Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Gradient Boosting Algorithms versus Deep Neural Networks for Bitcoin Price Prediction

Author:
Boyu Han (CID: 02094329)

Supervisor:
Dr. Paul A. Bilokon

Second Supervisor:
Dr. Arthur Gervais

A thesis submitted for the degree of
MSc in Computing (Artificial Intelligence and Machine Learning), 2021-2022

August 2022

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgments

I would like to take this opportunity to express my heartfelt appreciation to my supervisor, Dr Paul A. Bilokon, for his unwavering support, encouragement, and patience throughout my thesis. His extensive knowledge and broad perspective in the field of High-frequency Trading have benefited me significantly. The progress that I have made is the result of his profound concern and selfless devotion.

I am also greatly indebted to my personal tutor, Prof. Kin Leung, and all the staff in the Department of Computing, who have helped me directly and indirectly in my studies, their devoted teaching and enlightening lectures brought considerable benefits to me and provided the thesis with academic foundation.

Also, I would like to express my gratitude to my parents, who care about me all the time, and my friends Qinjie, Yitao, and many others who offered me suggestions for this piece of work and motivated me to work harder.

Abstract

The cryptocurrency price forecasting is of vital importance in financial markets. For a long time, the topic that statistical machine learning and deep neural networks are perform better in the prediction of price has become a controversial. At present, gradient boosting, random forest, and LSTMs remain the dominant architecture in the prediction of high-frequency time series prices. However, due to the 24/7 trading policy, BTC, the most famous cryptocurrency, is faced with marked fluctuations. Thus, the forecasting power of classical models is weakened. Given this, the thesis proposes a novel deep learning model, TCN, to forecast Bitcoin's next 100-step difference prices. Two gradient boosting forecasting algorithms (XGBoost and LightGBM) are applies as a benchmark, and two alternative deep learning models (LSTM and TCN) are put forward to compare the performance of different methods. Analysing the real-world Bitcoin trading data, the TCN is contrasted with other state-of-the-art price forecasting models. The results show that the TCN has a better performance in prediction; the R^2 scores 28.9%. This demonstrates that its performance on price forecasting by 15% better than statistical machine learning trading algorithms. To assist related work, the code is available at https://github.com/simon6379/cryptocurrency_prices.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The cryptocurrency is defined as a digital or virtual currency used as a mode of exchange and transfer of assets digitally. It uses cryptography to transfer assets in a secure way, controls and regulates the addition of cryptocurrencies, and completes their transactions (Garcia et al. (2014)), hence the name cryptocurrency. The cryptocurrencies are established on the principle of decentralized control when compared with standard currencies, which rely on the central banking systems. It pushes the high utility of the digital currencies. What can be clearly seen in Figure 1.1 is that shows the global search trends in the field of cryptocurrency during the period 2012–2022 (https://trends.google.com). Essentially, the decentralization behavior of a cryptocurrency has reduced the interference of a central entity. Therefore, the cryptocurrency market has evolved exponentially in the past ten years owing to its uncontrolled and untraceable nature.

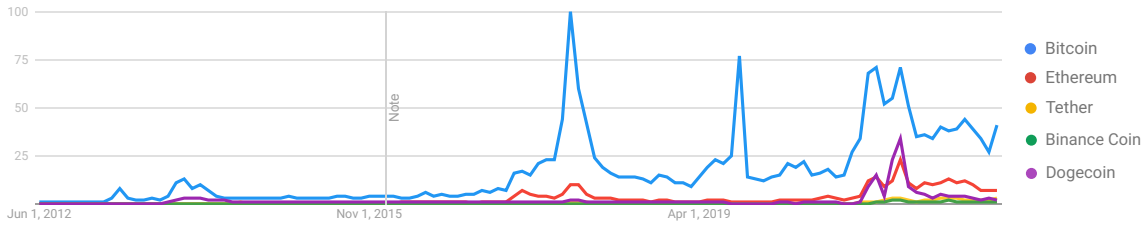

Figure 1.1: Global search trends in the field of cryptocurrency

Bitcoin (BTC) is proposed by Satoshi Nakamoto in Bitcoin: A Peer-to-Peer Electronic Cash System (Nakamoto (2008)). Especially after the boom and bust of cryptocurrencies prices in recent years, BTC has been increasingly regarded as an investment asset. Developing from a virtual currency without market capitalization, BTC has developed into the world's biggest cryptocurrency in circulation within 10 years with a market valuation topping $100 billion. The BTC market opens 24/7, and its price data can be regarded as a time series. That is, it changes with the demand and supply factors at different times. Therefore, wide fluctuations in cryptocurrency prices motivate the urgent requirement for an accurate model to predict its price and analyze its impact on the real life.

Unlike the stock market, the cryptocurrency suffers no effects from seasonality, which makes it hard to predict with a statistical approach (Rathore et al. (2022)). Traditional time series models represented by ARIMA and Holt-Winters are simple to implement and interpret, but they require a lot of statistical assumptions. It leaves machine learning as a more competitive technology in this field, because machine learning is capable of predicting prices based on experience. The main purpose of this thesis is to predict the future 100-step difference price of BTC by comparing the accuracy of boosting algorithms and deep learning methods. The thesis has been organized in the following way: Chapter 2 gives a brief review of the related studies about different trading algorithms on price forecasting in the cryptocurrency market. Chapter 3 describes the experimental approaches in the gradient boosting and deep learning models. Specifically, XGBoost, LightGBM, LSTM and TCN models are introduced. Exploratory analysis is carried out in the Chapter 4 on the BTC time series data, including correlation analysis, bid and ask size, bid and ask price, etc. In Chapter 5 illustrates the methodologies of feature augmentation based on the original features. The details of the four models and related experimental results are demonstrated in Chapter 6 and 7. Chapter 8 draws a conclusion on this thesis and includes a discussion of the implication of the findings to future research.

Chapter 2

Literature Review

The prediction of High-Frequency Trading (HFT) is a dominant topic among researchers worldwide. This section reviews several related studies that aim at the comparative analysis of the prediction of BTC price. Some comprehensive studies show that the Ensemble Learning is an effective approach to predict the price of the cryptocurrency. Guo et al. (2018) conduct a study about the BTC short-term volatility forecasting by extracting the order book features, such as volume, depth, spread, and slope for bid and ask sides. The research indicates that the performance of XGBoost is significantly better than that of Random Forests, and the XGBoost achieves the most remarkable accuracy among all models. However, the XGBoost is more complicated than random forests in tuning and have more chances of overfitting in the case of data sensitivity. More recently, Huang et al. (2019) utilize 124 input variables in five categories including overlap study indicators, momentum indicators, cycle indicators, volatility indicators, and pattern recognition indicators to predict daily BTC returns via tree-based prediction models. The result points to the acceptance of this approach in the markets of assets with hard-to-value fundamentals.

Although machine learning algorithms become a more efficient predictor models in analyzing non-linear historical time-series data, some researchers maintain the idea that traditional statistical learning methods perform more competently. Khedmati et al. (2020) compares AutoRegressive Integrated Moving Average (ARIMA), Bayesian Regression, and Support Vector Machine (SVM) to forecast the BTC maximum, minimum, and opening daily price. Based on RMSE and MAPE measures, the results prove that the SVM gives the most satisfactory performance among all models. However, the ARIMA outperforms other univariate models. One of the notable features of BTC is the extreme volatility. To provide a constructive decision-making analysis and risk management for BTC investors, Shen et al. (2021) demonstrate that although the Recurrent Neural Network (RNN) outperforms conventional econometric models in average forecasting performance, the Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) and the Exponentially Weighted Moving Average (EWMA) provide a better feature in explaining the extreme volatility. The results suggest again that the machine learning approaches are not always more advanced than statistical models, which is contrary to the common belief.

To model the nonlinear relationship between BTC price and social sentiment data, Pang et al. (2019) compare neural networks (NN) and ensemble algorithms based on decision trees to predict the price of BTC with some lead time. The analysis indicates that the neural network models are robust and offer a higher accuracy level in the prediction of the BTC prices. In addition, Ji et al. (2019) take 18 most significant features into consideration (such as 'difficulty', 'est-trans-vol-usd', 'hash-rate', 'trade-vol', etc.) and conduct a comparative study on the prediction of the BTC prices. By employing Deep Neural Network (DNN), LSTM, Deep Residual Network (DRN), and their combinations, they find that LSTM-based prediction models slightly outperform the other prediction models for Bitcoin price prediction (Regression), while DNN-based models perform the best for price ups and downs prediction (Classification).

There are several papers that introduce some innovative stock volatility models based on the TCN. The TCN model is a framework that employs casual convolutions and dilations and it is adaptive for sequential data with its temporality and large receptive fields. One of the latest studies shows that this approach has currently provided enormous improvements for the cryptocurrency price prediction of task, and achieved state-of-the-art results in terms of accuracy. Based on the inter-exchange transactions data, inner-exchange market prices data and social interest data, Guo et al. (2021) propose a price forecasting model WT-CATCN, which leverages Wavelet Transform (WT) and Casual Multi-Head Attention (CA) TCN, to forecast the cryptocurrency prices. The model trained gives an impressive MSE of 0.55 and improves the price forecasting performance by 25%. To date, however, there has been little convincing evidence that the TCN is applicable in the forecasting of time-series with noisy data. Therefore, one of the targets of this thesis is to compare the performance of the TCN and the existing state-of-the-art time-series model, such as LSTM on the BTC data.

Chapter 3

Methodology

3.1 Gradient Boosting Methods

Tree-based models apply a decision tree to learn attribute-class relationships, which are fast to train and well interpretable. However, unpruned, single decision trees are prone to overfit the training data. Gradient boosting (Breiman (1996); Friedman (2001)) is a sequential ensemble method for improving out-of-sample prediction. It provides a feasible way to improve the predictive performance of a given model fitting technique by constructing a linear combination of some fitting methods, instead of using a single fit of the method. Boosted trees have been proven to be powerful prediction machines, which often perform well in the competitions of prediction.

To understand the boosted trees, the basic structure of a decision tree is introduced. The decision tree is a simple nonparametric method, which builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is that a tree with decision nodes and leaf nodes is built. In the process of growing a tree, we are faced with the bias-variance tradeoff to balance the prediction accuracy and the generalization performance of the learning algorithm. Due to the fact that the cryptocurrency returns are extremely noisy, it is necessary to guard against over-responding to the substantial noise in the data when making out-of-sample predictions.

