UNIVERSITY OF CALIFORNIA
Los Angeles

Stock Trend Prediction:
Based on Machine Learning Methods

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Yuan Song

2018

ABSTRACT OF THE THESIS

Stock Trend Prediction:
Based on Machine Learning Methods

by

Yuan Song
Master of Applied Statistics
University of California, Los Angeles, 2018
Professor Yingnian Wu, Chair

Nowadays, people show more and more enthusiasm for applying machine learning methods to finance domain. Many scholars and investors are trying to discover the mystery behind the stock market by applying deep learning. This thesis compares four machine learning methods: long short-term memory (LSTM), gated recurrent units (GRU), support vector machine (SVM), and eXtreme gradient boosting (XGBoost) to test which one performs the best in predicting the stock trend. I chose stock price indicators from 20 well-known public companies and calculated their related technical indicators as inputs, which are Relative Strength Index, the Average Directional Movement Index, and the Parabolic Stop and Reverse. Experimental results show that recurrent neural network outperforms in time-series related prediction. Especially for gated recurrent units, its accuracy rate is around 5% higher than support vector machine and eXtreme gradient boosting.

The thesis of Yuan Song is approved.

Nicolas Christou

Hongquan Xu

Yingnian Wu, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

The stock market is one of the most significant approaches for public companies to raise funds, and it is also the main component of the capital market. The stock market has been known as 'the barometer of the economy', which means its fluctuation is closely related to the prosperity or recession of the economic market.

Stock as for the public companies is the way of financing. Through issuing shares and bonds in the stock market, public companies can raise idle funds to solve the problem of temporary capital scarcity. For private investors, the stock is one of the easily accessible financial products. Through holding suitable stocks and doing appropriate portfolio management, personal assets will be appreciated to some degree. And also, stock trading is the main business for securities companies and investment banks. With the development of computer science technology, more and more companies are applying the technology of big data and Artificial Intelligence to financial investment area. For example, according to the news report, in 2000, Goldman Sachs used to have more than 600 traders of U.S. cash equities trading desk in New York; while in 2017, there were only two human equity traders left, with the computer doing the rest of work [1].

Stock trend and price prediction is the act of making judgment on the future value of a company stock, which has long existed since the establishment of stock market. Efficient prediction of future stock price can yield considerable profits. However, the stock market is affected by many macro and micro factors, such as interrelated political and economic indicators, supply-demand relationship of stock, operation conditions of the company and so on.

[1] http://www.newsweek.com/2017/03/10/how-artificial-intelligence-transform-wall-street-560637.html

In 1965, Fama [1] raised the Efficient Market Hypothesis (EMH). It posits that the stock market is an efficient market, which the price of stocks is the reflection of all publicly known or newly revealed information and rational expectations. This theory established solid foundation and basis for follow-up stock prediction study. Based on the size of market information and the reaction speed of market on information, Fama categorized the efficient market hypothesis into three levels, which are weak form, semi-strong form, and strong form. The U.S. market is defined as a semi-strong form market that the market is efficient and stocks can adjust instantly to absorb new information. Burton Malkie [2], the leading proponent of EMH, stated that the stock prices therefore are unpredictable and random.

While the Efficient Market Hypothesis is highly accepted among the financial academia, the real-world market experience gives different opinions and challenges the hypothesis of unpredictability. Nowadays, analysts are trying to unveil the mysteries of the stock market, and some large companies devote a tremendous amount of money on hiring experts to build statistical models for prediction. There are many reasons to explain this phenomenon. First, with the rise of big data technology and cloud computing, people can handle and analyze mass stock exchange datasets with great ease. In many fund companies and securities investment companies, quantitative analysis has moved from manual work to artificial intelligence. Second, deep learning achieved great progress on natural language processing. Thus, the ability of computer to collect and analyze public information of stock market is improved greatly. Third, the more machine learning techniques the computer absorbs, the 'smarter' it will be. Artificial intelligence could be the new solution to tackle the stock prediction problem down. In May 2017, the latest version of Google AlphaGo[2] beat the world's best human Go player, Jie Ke, through the perfect deep learning method. We can speculate that computer also has the power of analyzing and predicting stock market over human securities analysts one day.

