the Fear Greed Index indicates the market sentiment for each day. It is reasonable to believe that the trading activity under a “fear” market will be essentially different from the trading activity under a “greed” market. Based on this index, we divided our datasets into two classes, and fed them into two prediction models to train. As a result, we will get two trained models (called as "Fear Model" and "Greed Model") to forecast the cryptocurrency price trend in the fear market and the greed market respectively.

When we get the preprocessed sample sets, there is one more step before we feed them into the prediction model, that is, determine which prediction model should be employed to carry out the prediction task. In this step, we firstly query the Fear Greed Index of the day from which the input data is got. If the FNG index is less than 50, we employ the "Fear Model", otherwise we employ the "Greed Model".

3.4 Experiments and Result Analysis

In our previous work [5], we carried out many experiments on our multi-class SVM classifier, and got some unsatisfactory results. In this section, we only concentrate on the experiment results on the LSTM/GRU prediction models. Generally, we trained 5 models for experiments:

- Model 1: Pure GRU model trained with order book data sourcing from 2021-08-21 (The Fear Greed Index is 78, which is extreme greed), setting $time\_steps = 30$, $n\_timestamp = 300$
- Model 2: Pure GRU model trained with order book data sourcing from 2021-09-28 (The Fear Greed Index is 25, which is extreme fear), setting $time\_steps = 30$, $n\_timestamp = 300$
- Model 3: Pure LSTM model trained with order book data sourcing from 2021-08-21, setting $time\_steps = 30$, $n\_timestamp = 300$
- Model 4: LSTM model with an Attention Layer trained with order book data sourcing from 2021-08-21, setting $time\_steps = 30$, $n\_timestamp = 300$

In the first experiment, we used model 1 to predict the mid-price on 2021-08-23 (greed-greed test). In the second experiment, we used model 2 to predict the mid-price on 2021-08-23 (fear-greed test). In the Third experiment, we used model 3 to predict the mid-price on 2021-08-23 (LSTM test). In the fourth experiment, we used model 4 to pridict the mid-price on 2021-08-23 (AT-LSTM test). Figure 12 shows the training curve of the first experiment, and all other experiments showed the similar training curves. The results of the above 4 experiments are shown in figure 13 and table 2:

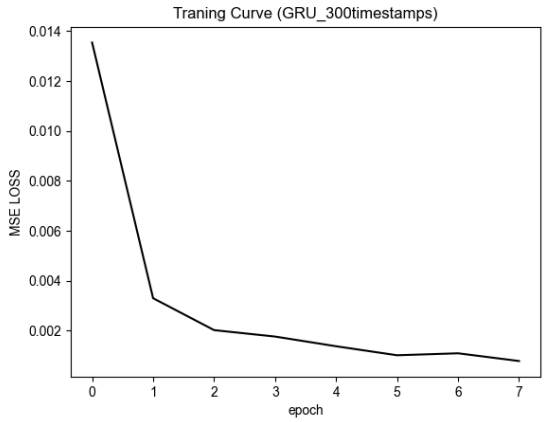
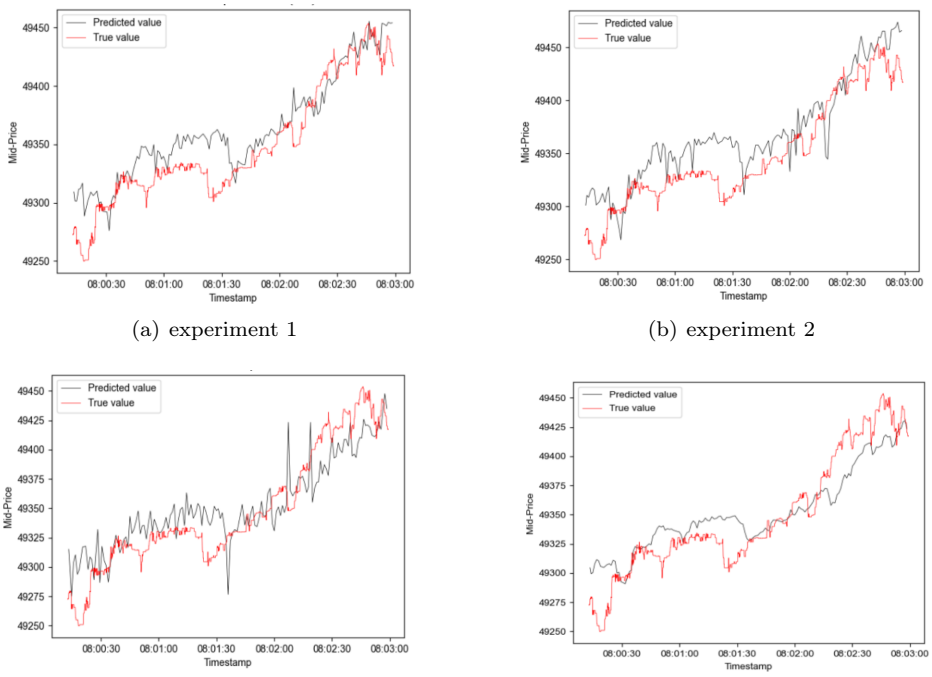