3.1.1 Extreme Gradient Boosting (XGBoost)

Similar to gradient boosting, Extreme Gradient Boosting (XGBoost), as a decision-tree-based ensemble Machine Learning algorithm, builds an additive expansion of the objective function by minimizing a loss function (Chen and Guestrin (2016)). In the prediction that involve structured or tabular data, decision tree-based algorithms

are considered the best-in-class currently. This approach additively fits $f(x_t)$:

$$f(x_t) = \sum_{m=1}^{M} f_m(x_t). \tag{3.1}$$

where f_m represents an independent regression tree and $f_m(x_t)$ denotes the prediction score given by the m^{th} tree to the t^{th} sample. Each function $f_m(x_t)$ in $f(x_t)$ is a relatively simple model. While "weak" learners help to guard against overfitting, they are more likely to suffer from biases and thus poor fit is evinced. To improve the fit of the model, the key perspective of Boosting methods is to add an $f_m(x_t)$ element that explains the residuals from the previous functions in the sequence. To be more specific, the prediction value at step q is noted as $\hat{r}_{t+1}^{(q)}$. For step q,

$$\hat{r}_{t+1}^{(0)} = 0, \tag{3.2}$$

$$\hat{r}_{t+1}^{(1)} = f_1(x_t) = \underbrace{\hat{r}_{t+1}^{(0)}}_{0} + f_1(x_t), \tag{3.3}$$

$$\hat{r}_{t+1}^{(2)} = \underbrace{f_1(x_t)}_{=\hat{r}_{t+1}^{(1)}} + f_2(x_t), \tag{3.4}$$

$$\hat{r}_{t+1}^{(3)} = \underbrace{f_1(x_t) + f_2(x_t)}_{=\hat{r}_{t+1}^{(2)}} + f_3(x_t), \tag{3.5}$$

$$\vdots \tag{3.6}$$

$$\hat{r}_{t+1}^{(q)} = \sum_{m=1}^{q} f_m(x_t) = \underbrace{f_1(x_t) + \ldots + f_{q-1}(x_t)}_{=\hat{r}_{t+1}^{(q-1)}} + f_q(x_t). \tag{3.7}$$

The model ought to be trained, the objective function based on a loss function and a regularization term needs to be minimized:

$$\mathcal{L} = \sum_{t=1}^{T-1} l(r_{t+1}, \hat{r}_{t+1}) + \sum_{m=1}^{M} \Omega(f_m(x_t)), \tag{3.8}$$

The $l(r_{t+1}, \hat{r}_{t+1})$ is the loss function and $\Omega(f_m(x_t))$ is a function that gives a penalty for a model with higher complexity. The objection function for step q can be rewritten as:

$$\mathcal{L}^{(q)} = \sum_{t=1}^{T-1} l\left(r_{t+1}, \hat{r}_{t+1}^{(q)}\right) + \sum_{m=1}^{q} \Omega(f_m(x_t)) \tag{3.9}$$

$$= \sum_{t=1}^{T-1} l\left(r_{t+1}, \hat{r}_{t+1}^{(q-1)} + f_q(x_t)\right) + \Omega(f_q(x_t)) + \text{constant}. \tag{3.10}$$

Generally speaking, the decision trees are sensitive to the data on which they are trained; a subtle change in the data causes a distinct difference in the structure of the

decision tree, which contributes to instability in the task of prediction. Furthermore, the training cost of the decision tree is relatively considerable because it is more complex and takes a long time. In this case, it will easily involve a substantial risk of overfitting.

As a bias-reduction technique, XGBoost typically improves the performance of a single tree model to address the weaknesses mentioned above. Therefore, XGBoost is built as a more effective solution by taking a linear combination of trees. In presence of high-dimensional predictors, this approach is also highly applicable as a regularization technique for additive or interaction modelling.

Algorithm 1 Histogram-based Algorithm

Input: I: training data, d: max depth
Input: m: feature dimension
$nodeSet \leftarrow \{0\}$▷ tree nodes in current level
$rowSet \leftarrow \{\{0, 1, 2, \ldots\}\}$▷ data indices in tree nodes
for $i = 1$ ***to*** d **do**
  for *node* ***in*** *nodeSet* **do**
    usedRows$\leftarrow rowSet$[node]
    for $k = 1$ ***to*** m **do**
      $H \leftarrow$new Histogram()
      ▷ Build histogram
      for j ***in*** *usedRows* **do**
        bin$\leftarrow I$.f[k][j].bin
        H[bin].y$\leftarrow H$[bin].y+I.y[j]
        H[bin].n$\leftarrow H$[bin].n+1
        Find the best split on histogram H.
      end for
    end for
  end for
  Update $rowSet$ and $noteSet$ according to the best split points.
end for

3.1.2 Light Gradient Boosting Machine (LightGBM)

Light Gradient Boosting Machine (LightGBM) (Ke et al. (2017)) is the newly introduced framework of gradient boosting which is built on the base of decision trees and it is known for its efficiency and low computational complexity. This algorithm inherits most of advantages of the XGBoost, including parallel training, multiple loss functions, regularization, etc. But a major difference between the two lies in the construction of the trees. LightGBM is capable of growing the tree vertically, while other algorithms grow trees in a horizontal way. Such merit makes LightGBM an effective method for processing large-scale data and features. Besides, LightGBM implements a highly optimized histogram-based decision tree learning algorithm

(refer to Algorithm 1), which yields great advantages in both efficiency and memory consumption. Therefore, LightGBM proposes a novel technique — Gradient-based One-side Sampling (GOSS) to achieve the targets above.

Algorithm 2 Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models$\leftarrow$ {}, fact$\leftarrow$ $\frac{1-a}{b}$
topN$\leftarrow$a$\times$len(I), randN$\leftarrow$b$\times$len(I)
for $i = 1$ ***to*** d **do**
preds$\leftarrow$models.predict(I)
g$\leftarrow$$loss(I,$ preds$)$, w$\leftarrow$ $\{1, 1, \ldots\}$
sorted$\leftarrow$GetSortedIndices(abs(g))
topSet$\leftarrow$sorted[1:topN]
randSet$\leftarrow$RandomPick(sorted[topN:len(I)],randN)
usedSet$\leftarrow$topSet+randSet
w[randSet]$\times$ =fact$\triangleright$Assign weight $fact$ to the small gradient data.
newModel$\leftarrow$L$(I[usedSet], -g[usedSet], w[usedSet])$
models.append(newModel)
end for

Typically, instances with larger gradients will contribute more to information gain, compared to the ones with smaller gradients. Thus, to retain the accuracy of the information, GOSS keeps the instances with large gradients and removes the smaller ones. However, this will cause the data distribution to be distorted and reduce the prediction accuracy if all small gradient data are directly removed. Therefore, as shown in Algorithm 2, LightGBM sets a sampling ratio a to select the top ones, and randomly drops the instances with small gradients by using a ratio b. To compensate for the data loss, LightGBM applies a constant $\frac{1-a}{b}$ to amplify the instances with small gradient when calculating the information gain. The whole algorithm can be described as below.

3.2 Deep Learning Methods

3.2.1 Long Short-Term Memory Networks (LSTMs)

Recurrent Neural Networks (RNN) can use their feedback connections to store representations of recent input events in form of activations. However, this algorithm for learning what to put in short-term memory have obvious drawbacks with relatively low efficiency and accuracy, especially when minimal time lags between inputs and corresponding output signals are long. To minimize the weakness, some mechanisms

for storing vital early information in a memory cell is applied. For learning issues involving long-term sequential data, Long Short-Term Memory (LSTM) has emerged as an effective and scalable approach (Hochreiter and Schmidhuber (1997)). Thus, it is highly applicable for capturing long-term temporal dependencies since the fact that their networks are looped, which allows information to persist (Lindemann et al. (2021)).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The data feeding into the LSTM gates are the input at the current time step and the hidden state of the previous time step. They are processed by three fully connected layers with a sigmoid activation function to compute the values of the input. Formally, it is supposed that there are h hidden units, the batch size is n, and the number of inputs is d. Thus, the input is $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ and the hidden state of the previous time step is $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$. Correspondingly, the gates at time step t are defined as follows: the input gate is $\mathbf{I}_t \in \mathbb{R}^{n \times h}$, the forget gate is $\mathbf{F}_t \in \mathbb{R}^{n \times h}$, and the output gate is $\mathbf{O}_t \in \mathbb{R}^{n \times h}$. They are calculated as follows:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i), \tag{3.11}$$

$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f), \tag{3.12}$$

$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o). \tag{3.13}$$

The $\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho} \in \mathbb{R}^{h \times h}$ are weight parameters and $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o \in \mathbb{R}^{1 \times h}$ are bias parameters. In order to update the state, a $\tanh$ layer creates a vector of new candidate values, $\tilde{\mathbf{C}}_t$, that could be added to the state (Figure 3.1). The details of Candidate Memory Cell $\tilde{\mathbf{C}}_t$ as follow:

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c), \tag{3.14}$$

where $\mathbf{W}_{xc} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{ho} \in \mathbb{R}^{h \times h}$ are weight parameters and $\mathbf{b}_c \in \mathbb{R}^{1 \times h}$ are bias parameters.

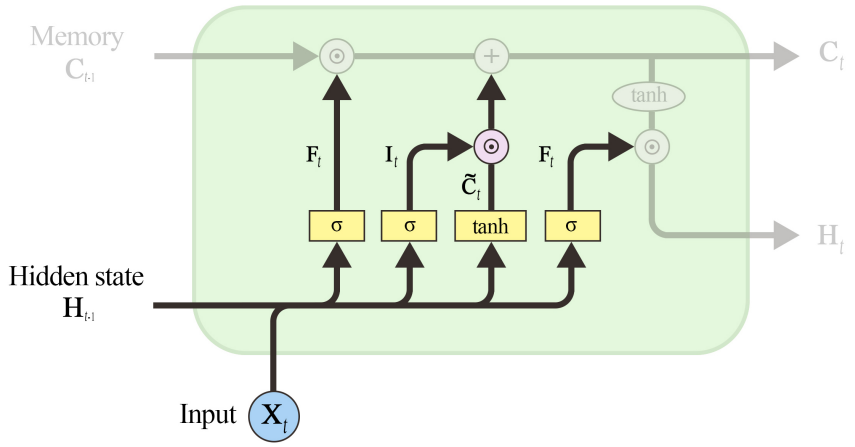

Figure 3.1: Computing the candidate memory cell in an LSTM model

Based on the above-mentioned components, we use the input gate $\mathbf{I}_t$ governing how much we take new data into account via $\tilde{\mathbf{C}}_t$, and the forget gate $\mathbf{F}_t$ addressing how much of the old memory cell content $\mathbf{C}_{t-1} \in \mathbb{R}^{n \times h}$ we retain. Thus, we arrive at the following updated equation:

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t, \tag{3.15}$$

and a graphical illustration of the data flow up to the present is shown in Figure 3.2.

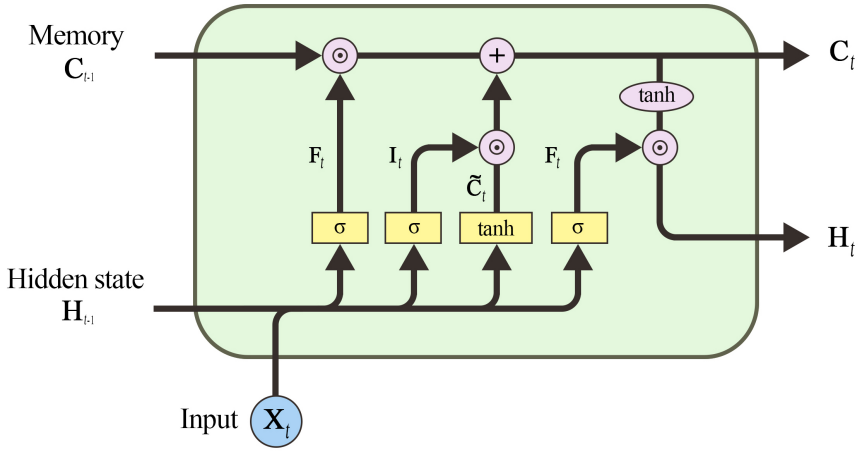

Figure 3.2: Computing the memory cell in an LSTM model

Finally, we need to use the output gate to compute the hidden state $\mathbf{H}_t \in \mathbb{R}^{n \times h}$. It is simply a gated version of the tanh of the memory cell,

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t) \tag{3.16}$$

Whenever the output gate approximates 1 we effectively pass all memory information through to the predictor, whereas for the output gate close to 0 we retain all the information only within the memory cell and perform no further processing. The whole chain structure of LSTM is shown in Figure 3.3.