The important position of stock in the economic market makes it meaningful to predict its trend and future price. In the following chapters, under the assumption that stock

[2]https://deepmind.com/research/alphago/alphago-china/

prices can be predicted, I will develop how different machine learning methods will perform on stock trend prediction. The rest of this paper is organized into 7 chapters. Chapter 2 presents previous academic work on forecasting the stock market and its trend. This chapter mainly focuses on works by using traditional statistical methods, neural networks, and classification. Chapter 3 gives the brief definition of the U.S. stock market. Chapter 4 presents the mathematical concepts, which are at the core of the recurrent neural network, support vector machine, and eXtreme gradient boosting. Chapter 5 is the introduction of the dataset, dataset pre-processing and methodology for models. Chapter 6 summarizes the observed results and Chapter 7 makes the conclusion and discusses future work.

CHAPTER 2

Literature Review

2.1 Previous Stock Prediction

Financial time series forecasting, such as stock market prediction, is usually considered a challenging task due to its volatile and chaotic characteristics [3]. Precisely predicting stock trend and price movement is still an open question nowadays. During the past decades, various methodologies have been raised to perfect the prediction. In 1952, American economist Markowitz [4], in his paper *Portfolio Selection*, first raised the concept of portfolio theory. In this theory, he introduced two quantitative investment indexes, the mean and variance of portfolio asset. Markowitz formalized investors preferences mathematically, used formulas to explain the investment diversification, and discussed portfolio selection and management systematically. This achievement represents the beginning of using mathematical ways to understand and master the trend of economic market. Besides predicting the overall trend of stock market, some scholars also dig into the price fluctuation of each stock by statistical methods. For example, Fung et al. [5] used hypothesis testing method together with piecewise segmentation algorithm to predict the trends on the time series. The rise or drop trend is categorized by the slope of each piecewise segment and the coefficient of determination.

2.2 Neural Network

Given the sophistication of financial time series data, introducing deep learning into financial market prediction becomes such a popular topic [6]. In technological methods area, one of the most promising technique is Artificial Neural Networks (ANN). ANN is the non-

linear computational models, which is capable of doing prediction based on market data, without previous learning of the relationships between input and output variables [7]. For stock prediction, the most popular type of ANN is using the feed-forward neural network and back-propagation for weight training. Another form of ANN suitable for stock prediction is the recurrent neural network (RNN).

Many scholars have devoted using the neural network to predict the financial market. Wilson and Sharda [8] compared the predictive capability of neural networks and classical multivariate discriminant analysis for firm bankruptcy. The result turned out that neural networks performed significantly better than the other models at predicting bankruptcy. Hsieh et al. [9] applied an integrated model of wavelet transforms and recurrent neural network to forecast stock markets based on four stock indexes: the Dow Jones Industrial Average, FTSE 100 Index, Nikkei 225 Index, and Taiwan Stock Exchange Capitalization Weighted Stock Index. They proved that the model works great on four indexes and could be used in real life to earn profits. Tanigawa and Kamijo [10] used recurrent neural networks to recognize the stock pattern and 15 out of 16 of the recognition experiments were accurate.

Unlike conventional recurrent neural network, long short-term memory (LSTM) [11] and Gated Recurrent Unit (GRU) [12] are more suitable to learn from past experience to classify, process, and predict time series with arbitrary size of time steps. These two models are designed specifically to deal with the problem of vanishing gradients. Akita et al. [13] confirmed that LSTM is able to capture the time series influence on stock price than other models. Persio and Honchar [14] used LSTM model on stock price prediction and concluded that with the algorithm, the reliability of prediction has been significantly increased. Similarly, Chen et al. [15] conducted stock market prediction using GRU, which also achieved a good prediction performance with small error.