Figure 12: Training curve



(a) experiment 1

(b) experiment 2

(c) experiment 3

(d) experiment 4

Figure 13: Results of experiments 1-4

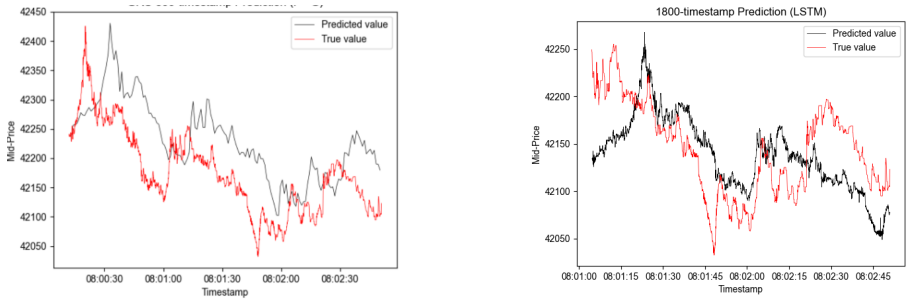
experiment no.	Model Description	Predict on Date	R2_score
1	GRU greed model	2021-08-23	0.79
2	GRU fear model	2021-08-23	0.68
3	LSTM model	2021-08-23	0.77
4	AT-LSTM model	2021-08-23	0.82

Table 2: R2 score of experiments 1-4

Comparing the results of experiment 1 and 2, we found that the conditional training/prediction method

is more accurate than normal prediction, since the greed model has higher r2-score than the fear model when predicting the order book data sourcing from a greed market. The results of experiment 1 and 3 implies that, although the GRU model has less complex structure and fewer parameters, its performance on forecasting time-series is not worse than LSTM model. The experiment 4 obtained the highest r2-score, which indicates that the attention layer actually improve the prediction performance of the LSTM model.

However, our model is not robust actually. In experiment 5, we use the prediction model 1, which have a good performance on predicting the mid-price on 2021-08-23, to predict the mid-price on 2021-07-31, but the final r2-score of the prediction is only 0.18. Additionally, in the previous 5 experiments, we set the $n\_timestamp = 300$, which only predict the mid-price after about 10 seconds. Thus, we trained another LSTM model with $n\_timestamp = 1800$, and use the model to predict the mid-price on 2021-09-29, finally got a r2-score of only 0.04. The prediction results is shown in figure 14.



(a) experiment 5

(b) experiment 6

Figure 14: Results of experiments 5 and 6

Due to the lack of data source, we only got the tick-level order book data of BTCUSTD from the cryptocurrency exchange Binance, and we did not carry out experiments on other cryptocurrency. However, from our current experiments result, we can conclude that our prediction model is not robust enough. The prediction may be more accurate when the mid-price is growing in a relatively stable trend, and be less reliable when the fluctuation of the mid-price is notable. Additionally, if we require the model to predict the mid-price in the further future, the prediction will be more tricky as well.

3.5 Trading

3.5.1 Trading Strategies

Two trading strategies are implemented respectively:
The most easy strategy is to simply buy and sell using a constant amount of money.

- If the predicted price increase above the positive threshold, then a buy order is submitted. If the current position is short (sell), then we will close all position;
- If the predicted price decrease below the negative threshold, then sell order is submitted If the current position is buy (long), then we will close all position;
- Else, no trading transaction will be submitted.