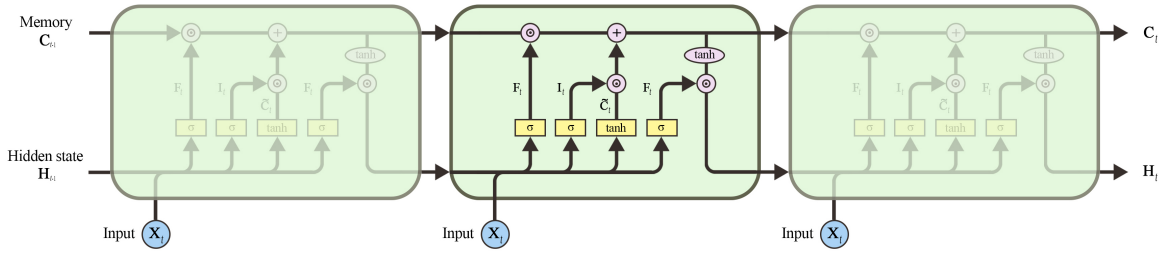

Figure 3.3: The detailed internals of an LSTM

3.2.2 Temporal Convolutional Network (TCN)

TCN is known as a type of convolutional neural network, which is applied in many time-series forecasting tasks, such as action segmentation (Lea et al. (2016)). It is deliberately kept simple, combining some of the best practices of modern convolutional architectures (Bai et al. (2018)). TCN has two specific designs, the first one is 1D fully-convolutional network (FCN) architecture, where each hidden layer has the same length as the input layer, and zero padding of length (kernel size-1) is added to keep the network outputting the same length as the input sequence (Long et al. (2014)). The outputs are only influenced by information of the present and past inputs in each layer by using **causal convolutions**. Causal convolution is different from standard convolution in that the output at time t is not convolved with future values (Wan et al. (2019)).

However, simple causal convolution is not competent in traditional convolutional neural networks, and the modeling length of time is limited by the size of the convolution kernel (Oord et al. (2016)). In this case, if longer dependencies between data need to be learned, stacking many layers linearly will be necessary. To solve this problem, the TCN uses one-dimensional dilated convolution. The difference between **dilated convolution** and traditional convolution is that it allows the input of convolution to have interval sampling. Without the pooling operation, this convolution increases the receptive fields of the network, and is adaptive for sequential data with its temporality, thus there is no loss of resolution. More formally, for an input sequence $\mathbf{x} \in \mathbb{R}^T$ and a filter $h : \{0, 1, \ldots, k-1\} \to \mathbb{R}$, the dilated convolution operation $\mathbf{H}$ on the element $\mathbf{x}$ of the sequence is defined as

$$\mathbf{H}(x) = (\mathbf{x} *_d h)(x) = \sum_{i=0}^{k-1} f(i)\mathbf{x}_{s-d \cdot i} \tag{3.17}$$

where $d = 2^v$ is the dilation factor, with v the level of the network; k is the filter size, and the term $s - d \cdot i$ accounts for the direction of the past. Dilation is equivalent to introducing a fixed step between every two adjacent filter taps, this can be seen in the following Figure 3.4:

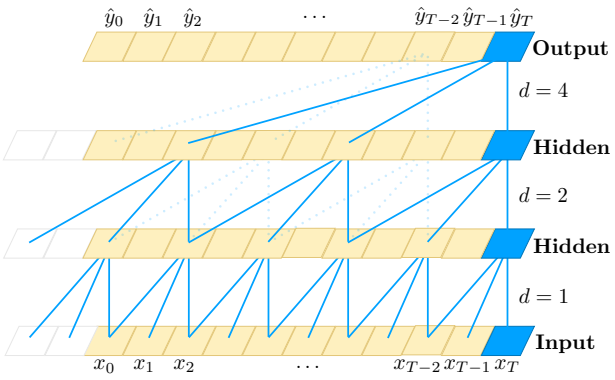

Figure 3.4: The structure of TCN dilated causal convolution (Bai et al. (2018))

Another architectural element of a TCN is **residual connections**, which have proven to be effective in training deep networks. In a residual network, skip connections are used throughout the network, to speed up training process and avoid vanishing gradient problem in deep models. TCNs employ a generic residual module. Each residual block contains a branch leading out to a series of transformations $\mathcal{F}$, whose outputs are added to the input $\mathbf{x}$ of the block

$$o = \text{Activation}(\mathbf{x} + \mathcal{F}(\mathbf{x})) \tag{3.18}$$

This effectively allows layers to learn modifications to the identity mapping rather than the entire transformation, which has been shown to benefit deep neural networks (He et al. (2015)). For example, in the case where the prediction depends on a large history size with a high-dimensional input sequence.

As shown in the Figure 3.5, a residual block has two layers of dilated causal convolutions and rectified linear units (ReLU) as non-linearities. Additionally, the weight normalization (Salimans and Kingma (2016)) is also applied to the convolutional filters and a spatial dropout (Srivastava et al. (2014)) is added after each dilated convolution for regularization.

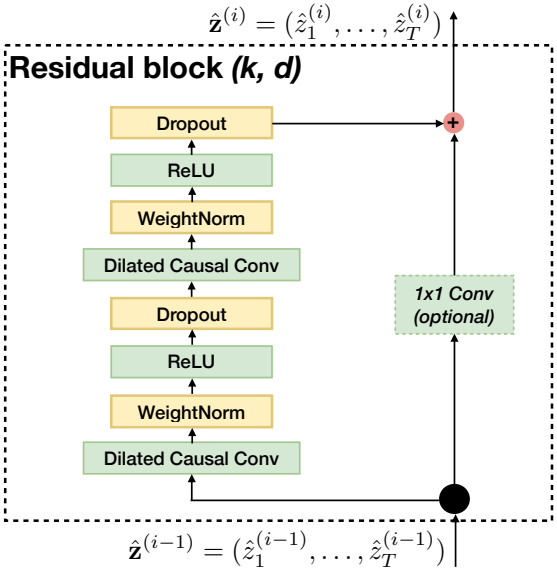

Figure 3.5: The structure of TCN residual block (Bai et al. (2018))

Chapter 4

Exploratory Data Analysis

4.1 Data Acquisition and Preprocessing

We use the official python package provided by BitMEX for data acquisition. It can be downloaded at

https://github.com/BitMEX/api-connectors/tree/master/official-ws/python

We collect the data including market depth, ticker, and recent trade data for research, as represented in Figures 4.1-4.3. It is practical for researchers to register a BitMEX exchange account and acquire more trading information by paying. But the free data is qualified enough for the study.

	symbol	id	side	size	price	timestamp
0	XBTUSD	15500000000	Sell	1000	1000000.0	2022-05-30T12:50:27.507Z
1	XBTUSD	15500000100	Sell	100000	999999.0	2022-05-30T12:50:27.507Z
2	XBTUSD	15502119900	Sell	5000	978801.0	2022-05-30T12:50:27.507Z
3	XBTUSD	15504648350	Sell	2191000	953516.5	2022-05-30T12:50:27.507Z
4	XBTUSD	15515440800	Sell	300	845592.0	2022-05-30T12:50:27.507Z

Figure 4.1: Table structure for Market Depth Data

We have collected data of several days, and there are two points need to be highlighted in the process of data collection: policies and market fluctuations. The IPs of the United States, China, and other countries impose prohibition against BitMEX exchange. It is necessary to use the Virtual Private Network (VPN), which leads to the consequence that the data collection process is greatly affected by the network fluctuations. Some of the data are collected with many missing values. Under

	last	buy	sell	mid
0	30414.0	30402.0	30414.0	30408.0
1	30414.0	30402.0	30414.0	30408.0
2	30414.0	30402.0	30414.0	30408.0
3	30414.0	30402.0	30414.0	30408.0
4	30414.0	30402.0	30414.0	30408.0

Figure 4.2: Table structure for Ticker Data

	timestamp	date	time	symbol	side	size	price	tickDirection	trdMatchID	grossValue	homeNotional	foreignNotional
0	2022-05-30 12:50:16.726000+00:00	2022-05-30	12:50:16.726000	XBTUSD	Buy	100	30413.5	ZeroMinusTick	d51d902e-24cc-4730-71ab-63d761b1d622	328801	0.003288	100
1	2022-05-30 12:50:46.096000+00:00	2022-05-30	12:50:46.096000	XBTUSD	Buy	600	30413.5	ZeroMinusTick	da91c0b6-4174-9df9-5f42-7b189ed742c7	1972806	0.019728	600
2	2022-05-30 12:51:16.676000+00:00	2022-05-30	12:51:16.676000	XBTUSD	Buy	100	30413.5	ZeroMinusTick	f2bb5adb-f01e-0f0c-9f7e-e4dd2f145149	328801	0.003288	100
3	2022-05-30 12:51:16.682000+00:00	2022-05-30	12:51:16.682000	XBTUSD	Buy	100	30413.5	ZeroMinusTick	f5fbb3c3-13ee-1d3d-23dd-4acf53eed3c7	328801	0.003288	100
4	2022-05-30 12:52:16.799000+00:00	2022-05-30	12:52:16.799000	XBTUSD	Buy	100	30413.5	ZeroMinusTick	7ab021e2-e0a4-51a8-78f6-c818c802fb08	328801	0.003288	100

Figure 4.3: Table structure for Recent Trade Data

the influence of the tighter monetary policies of different countries, the price of the BTC fluctuates wildly. As shown in Figure 4.4, from the highest price of $47,686.81 to the lowest of $19,017.64, the price fluctuates considerably, which increased the difficulty of forecasting.

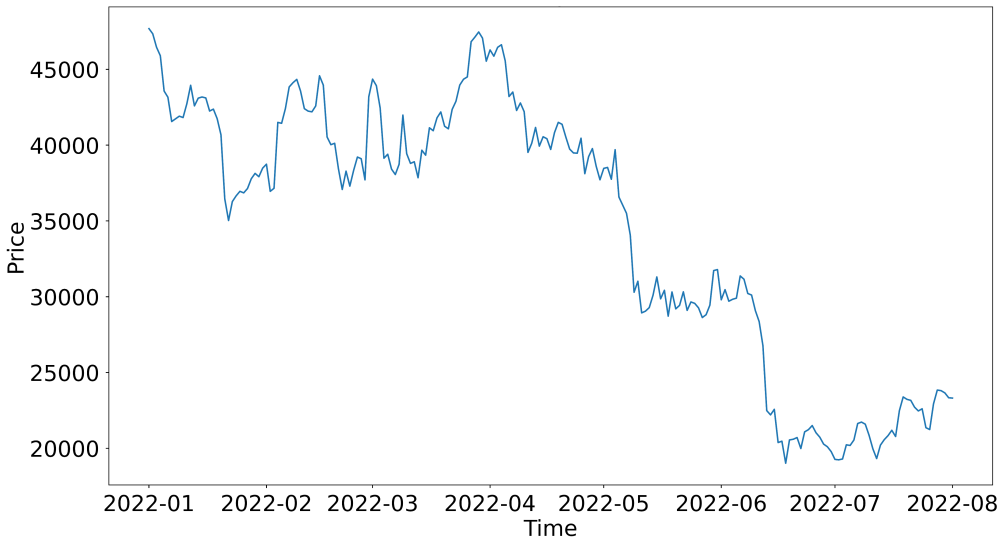

Figure 4.4: Bitcoin price trends in 2022

Through the observation of data, as shown in Figure 4.5, the eight periods of time from May 31st to June 3rd were finally selected as the total data set of the task for the following analysis. The first three days are chosen for training, a half a day is designed for validation, and another half a day for testing.