Thus, in this experiment, I will use LSTM and GRU to predict the stock trends.

2.3 Classification

Support Vector Machine (SVM) has been widely used to predict the direction of financial time series and gained high accuracy. Prasaddas and Padhy [16] showed that when predict futures prices in Indian stock market, the performances of support vector machine are better than the results of back-propagation technique. Tay and Cao [17] compared the support vector machine and multi-layer back propagation neural network for financial time series forecasting. The experiment proved that SVMs showed superior performance over back propagation in forecasting. Jain et al. [18] researched on the performances of the Extreme Gradient Boosting (XGBoost) on sales forecasting, and found that XGBoost algorithm outperforms traditional regression algorithms with higher accuracy. Sawon and Hosen [19] performed a research of prediction on large scale of data to compare the performance of XGBoost and other traditional models like linear regression and random forest regression. The result confirmed that XGBoost outperforms the traditional ones with higher accuracy.

So I will also conduct support vector machine and extreme gradient boosting in the experiment to compare with the performance of recurrent neural network.

CHAPTER 3

Financial Definition

3.1 Stock Market

Stock prediction is using historical price, related market information and so on to forecast exact price or price trend of the stock in the near future. According to the time granularity of price information, stock trading can be divided into low-latency trading based on daily basis and high-frequency trading, which market exchanges in a matter of hours, minutes, even seconds. High-frequency trading analysis is more common in hedge funds, investment banks, and large institutional investors. It masters the trading signals before prices' ups and downs through analyzing great amount of trading data [20]. In this thesis, only low-latency trading is taking into consideration, which is more common in academia. Its core concept is to increase the accuracy of stock prediction based on the related market information.

The U.S. stock market mainly studies stocks traded at National Association of Securities Dealers Automated Quotations (NASDAQ) and New York Stock Exchange (NYSE). Stocks in the U.S. market are issued by world well-known companies, such as Apple Inc., Google, Microsoft, etc.

The trading rule of NASDAQ and NYSE is T+0, which is also called same-day affirmation. It means that completing the entire trading is processed on the same day. Investors can buy or sell stocks at any time on trading days. Also, U.S. stock market does not set limit up and limit down, rather it adopts circuit breakers/ trading curb to reduce market volatility and prevent stock prices from increasing or decreasing suddenly.

3.2 Stock Trend Definition

In this thesis, I will mainly focus on predicting ups and downs of stock. Stocks leave some important trading data after each trading day, such as open price, close price, adjusted close price, highest price, lowest price, volume, etc. Among all, adjusted close price usually represents the stock price of that trading day. In the trading period, there will be a series of adjusted close prices. Let us denote it as:

$$\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3,\ldots,\ \mathbf{p}_T.$$

Here, $\mathbf{p}_t$ is the close price on $\mathbf{t}$ trading day, $\mathbf{T}$ is total trading days in this period.

Stock price of a certain trading day will rise or drop comparing to previous trading day, thus, here I used the change of closing price of two consecutive trading days as the judgment. Let us denote trading situation as:

$$y_t = \begin{cases} 1 & \text{if } p_t > p_{t-1}, \\ 0 & \text{if } p_t \leq p_{t-1}. \end{cases} \tag{3.1}$$

If it is 1, it means the price goes up on the second trading day. Otherwise, if it is 0, it means the price goes down or remains the same.

CHAPTER 4

Math Theory and Definition

4.1 Recurrent Neural Network

Recurrent Neural Network is a category of neural networks, which is similar to the concept of feedback loop. After the data have been processed by the hidden layers, the original data will be put back to the hidden layers together with the new input. RNNs are very powerful and suitable for time serial prediction. They have the following two properties: (1) the distributed hidden states make them to improve the efficiency of storing past information; (2) the non-linear dynamics can help to update the hidden state in a more sophisticated way [21]. There are several types of recurrent neural networks with different structures. In the following, I will briefly introduce two similar models, long short-term memory and gated recurrent units.