The second strategy is that the trading amount is proportional to the predicted price change percentage. Suppose the predicted price change percentage equals to 0.1%, then we will use 10% of total cash in this

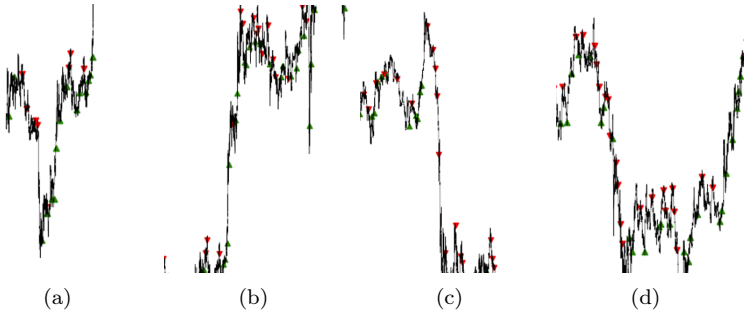
transaction. And we will not close all the positions at once if an opposite sign is detected, we will only close a portion of the current position, and this portion is also proportional to the predicted price change percentage.
This strategy is more flexible and closer to the real life situation. But it also adds more difficulty at risk management.

3.5.2 Backtesting

The backtesting results for two strategies are shown as below:
For the simple strategy, we first use 2021 August 21st as the training data set, and back test on 2021 August 22nd. These two days all have great market performance, and our model can submit trading orders accurately.

Figure 15: Simple strategy backtesting trading activities

It's worth noting that our model can detect those extreme cases when acute increase and decrease very accurately. In the graph below, the red triangles indicate to sell and the green triangles indicate to buy.



(a) (b) (c) (d)

Figure 16: Extreme cases in predicting

And the daily return of our model equals to 11.72%, if everytime we trade we use all the money we have. And the return graph is shown as below:

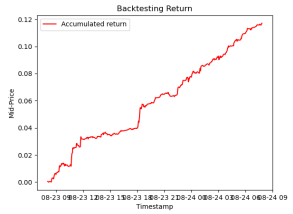


Figure 17: Simple Strategy Backtesting Return

Below is the comparison of the second and more realistic trading strategy, since the type of trading activities is the same, only the trading amount is different. So, the trading activity graph is the same as Figure 15 shown previously, but the return is better than Figure 17.

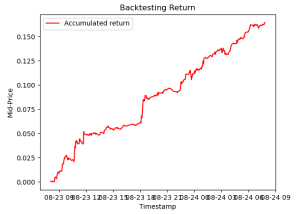


Figure 18: Flexible Strategy Backtesting Return

There is a summary of the trading activity accuracy, as we call see the accuracy are both bigger than 60%, so it can generate profit for sure.

Return	11.71% (Simple), 15.8% (Flexible) %
Maximum drawdown	4.5% (Simple), 3.5% (Flexible)
Trading accuracy	62.6%
Buying accuracy	64.5%
Selling accuracy	61.8%

3.5.3 Live Trading

Quite a lot of effort has been spent on live trading, we have finished the code implementation of it, but after testing for several times, we have to sorrily draw the conclusion that about 50% of transactions can't be completed as expected.

The ideal design of this trading system is as below:

- Using Webscoket framework to get real-time order book data from the exchange
- Input the data we just got into our model, get a predicted value at the end of the upcoming interval
- If the absolute predicted price change percentage is bigger than the threshold, then we submit a market order as soon as possible to the exchange

The first problem is the data inconsistency among different exchange platforms. A lot of crypto currency exchanges have restricted Mainland and Hong Kong users' trading activity, e.g. Binance, Houbi and FTX.Even though we got our order book data from Binance when our project starts at 2021 September, but now Binance has stopped our API for future trading. So we switch to a new exchange platform called Bitget, and this lead to the first problem of "data inconsistency", because each exchange platform's trading activity is different, so the orders in each market will differ a little in the range of -0.05% - 0.05%. This problem is mainly due to the policy restriction, we can solve this problem is to buy the targeted exchange's data again or collect by ourselves.