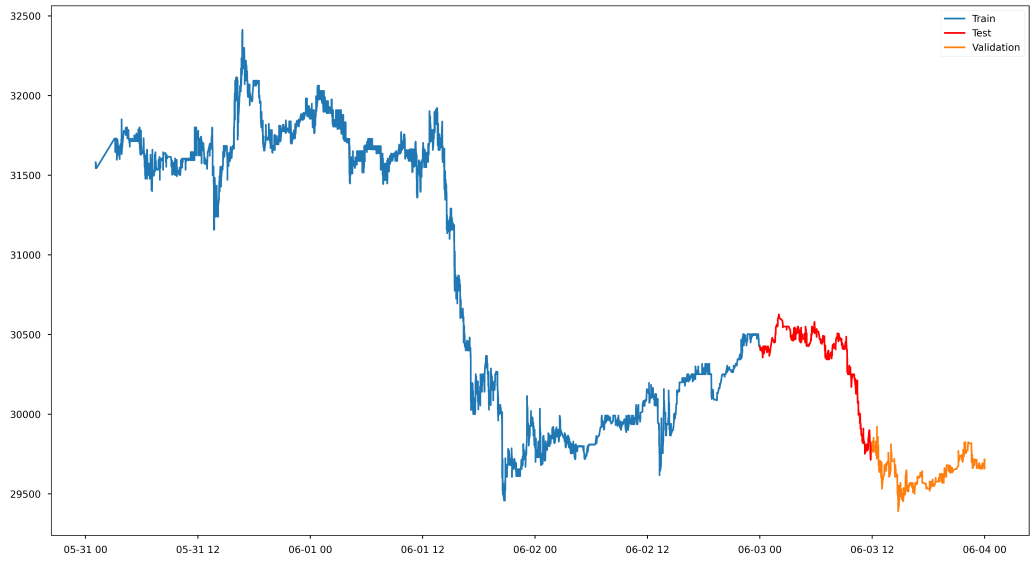

Figure 4.5: Price from 2022-05-31 to 2022-06-03

We use the **merger_asof** command from pandas to merge market depth, ticker, and recent trade data into a whole data frame. The **merger_asof** matches on nearest time rather than equal time, so it is more applicable in time series data tasks. The result is shown in Figure 4.6.

	timestamp	Bid0	Bid1	Bid2	Bid3	Bid4	Bid5	Bid6	Bid7	Bid8	...	Asksize0	Asksize1	Asksize2	Asksize3	Asksize4	Asksize5	Asksize6	Asksize7	Asksize8	Asksize9
0	2022-05-30T12:50:27.507000000	30402.5	30402.0	30380.0	30329.5	30328.5	30320.0	30298.0	30279.0	30267.5	...	4800.0	500.0	5700.0	106400.0	100.0	100.0	106800.0	300.0	100.0	106900.0
1	2022-05-30T12:50:27.507000000	30402.5	30402.0	30380.0	30329.5	30328.5	30320.0	30298.0	30279.0	30267.5	...	4800.0	500.0	5700.0	106400.0	100.0	100.0	106800.0	300.0	100.0	106900.0
2	2022-05-30T12:50:27.507000000	30402.5	30402.0	30380.0	30329.5	30328.5	30320.0	30298.0	30279.0	30267.5	...	4800.0	500.0	5700.0	106400.0	100.0	100.0	106800.0	300.0	100.0	106900.0
3	2022-05-30T12:50:47.119000000	30402.5	30402.0	30380.0	30329.5	30328.5	30320.0	30298.0	30279.0	30267.5	...	4200.0	500.0	5700.0	106400.0	100.0	100.0	300.0	100.0	106900.0	107000.0
4	2022-05-30T12:50:47.119000000	30402.5	30402.0	30380.0	30329.5	30328.5	30320.0	30298.0	30279.0	30267.5	...	4200.0	500.0	5700.0	106400.0	100.0	100.0	300.0	100.0	106900.0	107000.0

Figure 4.6: Table structure for bitcoin quote

4.2 Statistical Analysis & Visualization

We performed exploratory data analysis on the dataset. The following two tables summarize the trade prices and sizes for bid and ask markets separately. Overall, there are no significant differences among the 10 bid prices; the maximum difference in mean between bid0 and bid9 is 103.24, as illustrated in Table 4.1. However, in the seller's market, the mean of ask0 is 169.41 lower than ask9. Additionally, Table 4.2 shows that the average difference in trade size of the buyer's market is 25,276.91, while that of the seller is almost twice that of the former. The evidence demonstrates that the volatility of the seller's market is noticeably higher than the buyers', and further declare that the trading market starts to deflate.

	Bid0	Bid9	Ask0	Ask9
Minimum	29,390.50	29,344.00	29,416.00	29,609.50
Maximum	32,395.50	32,118.50	32,413.00	32,559.00
Mean	30,753.99	30,650.75	30,790.52	30,959.93
Median	30,482.50	30,393.00	30,503.00	30,663.50
Standard deviation	868.88	859.34	879.56	865.26

Table 4.1: Summary statistics of bid and ask price

	Bidsize1	Bidsize9	Asksize1	Asksize9
Minimum	100.00	100.00	100.00	100.00
Maximum	1,000,000.00	500,600.00	1,000,000.00	899,800.00
Mean	20,279.37	45,556.28	28,865.34	86,537.56
Median	200.00	400.00	200.00	107,300.00
Standard deviation	73,322.82	58,745.00	84,380.77	103,101.40

Table 4.2: Summary statistics of bid and ask size

The results of the correlation analysis in Figure 4.7 indicate that the price of the buyer and the seller has the greatest correlation with the final predicted price, followed by the buyer's bid size and the seller's ask size. In particular, Bidsize 6∼9 and Asksize 0∼2 have the closer correlation with the final price.

From the bid price, we know that top buyers bid conservatively, and there is not much possibility of price increase. From bid size, it can be inferred that the scale of most transactions is relatively small. Finally, from the buyer's market summary, the current buying desire and the price are lower, which is also the general trend of the BTC price in this year.

	size	price	grossValue	homeNotional	foreignNotional	Bid0	Bid1	Bid2	Bid3	Bid4	...	Asksize1	Asksize2	Asksize3	Asksize4	Asksize5	Asksize6	Asksize7	Asksize8	Asksize9	price
size	1.000000	0.014166	0.999599	0.999599	1.000000	0.011938	0.011617	0.011419	0.011092	0.010978	...	0.010444	0.010288	0.007391	0.005164	0.001773	-0.011267	-0.000867	-0.004966	-0.016544	0.014166
price	0.014166	1.000000	0.011248	0.011248	0.014166	0.998970	0.998907	0.998831	0.998714	0.998614	...	-0.005212	0.038731	0.000049	0.000611	-0.032920	-0.056959	-0.059626	-0.094942	-0.042148	1.000000
grossValue	0.999599	0.011248	1.000000	1.000000	0.999599	0.009030	0.008690	0.008490	0.008164	0.008046	...	0.011299	0.010586	0.007888	0.005445	0.002293	-0.010439	-0.000148	-0.004322	-0.016510	0.011248
homeNotional	0.999599	0.011248	1.000000	1.000000	0.999599	0.009030	0.008690	0.008490	0.008164	0.008046	...	0.011299	0.010586	0.007888	0.005445	0.002293	-0.010439	-0.000148	-0.004322	-0.016510	0.011248
foreignNotional	1.000000	0.014166	0.999599	0.999599	1.000000	0.011938	0.011617	0.011419	0.011092	0.010978	...	0.010444	0.010288	0.007391	0.005164	0.001773	-0.011267	-0.000867	-0.004966	-0.016544	0.014166
Bid0	0.011938	0.998970	0.009030	0.009030	0.011938	1.000000	0.999854	0.999706	0.999584	0.999491	...	-0.002112	0.039341	-0.000135	0.001823	-0.032185	-0.054586	-0.058547	-0.095397	-0.041237	0.998970
Bid1	0.011617	0.998907	0.008690	0.008690	0.011617	0.999854	1.000000	0.999870	0.999743	0.999653	...	-0.003719	0.038421	-0.001720	0.001226	-0.033062	-0.055386	-0.057810	-0.095538	-0.041156	0.998907
Bid2	0.011419	0.998831	0.008490	0.008490	0.011419	0.999706	0.999870	1.000000	0.999883	0.999789	...	-0.004257	0.037509	-0.003069	-0.000006	-0.033654	-0.055759	-0.057942	-0.095384	-0.040667	0.998831
Bid3	0.011092	0.998714	0.008164	0.008164	0.011092	0.999584	0.999743	0.999883	1.000000	0.999921	...	-0.005204	0.036874	-0.002899	0.000364	-0.034040	-0.055332	-0.057822	-0.095475	-0.040893	0.998714
Bid4	0.010978	0.998614	0.008046	0.008046	0.010978	0.999491	0.999653	0.999789	0.999921	1.000000	...	-0.006063	0.037195	-0.002834	0.000231	-0.033891	-0.055346	-0.058115	-0.095890	-0.041028	0.998614
Bid5	0.010773	0.998526	0.007849	0.007849	0.010773	0.999391	0.999568	0.999712	0.999840	0.999923	...	-0.006221	0.036967	-0.002868	0.000052	-0.034140	-0.055720	-0.058244	-0.095835	-0.041378	0.998526
Bid6	0.010439	0.998444	0.007525	0.007525	0.010439	0.999298	0.999488	0.999627	0.999743	0.999839	...	-0.006446	0.036464	-0.003092	-0.000609	-0.034168	-0.055840	-0.058093	-0.095559	-0.041273	0.998444
Bid7	0.010421	0.998347	0.007509	0.007509	0.010421	0.999220	0.999422	0.999553	0.999662	0.999759	...	-0.006789	0.036275	-0.003129	-0.000502	-0.034225	-0.055590	-0.057684	-0.095254	-0.041010	0.998347
Bid8	0.010228	0.998268	0.007321	0.007321	0.010228	0.999144	0.999356	0.999491	0.999594	0.999684	...	-0.007315	0.035713	-0.003356	-0.000559	-0.034228	-0.055438	-0.057335	-0.094752	-0.040978	0.998268
Bid9	0.009981	0.998217	0.007080	0.007080	0.009981	0.999092	0.999303	0.999438	0.999535	0.999617	...	-0.007117	0.035866	-0.002845	-0.000246	-0.033930	-0.055309	-0.056956	-0.094264	-0.040314	0.998217
Bidsize0	0.018302	0.013124	0.017590	0.017590	0.018302	0.006114	0.008421	0.008212	0.007858	0.008125	...	-0.068057	0.016530	-0.053316	-0.021604	-0.056578	-0.055615	0.023072	0.024068	-0.012538	0.013124
Bidsize1	0.007683	-0.022873	0.007247	0.007247	0.007683	-0.026747	-0.027120	-0.026697	-0.026893	-0.027090	...	-0.028538	-0.004879	-0.002329	0.014546	-0.019688	-0.012895	-0.033252	-0.011847	-0.021502	-0.022873

Figure 4.7: Correlation of the Data frame

From the ask size (Figure 4.11), it can be noticed that the seller price is not stable. The seller's market is dominated by large-scale transactions, indicating that many people display the willingness to sell. However, as shown in Figure 4.10, the top 10 ask price of the order books did not reflect significant differences. It can be seen that the seller is not willing to decrease the price, nor suffer dramatic losses. It is

attributed to the relatively high BTC price ($45,000) in this year. Finally, according to the situation of the seller's market, although sellers want to complete the deal, the selling price is will not drop any further.