4.1.1 Long Short-Term Memory

long short-term memory is one of the variations of recurrent neural networks. Though RNNs do great on time series data, it is hard to avoid the problem of long-term dependencies due to the vanishing gradient [22]. Then LSTM [11] was introduced as an effective way to solve vanishing gradient by having memory cells to retain the time related information. A memory cell includes in the input gate $\mathbf{i}_t$, output gate $\mathbf{o}_t$, and forget gate $\mathbf{f}_t$. The gates are used to control the interaction between itself and neighbor memory cells.

At time t, the input is $\mathbf{x}_t$, output is $\mathbf{h}_t$, and historic information that memory cell memorizes is $\mathbf{C}_t$. Input gate controls the portion of information at time t; forget gate chooses to remember or forget historic information at time t; and output gate decides how

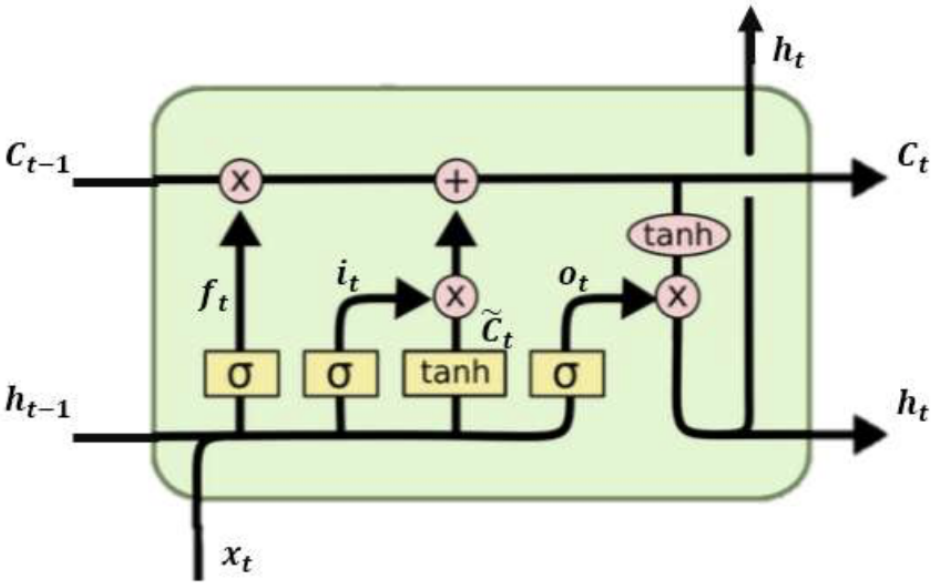

Figure 4.1: Memory Cell of LSTM [23]

much information will be delivered to the next layer. The structure of memory cell block is shown in Figure 4.1. Since the memory cell has the sequential feature, $\mathbf{C}_t$ contains all input information from time 0 to time t, which represents the status at time t.

The forget gate $\mathbf{f}_t$ is determined by input $\mathbf{x}_t$ and output $\mathbf{f}_{t-1}$ of last time, and then wrapped in the sigmoid function. See (4.1) for the formula. The value of forget gate bounces between 0 and 1. When $\mathbf{f}_t$ is 0, that means the previous value is forgotten in the calculation. Otherwise, if it is 1, the forget gate keeps previous information.

With input $\mathbf{x}_t$ and previous output $\mathbf{h}_{t-1}$, through transformation of *tanh* function, we can get $\Delta\mathbf{C}_t$ where its value is between -1 to 1; see (4.2). This means how much information will be added to $\mathbf{C}_t$ after introducing input $\mathbf{x}_t$.

Similar to the forget gate, the input gate $\mathbf{i}_t$ regulates the information of input $\mathbf{x}_t$ and previous output $\mathbf{h}_{t-1}$; see (4.3). If $\mathbf{i}_t$ is 0, then $\Delta\mathbf{C}_t$ can be neglected; otherwise, if $\mathbf{i}_t$ is 1, then $\Delta\mathbf{C}_t$ will be counted into $\mathbf{C}_t$.