The second problem is the time lag between "input data" and "output transaction". For example, we fetch data at 00:00am, we use our model to predict the price change after n timestamps, suppose it's 00:01am. Because we can only submit a market order, which means that we can only buy or sell at the current market price. The sooner we submit the transaction, the larger the price gap will be; vice versa, the later we submit the transaction, the room of profit is smaller, if the profit percentage is smaller than the transaction fee percent, we should not submit the order. However, we tested this procedure of dozens of times, in our local computers and using the exchange platform's api, none of thses transactions are opened and closed as soon as we expected.

4 Conclusion

In the first term of this final year project, starting from SVM model of binary classification, continue to multi-class SVM of multiple classes, we built different models and tested their performance on testing data sets. We got very successful accuracy of 97% in the model of using single time point, but failed in predicting using sliding windows.

In the second term, we learned from the previous shortcomings, change our mindset from "classification" to "regression", try to use LSTM and GRU to predict the future bitcoin price change based on the features we selected in the first term. We kept learning new technologies in during this project, we cooperate the "attention layer" into our model, and got a slight improvements in our model performance.

Although achiving $R^2 = 0.8$ on testing data set is hard and we could only get close to that, the model that we built is consistently making profit and outperform the market, with considering the real life situation e.g. the transaction fees etc. But we have to admit that this model is not robust at all situations, it's sensitive to market performance similarity of the training and testing datasets.

4.1 Labor division

The labor division of this project is below: For 11155107895:

- Improved LSTM SVM models
- Implemented the AT-LSTM model
- Implemented backtesting and live trading
- Implemented the GUI of this system

For 115124573:

- Implemented the main structure of LSTM SVM models
- Conducted the experiments of LSTM SVM models
- Implemented the data pre-process part

4.2 Future Work

So far, we only use the order book dataset of BTCUSDT from the cryptocurrency exchange Binance to train and examine the performance of our prediction data, which lacks robustness. Therefore, in our future work, we can tried to find more reliable dataset of different cryptocurrency type from diverse cryptocurrency exchanges.

Essentially, the biggest shortage of our machine learning model is that we did not extract enough features estimating the individual sentiment, traders' behaviors, political events, natural hazards, and many other factors that affecting the fluctuation of the cryptocurrency price. We think the "conditional training/prediction" turned out to be an inspiring attempt, although the improvement is not that obvious. In the future work, we can search for more useful information from various websites, news, and so on, to improve the diversity of our "conditional training/prediction".

5 Reference

[1] Kercheval, A., & Zhang, Y. (2015). *Modelling high-frequency limit order book dynamics with support vector machines*. Quantitative Finance, 15(8), 1315-1329.

[2] Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). *Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks*. 2017 IEEE 19th Conference on Business Informatics (CBI), 1, 7-12.

[3] Karevan, Z., & Suykens, J. (2020). Transductive LSTM for time-series prediction: An application to weather forecasting. Neural Networks, 125, 1-9.

[4]Ayoobi, N., Sharifrazi, D., Alizadehsani, R., Shoeibi, A., Gorriz, J., Moosaei, H., . . . Mosavi, A. (2021). Time series forecasting of new cases and new deaths rate for COVID-19 using deep learning methods. Results in Physics, 27, 104495.

[5] Patle, A., & Chouhan, D. (2013). SVM kernel functions for classification. 2013 International Conference on Advances in Technology and Engineering (ICATE), 1-9.

[6] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[7] Zhao, Y., Liang, Z., Du, J., Zhang, L., Liu, C., & Zhao, L. (2021). Multi-Head Attention-Based Long Short-Term Memory for Depression Detection From Speech. Frontiers in Neurorobotics, 15, 684037.

[8] Zhang, X., Liang, X., Zhiyuli, A., Zhang, S., Xu, R., & Wu, B. (2019, July). At-lstm: An attention-based lstm model for financial time series prediction. In *IOP Conference Series: Materials Science and Engineering* (Vol. 569, No. 5, p. 052037). IOP Publishing.

[9] G. (2021, June 9). *Crypto Quant: programmatic trading of BTC using Binance and Backtrader — Part 2 of 3*. Medium. [Online]. https://medium.com/@gk_/crypto-quant-programmatic-trading-of-btc-using-binance-and-backtrader-part-2-of-3-d8af44c93e2b. [Accessed: Dec. 1st, 2021].

[10] Fee Rate. (2021). Binance. [Online]. Available: https://www.binance.com/en/fee/schedule. [Accessed: Dec. 1st, 2021].