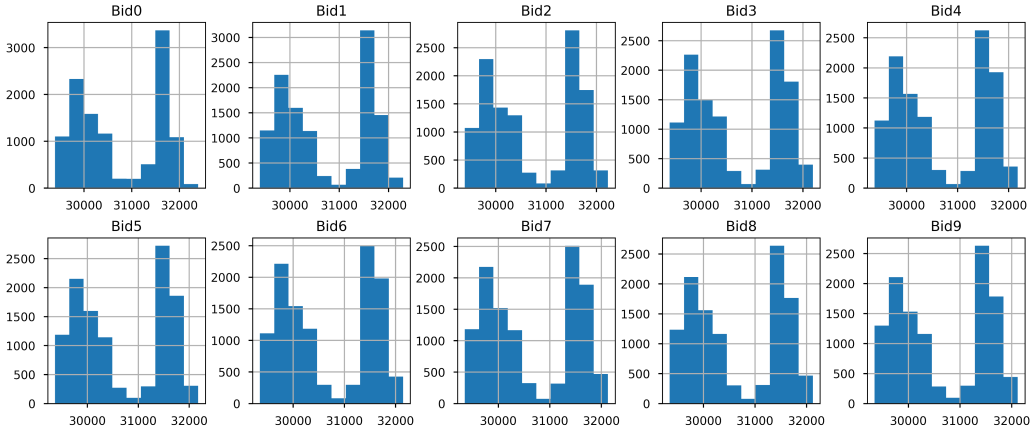

Figure 4.8: The top 10 bid price of the order book

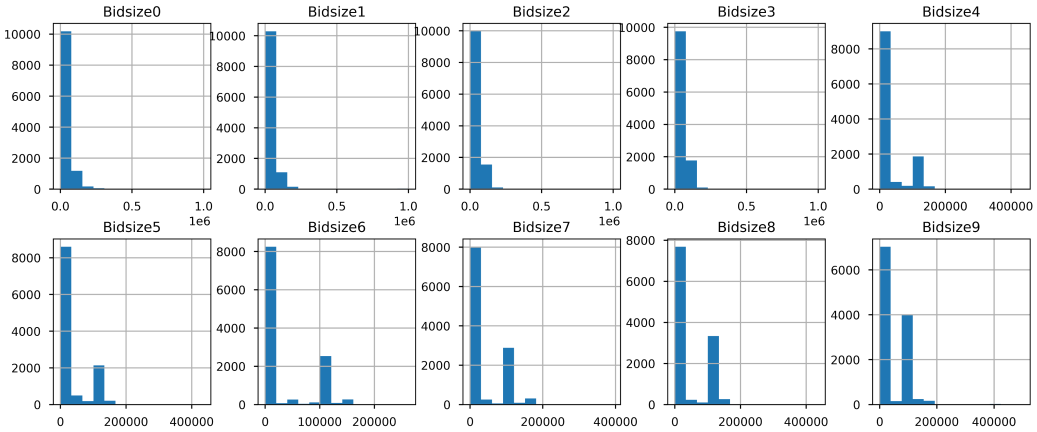

Figure 4.9: The top 10 bid size of the order book

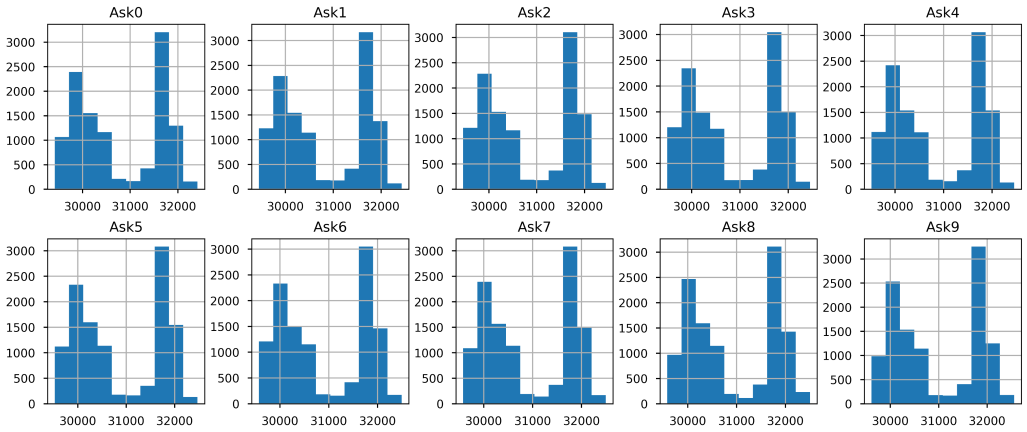

Figure 4.10: The top 10 ask price of the order book

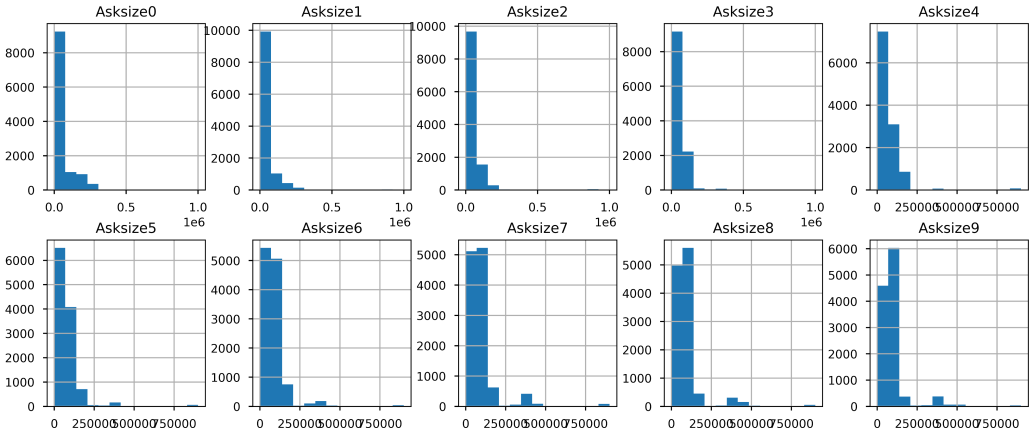

Figure 4.11: The top 10 ask size of the order book

Chapter 5

Feature Augmentation

5.1 Original Features

Our data come from market depth, ticker, and recent trade data and they are provided by the BitMEX. The original data is a simple record of the order book and need to be further explored. To begin with, the thesis introduces the original features:

- Data: **Exchange Timestamp**
- Symbol: Contract Object
- $Bid\{i\}$: The $\{i\}^{th}$ bid price $p_t^{b;i}$ sorted in order book from high to low, $\{i = 1, 2, 3 \dots 10\}$
- $Ask\{i\}$: The $\{i\}^{th}$ ask price $p_t^{a;i}$ sorted in order book from high to low, $\{i = 1, 2, 3 \dots 10\}$
- $BidVol\{i\}$: The $bid\{i\}'s$ volume $V_t^{b;i}$, $\{i = 1, 2, 3 \dots 10\}$
- $AskVol\{i\}$: The $ask\{i\}'s$ volume $V_t^{a;i}$, $\{i = 1, 2, 3 \dots 10\}$
- $Midprice\{i\}$: The average of $p_t^{b;i}$ and $p_t^{a;i}$, $\{i = 1, 2, 3 \dots 10\}$

What we need to do is to predict future prices based on the contents of the trading book, which is a high-frequency trading. There is few data available because many data statistics are delayed and cannot be updated in real-time. In the thesis, we will compare the performance of advanced machine learning methods such as XGBoost, LightGBM, and deep learning methods such as LSTM and TCN for the high-frequency price prediction. In this research, we predict the difference price between midprice from future $t + \tau$ and current t, where $\tau = 1, 2, \dots, 100$.

5.2 Temporal Features

Since the fact that temporal information cannot be processed by the Boosting tree model such as XGBoost and LightGBM, we build such timing features to help them improve their performance.

5.2.1 AskLag

The $ask\{i\}lag\{j\}$ is the lag of $ask\{i\}$ and $ask\{i\}lag\{j\}$ at time t is equal to $ask_{\{i-j\}} = p_{t-j}^{a;i}$, here $i = 1, 2, 3, \ldots, 10$ and $j = 1, 2, 3, \ldots, 50$. The lager the j here, the richer the ask price temporal information contained in the data.

5.2.2 BidLag

The $bid\{i\}lag\{j\}$ is the lag of $bid\{i\}$ and $bid\{i\}lag\{j\}$ at time t is equal to $ask_{\{i-j\}} = p_{t-j}^{b;i}$, here $i = 1, 2, 3, \ldots, 10$ and $j = 1, 2, 3, \ldots, 50$. The lager the j here, the richer the ask price temporal information contained in the data.

5.2.3 AskVolLag

$AskVol\{i\}Lag\{j\}$ is the lag of $AskVol\{i\}$ at time t is equal to $AskVol\{i-j\} = V_t^{a;i-t}$, here $i = 1, 2, 3, \ldots, 10$ and $j = 1, 2, 3, \ldots, 50$. The larger the j here, the richer the $AskVol$ price temporal information contained in the data.

5.2.4 BidVolLag

$BidVol\{i\}Lag\{j\}$ is the lag of $BidVol\{i\}$ at time t is equal to $BidVol\{i-j\} = V_t^{b;i-t}$, here $i = 1, 2, 3, \ldots, 10$ and $j = 1, 2, 3, \ldots, 50$. The larger the j here, the richer the $BidVol$ price temporal information contained in the data.

5.3 General Features

Besides the temporal features specially prepared for the tree models, we also create and analyze some general features.

5.3.1 Midprice Variance

We define the **midprice variance** with $windowsize = 5$ at time t,

$$\text{midprice variance} = \frac{\sum_{i=0}^{5} p_{t-i}}{5} \tag{5.1}$$

5.3.2 Signed Trade

We define the **signed trade** as a categorical binary variance,

$$\text{signed trade} = \begin{cases} 1, & \text{if signed is Buy} \\ 0, & \text{if signed is Sell} \end{cases} \tag{5.2}$$

5.3.3 Bid-Ask Imbalance

We define the **bid-ask imbalance** (Lipton et al. (2014)) at time t as

$$I_t^{ba} = \frac{V^{b;1} - V^{a;1}}{V^{b;1} + V^{a;1}} \tag{5.3}$$

If $I_t^{ab} > 0$, it means the order book is heavier on the bid side. If not, it is heavier on the ask side. It reflects the balance between the best prices and it is sensitive to market changes.