The state vector $\mathbf{C}_t$ is consisted of two parts. The first part is the state vector of previous time node $\mathbf{C}_{t-1}$, which is controlled by forget gate. And the second part is $\Delta\mathbf{C}_t$, which is

determined by how much input gate takes; see (4.4).

The output gate $\mathbf{o}_t$ is similar to input gate and forget gate; see (4.5). The size of output $\mathbf{o}_t$ decides how much likely the state vector $\mathbf{C}_t$ will be outputted and received by other layers.

The output vector $\mathbf{h}_t$ is the combination of the output gate $\mathbf{o}_t$ and the cell state $\mathbf{C}_t$; see (4.6). $\mathbf{C}_t$ needs to be transformed by *tanh* function, and then let output gate $\mathbf{o}_t$ decide the portion of output.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{4.1}$$

$$\Delta C_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4.2}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.3}$$

$$C_t = f_t * C_{t-1} + i_t * \Delta C_t \tag{4.4}$$

$$o_t = \Sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.5}$$

$$h_t = o_t * tanh(C_t) \tag{4.6}$$

4.1.2 Gated Recurrent Unit

Gated Recurrent Unit is very similar to long short-term memory. The main difference is how they apply gates. For GRU, there are only two gates instead of three, which are the reset gate and update gate . Figure 4.2 illustrated the procedure of GRU memory cell. The reset gate decides how to combine the new input and previous memory, and controls the extent of neglecting previous memory. The smaller value of reset gate is, the more previous memory will be neglected. The update gate determines how much of previous memory will be kept. The larger value of update gate is, the more previous information will be brought. When we set all reset gates into 1 and update gates into 0, then we back to the plain recurrent neural network model.

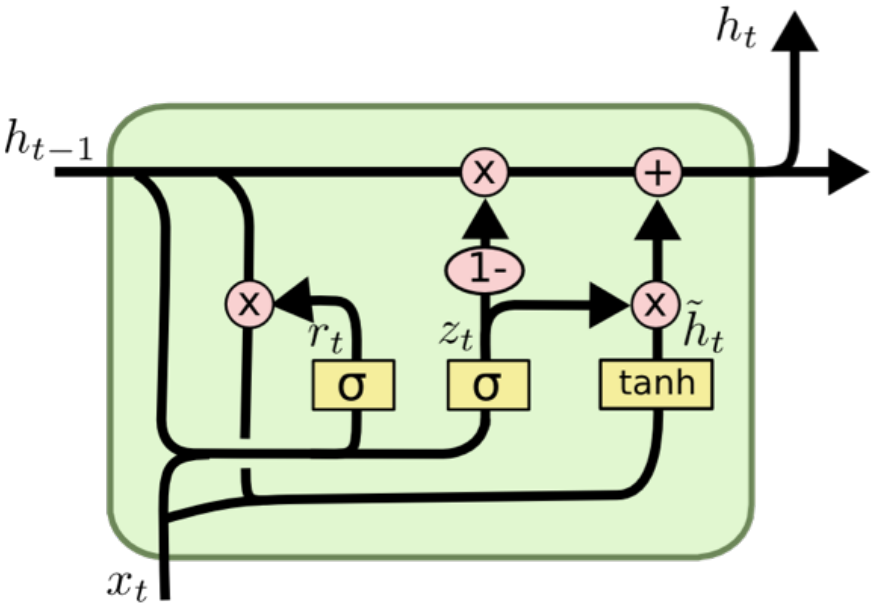

Figure 4.2: Memory Cell of GRU [23]

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{4.7}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{4.8}$$

$$\widetilde{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{4.9}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t} \tag{4.10}$$

There are few more differences between LSTM and GRU. First, LSTM can control on the exposure of memory, while GRU exposes the memory, which is not controllable. Then, GRU does not have the output gate like in LSTM. Also, the input gate and forget gate in LSTM are substituted by the update gate, and thus the reset gate is directly applied to the previous hidden state. Since GRU has fewer parameters than LSTM, we could make a preliminary guess that GRU will be faster than LSTM and need less data. However, if the data scale is large, LSTM may lead to a better result.