5.3.4 Cumulative Bid–Ask Imbalances

This section is similar to 5.3.2, What we focus now is not the best price but the top 10 prices of the order book. The cumulative bid–ask imbalances can describe the market in a more stable way. The details are represented as follow:

$$I_t^{ba;n} = \frac{\sum_{k=1}^{n} V^{b;k} - \sum_{k=1}^{n} V^{a;k}}{\sum_{k=1}^{n} V^{b;k} + \sum_{k=1}^{n} V^{a;k}} \tag{5.4}$$

5.3.5 Volume Order Imbalance

The **volume order imbalance** focuses on the volume of trade and it is sensitive to order flow changes. Shen (2015) defines the VOI as follows:

$$VOI_t = \delta V_t^{b;1} - \delta V_t^{a;1} \tag{5.5}$$

where

$$\delta V_t^{b;1} = \begin{cases} 0, & \text{if } P_t^{b;1} < P_{t-1}^{b;1} \\ V_t^{b;1} - V_{t-1}^{b;1}, & \text{if } P_t^{b;1} = P_{t-1}^{b;1} \\ V_t^{b;1}, & \text{if } P_t^{b;1} > P_{t-1}^{b;1} \end{cases} \quad \delta V_t^{a;1} = \begin{cases} V_t^{a;1}, & \text{if } P_t^{a;1} < P_{t-1}^{a;1} \\ V_t^{a;1} - V_{t-1}^{a;1}, & \text{if } P_t^{a;1} = P_{t-1}^{a;1} \\ 0, & \text{if } P_t^{b;1} > P_{t-1}^{a;1} \end{cases} \tag{5.6}$$

5.3.6 Cumulative Volume Order Imbalance

By the analogy with our generalizations for the bid–ask imbalance, we generalize the volume order imbalance (VOI) to **level VOI** as follows,

$$VOI_t^i = \delta V_t^{b;i} - \delta V_t^{a;i} \tag{5.7}$$

where

$$\delta V_t^{b;i} = \begin{cases} 0, & P_t^{b;i} < P_{t-1}^{b;i} \\ V_t^{b;i} - V_{t-1}^{b;i}, & P_t^{b;i} = P_{t-i'}^{b,i} \\ V_t^{b;i}, & P_t^{b;i} > P_{t-1}^{b;i}; \end{cases} \quad \delta V_t^{a;i} = \begin{cases} V_t^{a;i} & P_t^{a;i} < P_{t-1'}^{a;i} \\ V_t^{a;i} - V_{t-1}^{a;i}, & P_t^{a;i} = P_{t-1}^{a;i} \\ 0, & P_t^{a;i} > P_{t-1}^{a;i} \end{cases} \tag{5.8}$$

In addition, we define the **cumulative VOI** as

$$VOI_t^{1,2,\ldots,i} = \delta V_t^{b;1,2,\ldots,i} - \delta V_t^{a;1,2,\ldots,i}, \tag{5.9}$$

where

$$\delta V_t^{b;1,2,\ldots,i} = \begin{cases} 0, & \sum_{k=1}^i P_t^{b;k} < \sum_{k=1}^i P_{t-1'}^{b;k} \\ \sum_{k=1}^i V_t^{b;k} - \sum_{k=1}^i V_{t-1}^{b;k}, & \sum_{k=1}^i P_t^{b;k} = \sum_{k=1}^i P_{t-1'}^{b;k} \\ \sum_{k=1}^i V_t^{b;k}, & \sum_{k=1}^i P_t^{b;k} > \sum_{k=1}^i P_{t-1'}^{b;k} \end{cases} \tag{5.10}$$

$$\delta V_t^{a;1,2,\ldots,i} = \begin{cases} \sum_{k=1}^i V_t^{a;k}, & \sum_{k=1}^i P_t^{a;k} < \sum_{k=1}^i P_{t-1'}^{a;k} \\ \sum_{k=1}^i V_t^{a;k} - \sum_{k=1}^i V_{t-1}^{a;k}, & \sum_{k=1}^i P_t^{a;k} = \sum_{k=1}^i P_{t-1'}^{a;k} \\ 0, & \sum_{k=1}^i P_t^{a;k} > \sum_{k=1}^i P_{t-1}^{a;k} \end{cases} \tag{5.11}$$

5.3.7 Return

Frist, we define the **midprice** at time t as

$$\text{midprice}_t = \frac{p_t^{a;1} + p_t^{b;1}}{2} \tag{5.12}$$

Then **return** at time t is

$$R_t^w = \ln\left(\text{midprice}_t\right) - \ln\left(\text{midprice}_{t-w+1}\right) \tag{5.13}$$

5.3.8 Depth/Length Imbalance

We define the **depth/length imbalance** (Cao et al. (2009)) at time t and top j of the order book as

$$QR_t^j = \frac{Vol_t^{b,j} - Vol_t^{a,j}}{Vol_t^{b,j} + Vol_t^{a,j}} \tag{5.14}$$

where $j = 1, 2, \ldots, 10$.

5.3.9 Height Imbalance

We define the **height imbalance** (Cao et al. (2009)) at time t and top j of the order book as

$$HR_j = \frac{P_t^{b;j} - P_t^{b;j-1}}{P_t^{a;j} + P_t^{a;j-1}} \tag{5.15}$$

Both depth/length imbalance and height imbalance are applied to describe the shape of the order book.

5.3.10 Purchasing/Selling Press

We define the **purchasing press** at time t as

$$Press_{buy} = \frac{\sum_{i=0}^{n} V^{b;i}\text{midprice}/\left(P_t^{b;i} - \text{midprice}\right)}{\sum_{i=0}^{n} \text{midprice}/\left(P_t^{b;i} - \text{midprice}\right)} \tag{5.16}$$

and the **selling press** at time t as

$$Press_{sell} = \frac{\sum_{i=0}^{n} V^{a;i}\text{midprice}/\left(P_t^{a;i} - \text{midprice}\right)}{\sum_{i=0}^{n} \text{midprice}/\left(P_t^{a;i} - \text{midprice}\right)} \tag{5.17}$$

5.4 Difference price and Evaluation Metric

We define the **difference price** at time t as

$$\text{diference price}_{\tau} = \text{midprice}_{t+\tau} - \text{midprice}_{t}, \tag{5.18}$$

where $\tau = 1, 2, \ldots, 100$.

Taking the current moment as the starting point, we predict the difference price for the next 100 steps. In order to explore the difference in the performance of forecasting BTC price between machine learning and deep learning methods, we

use R^2 (Coefficient of determination) to evaluate the quality of the model (Chicco et al. (2021)). The calculation formula of R^2 is:

$$R^2 = 1 - \frac{(\hat{y}_i - y_i)^2}{(\bar{y}_i - y_i)^2} \quad \in [0, 1] \tag{5.19}$$

Generally speaking, when R^2 is close to 1, the prediction is accurate. Otherwise, the prediction is unvalued in applicability.

5.5 Feature Enrichment

There are some commonly utilized functions to enhance features, such as *logarithm, difference, sum, move average*, etc. and the details are represented as follow:

$$\begin{aligned} ln(feature) &= log_e(\text{feature}) \\ diff_t(n, \text{feature}) &= \text{feature}_{t+n} - \text{feature}_t \\ sum_t(n, \text{feature}) &= \text{feature}_{t+n} + \text{feature}_t \\ MA_t(n, \text{feature}) &= \frac{\sum_{t=i-n+1}^{t=i} \text{feature}_i}{n}, (i > n-1) \end{aligned} \tag{5.20}$$

Compared with machine learning algorithms, although deep learning methods are able to automatically build cross-features, we still attempt to build some cross features, such as **Multiplicative Cross Feature, Additive Cross Feature, Subtraction Cross Feature**, etc. It is noteworthy that the **Division Cross Feature** is rarely used as 0 may appear on the denominator.

$$\begin{aligned} mult_{i,j} &= \text{feature}_i * \text{feature}_j \\ add_{i,j} &= \text{feature}_i + \text{feature}_j \\ sub_{i,j} &= \text{feature}_i - \text{feature}_j \end{aligned} \tag{5.21}$$

In addition, there are also some other applicable automatic feature tools, such as TPOT (Olson and Moore (2016), Auto-Sklearn (Feurer et al. (2015)), H2O (LeDell and Poirier (2020)), etc. It has been proven that cross features can improve the prediction accuracy of the machine learning algorithm. However, the computational cost is very high, and it needs to be used in conjunction with feature selection (Karmaker et al. (2021)). Taking the temporal features into consideration, the Chapter 5.2, is designed separately for the machine learning algorithm. So, we leave no particular part for the cross feature.

Chapter 6

Experimental Settings

In our experiments, we attempt to explore the performance of machine learning and deep learning methods in high-frequency quantitative trading. We collect data from the BitMEX exchange and focus on the order book data. Judging from the overall performance of BTC, the BTC price this year fluctuates sharply, from the highest of $47,686.81 to the lowest of $19,017.64. The dramatic price changes add difficulties to our regression predictions. Therefore, we firstly choose XGBoost and LightGBM as benchmarks among machine learning algorithms. In the Kaggle competition, they performed very well and are also heated research topic in previous studies.

There are many choices for deep learning algorithms, which include Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Temporal Convolutional Network (TCN), Transformer and so on. The LSTM is the most widely used time series prediction algorithm. Recently, some studies have used TCN and Transformer as advanced approaches for time series prediction. However, unlike the Transformer which is made under the framework of the attention mechanism, the TCN is built based on CNN and make full use of historical time series information. We believe that this innovative algorithm can improve the performance of deep learning algorithms on time-series tasks. In the end we choose LSTM and TCN as benchmarks to be compared with machine learning algorithms.

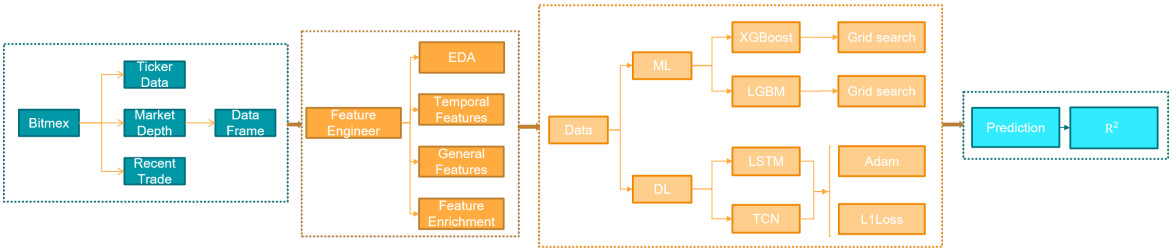

Figure 6.1: Forecasting process

6.1 Experiments with Boosting Methods

As explained in the former section, the idea behind boosting is to augment weak classifiers accuracy using ensemble averaging. Like Bagging, boosting is based on building an ensemble of models aggregated by averaging the estimations or voting. However, it differs in the way of construction. This ensemble proposes to build a sequence of models such that at each iteration, each model added to the combination improves the overall solution.

Neither XGBoost nor LightGBM can perform multivariate regression. Although MultiOutputRegressor in Scikit-Learn can be used to encapsulate the model to implement multivariate regression, but not true multivariate regression. Our task is to predict the 100-step difference price, so there are two options. One is to build a model, using this model to fit the features corresponding to 100 steps and get 100 regression results. The other is to build 100 models, each of which specifically predicts one of the 100 steps. After the consideration, we finally chose the first option. We will build one model on the first step features and forecast the last 99 steps.

6.1.1 Extreme Gradient Boosting (XGBoost)

The Extreme Gradient Boosting is an effective and optimized type of gradient tree boosting algorithm and it has recently gained immense popularity especially due to its exceptional performance in Kaggle competitions. By providing bagging-bootstrap aggregation and feature randomness, the XGBoost both prevents overfitting issues and considers bias-variance trade-off. Owing to such distinctive advantages, the XGBoost has lately become one of the most popular Machine Learning algorithms in cryptocurrency forecasting studies.

In our experiment, we use the grid search algorithm provided by Scikit-Learn package to find the best hyper-parameters for XGBoost. The parameter space we use is shown in Table 6.1.

Hyperparameter	Grid for XGBoost	Best
n_estimators	$[200, 250, 300, 350, \ldots, 500]$	250
booster	[gblinear, gbtree, dart]	gblinear
learning_rate	$[0.05, 0.06, 0.07, \ldots, 0.16]$	0.06
max_depth	$[4, 5, 6, 7, 8]$	5
min_child_weight	$[1, 2, 3, \ldots, 7]$	1
reg_lambda	$[0.05, 0.1, 1, 2, 3]$	0.1
reg_alpha	$[0.05, 0.1, 1, 2, 3]$	0.1

Table 6.1: Grid search hyperparameters space for XGBoost

6.1.2 Light Gradient Boosting Machine (LightGBM)

Light Gradient Boosting Machine (LightGBM) is a new gradient boosting library implemented by Microsoft in April 2017. The aim was to make gradient boosting on decision trees faster. The idea is that instead of checking all of the splits when creating new leaves, we only check some of them: We first sort all of the attributes and bucket the observation by creating discrete bins. When we want to split a leaf in the tree, instead of iterating over all of the leaves, we simply iterate over all of the buckets. This algorithm chooses the leaf with maximum delta loss to grow and does not grow the whole level. In our study, the best hyper-parameters setting of LightGBM are shown in the following table:

Hyperparameter	Grid for LightGBM	Best
n_estimators	$[50, 100, 150, 200, \ldots, 500]$	100
boosting_type	[rf, gbtree, goss, dart]	gblinear
learning_rate	$[0.05, 0.06, 0.07, \ldots, 0.16]$	0.08
max_depth	$[4, 5, 6, 7, 8]$	6
num_leaves	$[6, 8, 10, \ldots, 18]$	16
objective	[regression, regression_L1]	regression

Table 6.2: Grid search hyperparameters space for LightGBM

6.2 Experiments with Deep Learning Methods

Over the last years, deep learning methodologies were widely applied on time-series predictions, focusing on popular real-world application domains such as the cryptocurrency market. Most of these models exploit advanced deep learning techniques and special architectural designs based on convolutional and long short-term memory (LSTM) layers. Convolutional layers are utilized to filter out the noise in complex time-series data as well as extracting new valuable features while LSTM layers are used to efficiently capture sequence patterns as well as long and short-term dependencies.

The deep learning method itself can support multivariate regression, so we only need to set the output of the model as 100 to achieve the 100-step prediction. However, the deep learning algorithm is very sensitive to the input and it needs to be normalized, while XGBoost and LightGBM do not need that. In this section, we will first describe the experimental settings of LSTM, in order to highlight the specificities of the temporal convolutional network further.

6.2.1 Long Short-Term Memory (LSTM)

As a popular deep learning method in time series forecasting, the feasibility of LSTM network in the BTC price forecasting has recently been widely verified. It has an

architecture similar to the Recurrent Neural Network (RNN), which uses a loop to pass information from one step of the network to the next. The LSTM is known for the capability to learn from sequence dependency while also solving the traditional memory problem RNN. For the RNN, if the necessary information is further back of the sequence, the gradients, which update neural network weights, can be minimal and lead to a short-term memory problem. In this case, the RNN struggles to fetch the relevant information, and then the LSTM was introduced to tackle this problem.

Unlike the gradient boosting methods, the deep neural networks are sensitive to the input scale, so it needs to be normalized. After experiments, we finally chose the *standard scaler* in Scikit-learn. Normalization can also accelerate the convergence of the network, reduce the training time, and improve the final result significantly. Another important difference is that both LSTM and TCN do not need temporary features, and only need to input multiple pieces of historical information into the network at the same time. For the output of the LSTM network, we adopt the f_1 function to change its shape. In PyTorch, as follows:

```
out, (h,c) = self.lstm(x)
a, b, c = out.shape
out = out.reshape(a, 1, b*c)
```

Specifically, the shape of **previous input** is $(batch, seq, feature)$. The seq is the length of historical information for a single time series. We use the last 50 trade information to predict the difference price and set $seq = 50$. We attempt to add a ReLU activation layer and a linear layer between the linear layer and the output layer. However, we found that the effects of the two linear layers were not strong, so we only retained one linear layer in the end. Due to the small number of network layers, we added the Batch Normalization layer to avoid overfitting. The final configuration of network parameters and the best model structure are shown in Table 6.3 and Figure 6.2.

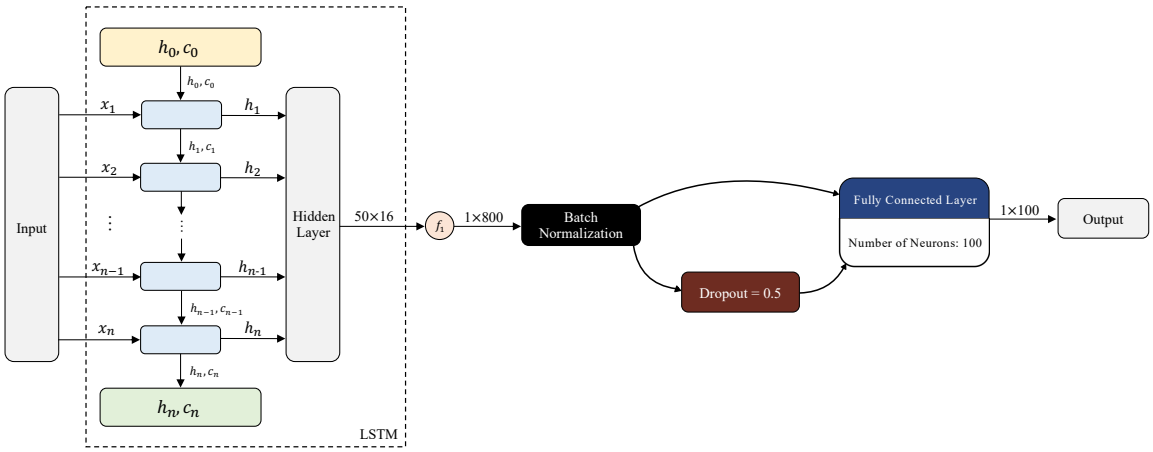

Figure 6.2: LSTM network structure

learning rate	epochs	dropout	hidden_size
0.01, 0.001, 0.0001	500	0.75	16
loss_fn	batch_size	optimizer	norm_type
L1 Loss	256	Adam	bn

Table 6.3: Configuration of LSTM network parameters

6.2.2 Temporal Convolutional Network (TCN)

The fixed-size input vector and inconsistent input and output sizes restrict the application of Convolutional Neural Network (CNN) in time-series prediction. The temporal convolutional network (TCN), as a variant of the CNN, employs casual convolutions and dilations. Therefore, it is adaptive for sequential data with its temporality and large receptive fields and exhibits longer memory than recurrent architectures with the same capacity. Furthermore, the TCN has a simple network structure and outperforms canonical recurrent networks, such as the RNN and LSTM, in terms of the accuracy and efficiency of time-series data analysis. This conclusion was verified again in the task of Bitcoin price prediction. We will discuss the details later in the experimental results.

The parameters setting and the model structure of TCN are shown below. Overall, the structure of TCN is roughly the same as LSTM, but there are two main differences in **previous input** and historical information processing. Firstly, the input is exactly the same, but the dimensional order of seq and feature is changed by the f_2 function. The second difference is that the TCN introduces the CNN convolution operation, which is a completely different model from the RNNs.

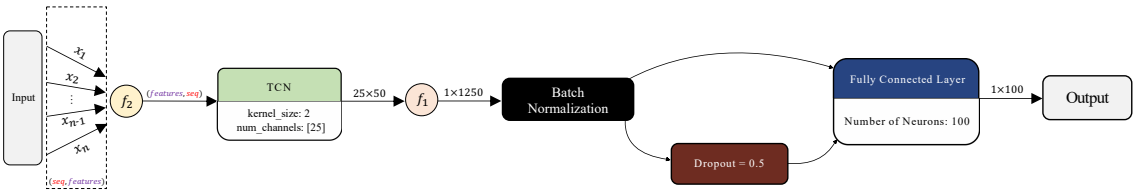

Figure 6.3: TCN network structure

learning rate	epochs	dropout	hidden_size	num_channels
0.01, 0.001, 0.0001	500	0.5	16	[25]
kernel_size	loss_fn	batch_size	optimizer	norm_type
2	L1 Loss	256	Adam	bn

Table 6.4: Configuration of LSTM network parameters

For the realization of TCN, we refer to '*An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*' (Bai et al. (2018)) and the corresponding code to build the network. There is one point to note that f_2 is the function that changes the dimension order of seq and $feature$. In PyTorch, the implementation detail is as follows: `x = x.permute(0,2,1)`

Chapter 7

Results and Discussion

7.1 Experimental Results of Boosting Methods

To solve the problem that machine learning methods cannot use time-series features, we created the feature augmentation separately to help machine learning methods. Both LightGBM and XGBoost are gradient boosting methods with the similar R^2 scores. The multivariable regression strategies of the two models are also identical, that is, the model obtained by the first step training is used to predict the next 99 steps. Therefore, the score of R^2 is continuously decreasing, which is completely different from the linear model or deep learning methods that support multivariable regression. Although we know that the best method is to train 100 models for 100 steps, the cost is completely unacceptable. As shown in Figure 7.1, the maximum R^2 score in XGBoost (13.9%) was slightly lower than that of LightGBM (16.1%), however, the XGBoost model has less volatility and the curve is smoother. This indicates that the performance of XGBoost in the long-term forecast is better than that of LightGBM.

7.2 Experimental Results of Deep Learning Methods

TCN and LSTM use different principles to process timing information, so they are discussed separately. Both TCN and LSTM support multivariate regression, so we only use one model to complete the 100-step prediction, which is different from that of XGBoost and LightGBM. The final results obtained by the deep learning methods show a trend of rising followed by a decrease. Overall, as shown in Figure 7.2, the performance of TCN (28.9%) is better than LSTM (26.7%). The TCN was built later than LSTM, and it makes use of deep residual networks (ResNet) and Dilated Convolution, which are more advanced in the composition module. However, it is noteworthy that the LSTM we employed here is a basic version, and the drawback of the study is that we made no comparison between TCN and other improved LSTMs, such as Bi-LSTM, CNN-LSTM, etc.

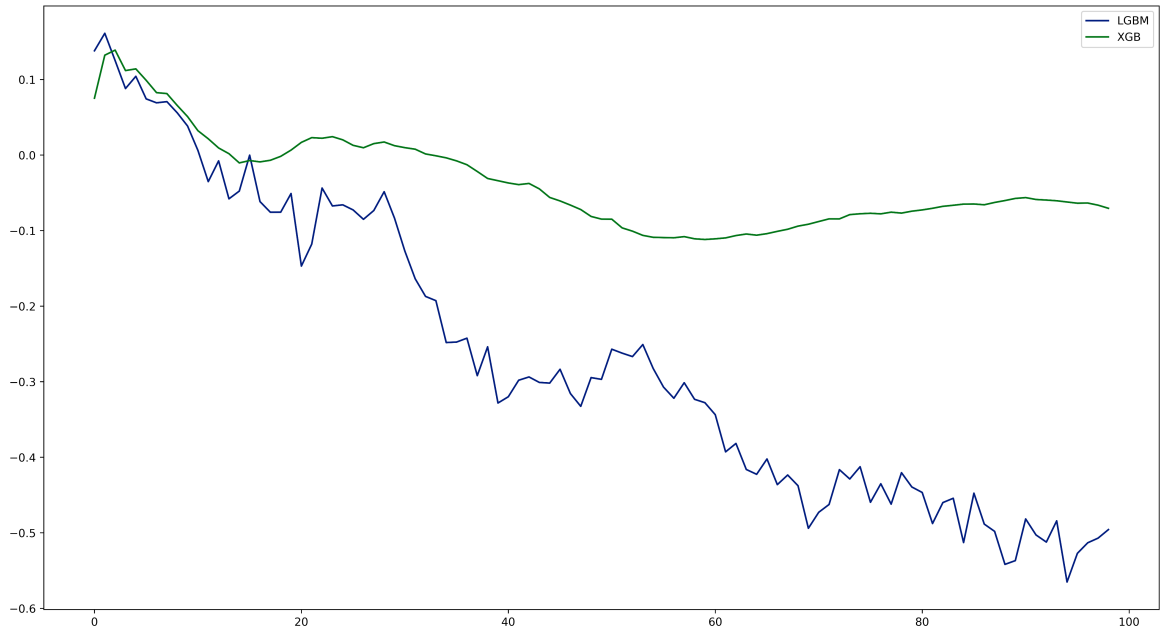

Figure 7.1: Comparison of results for boosting methods

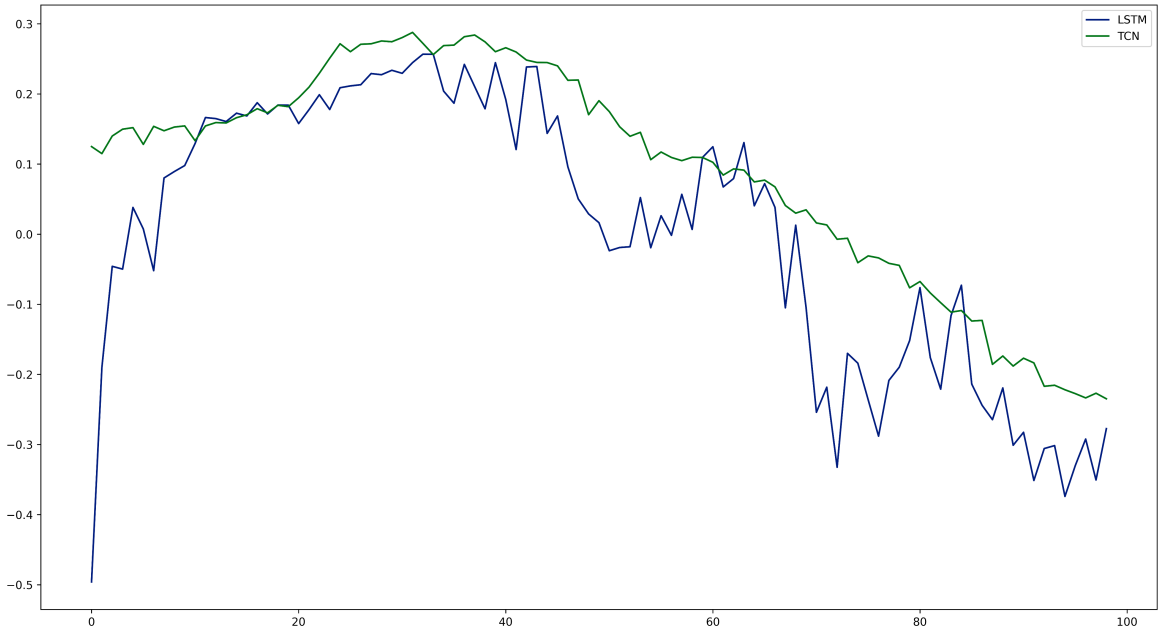

Figure 7.2: Comparison of results for deep learning methods

7.3 Discussion

After observing the results of R^2 in gradient boosting and deep learning methods, overall, we found that deep learning methods perform better than XGBoost and LightGBM. The best model of the machine learning method is the XGBoost since the fact that its performance on the prediction of the BTC price is more stable. Although

the overall performance of the gradient boosting methods is not as good as that of the deep learning methods, it is superior in its simplicity of training and strong interpretability. However, for the prediction accuracy and overall performance, there is no doubt that the best model is the TCN, which is better than LSTM and LightGBM in all aspects. However, if the parameters are not set properly, it is easy to cause the gradient vanishing problem during the training processes for the TCN.

Furthermore, it is important to note that, for some common prediction tasks, R^2 is effective when it scores more than 0.5. However, this criterion does not seem to be reasonable for the prediction of BTC price. This is attributed to the fact that there are too many over-the-counter factors in BTC price fluctuations, and it is difficult to predict by analyzing the order book data only; more market and policy data are required to improve the forecast performance.

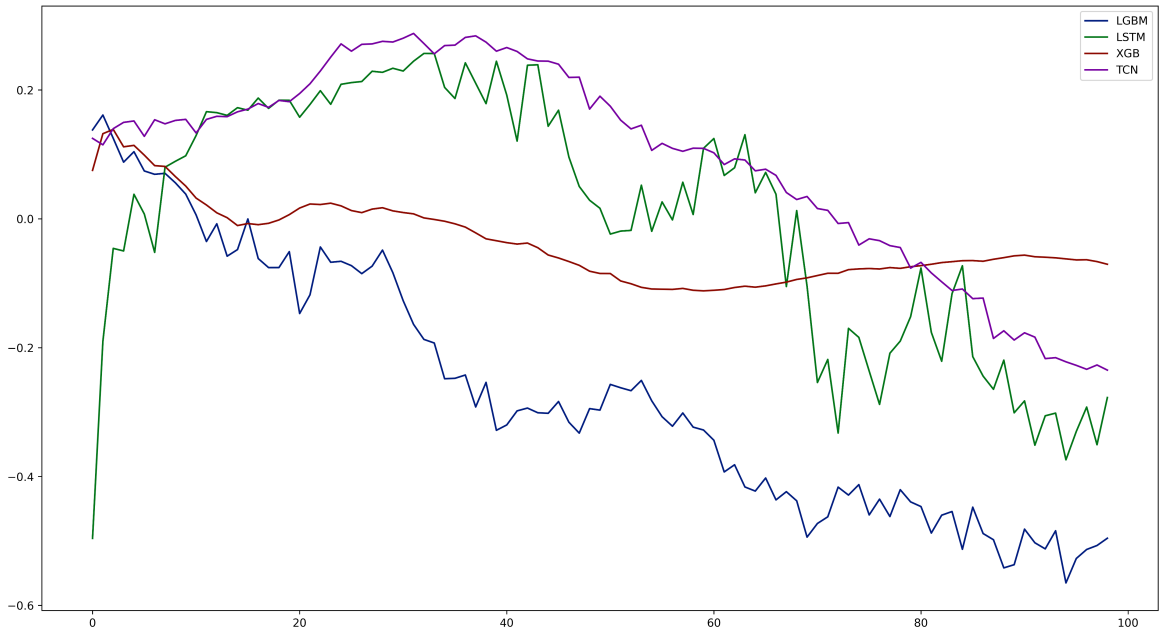

Figure 7.3: Overall model comparison

Prediction Model	Maximum R^2 Score	Minimum R^2 Score	Difference
XGBoost	13.9%	**-11.2%**	25.1%
LightGBM	16.1%	-56.5%	72.6%
LSTM	26.7%	-49.6%	76.3%
TCN	**28.9%**	-23.5%	52.4%

Table 7.1: R^2 Score comparison on four different models

Chapter 8

Conclusion

8.1 Summary

Since the BTC and the Blockchain technology were introduced in 2008, it has taken a predominant place in the cryptocurrency industry. The prediction of the cryptocurrencies price has been a central topic, from which we can take advantage of arbitrage for an investment. In this study, four models, i.e. Extreme Gradient Boosting, Light Gradient Boosting Machine, Long Short-term Memory Model, and Temporal Convolutional Network are implemented to forecast the future 100-step difference price.

From the experiment, it is concluded that the Temporal Convolutional Network gives the excellent performance in price forecasting; it has the out-of-sample R^2 value of around 0.28 between steps 30 and 40. In addition, the LSTM model has a comparable performance with the best one (0.26). The two above-mentioned deep learning methods perform better than gradient boosting methods on the prediction accuracy. Specifically, the R^2 for the Extreme Gradient Boosting and Light Gradient Boosting Machine are only approximately between 0.13 and 0.16 from steps 3 to 5. Although the former might lead to the worst result, its less volatility indicates a better performance in the long-term forecasting.

8.2 Future Work

One major challenge we are currently faced with is that the deep learning methods is not effective enough, particularly the TCN, which costs a day to obtain results. Therefore, we only select the data of three days as a training set and assume it provides sufficient information. However, the BTC conveys extraordinary complexity in the reality and correlates with other markets. Future studies are expected to search for an alternative algorithm that is more efficient. Additionally, the current dataset size should be increased, and additional information deserves more attention (i.e. the market index) in the improvement of performance.

Bibliography

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. pages 4, 17, 18, 36

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140. pages 11

Cao, C., Hansch, O., and Wang, X. (2009). The information content of an open limit-order book. *Journal of Futures Markets*, 29(1):16–41. pages 29

Chen, T. and Guestrin, C. (2016). XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. pages 11

Chicco, D., Warrens, M. J., and Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, 7:e623. pages 30

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28. pages 30

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232. pages 11

Garcia, D., Tessone, C. J., Mavrodiev, P., and Perony, N. (2014). The digital traces of bubbles: feedback cycles between socio-economic signals in the bitcoin economy. *Journal of the Royal Society Interface*, 11(99):20140623. pages 7

Guo, H., Zhang, D., Liu, S., Wang, L., and Ding, Y. (2021). Bitcoin price forecasting: A perspective of underlying blockchain transactions. *Decision Support Systems*, 151:113650. pages 10

Guo, T., Bifet, A., and Antulov-Fantulin, N. (2018). Bitcoin volatility forecasting with a glimpse into buy and sell orders. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. pages 9

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. pages 18

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. pages 15

Huang, J.-Z., Huang, W., and Ni, J. (2019). Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3):140–155. pages 9

Ji, S., Kim, J., and Im, H. (2019). A comparative study of bitcoin price prediction using deep learning. *Mathematics*, 7(10):898. pages 10

Karmaker, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. (2021). Automl to date and beyond: Challenges and opportunities. *ACM Comput. Surv.*, 54(8). pages 30

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30. pages 13

Khedmati, M., Seifi, F., and Azizi, M. (2020). Time series forecasting of bitcoin price based on autoregressive integrated moving average and machine learning approaches. *International Journal of Engineering*, 33(7):1293–1303. pages 9

Lea, C., Vidal, R., Reiter, A., and Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision*, pages 47–54. Springer. pages 17

LeDell, E. and Poirier, S. (2020). H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020. pages 30

Lindemann, B., Müller, T., Vietz, H., Jazdi, N., and Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020. pages 15

Lipton, A., Pesavento, U., and Sotiropoulos, M. (2014). Trading strategies via book imbalance. *Risk*, page 70. pages 27

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. pages 17

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260. pages 7

Olson, R. S. and Moore, J. H. (2016). Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pages 66–74. PMLR. pages 30

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. pages 17

Pang, Y., Sundararaj, G., and Ren, J. (2019). Cryptocurrency price prediction using time series and social sentiment data. BDCAT '19, page 35–41, New York, NY, USA. Association for Computing Machinery. pages 10

Rathore, R. K., Mishra, D., Mehra, P. S., Pal, O., HASHIM, A. S., Shapi'i, A., Ciano, T., and Shutaywi, M. (2022). Real-world model for bitcoin price prediction. *Information Processing & Management*, 59(4):102968. pages 8

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. pages 18

Shen, D. (2015). *Order imbalance based strategy in high frequency trading*. PhD thesis, oxford university. pages 27

Shen, Z., Wan, Q., and Leatham, D. J. (2021). Bitcoin return volatility forecasting: A comparative study between garch and rnn. *Journal of Risk and Financial Management*, 14(7):337. pages 9

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. pages 18

Wan, R., Mei, S., Wang, J., Liu, M., and Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, 8(8):876. pages 17