4.2 Support Vector Machine

Support Vector Machine algorithm is a supervised learning method proposed by Vapnik [24]. It is based on the statistical learning theory for both classification and regression. In the binary classification case, the goal is to find an optimal separating hyperplane, which is a line here. It separates the data into two categories and each class lays on either side of the line. The SVM is also called the maximal margin classifier, where margin is the distance between the nearest training spots of each category [25].

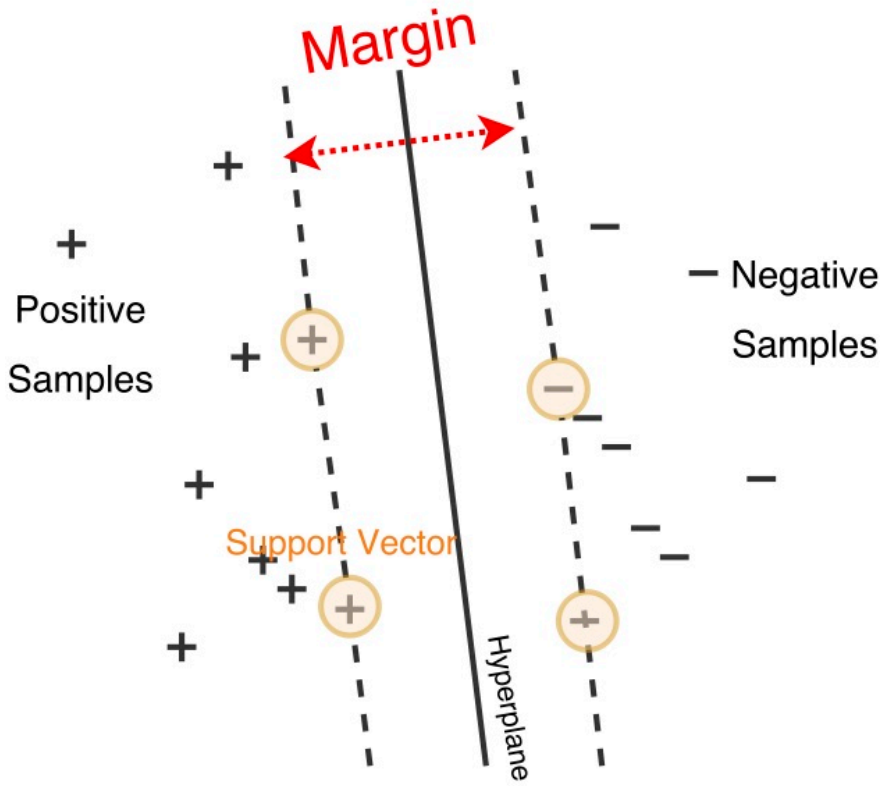

Figure 4.3: Illustration of the Maximum Margin for SVM

Let $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_n\}$ be the dataset and let $\mathbf{y}_i \in \{0,1\}$ be the class label of $\mathbf{x}_i$. A weight vector $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, ..., \mathbf{w}_n\}$ determines the contribution of each input vector to the prediction.

The binary linear classifier separates all points correctly by using

$$y_i(w_i * x_i + b) \geq 1, \forall i. \tag{4.11}$$

Since we want the decision boundary to be as far away from the data points (support vectors) of both categories as possible, we need to maximize the margin, $\mathbf{m} = \frac{2}{\|w\|}$. In other words, the larger the margin is, the less generalization error of the classifier will be. So, we need to

$$\begin{aligned} &minimize\ \frac{1}{2}\|w\|^2, \\ &\text{subject to } y_i(w_i * x_i + b) \geq 1, \forall i. \end{aligned} \tag{4.12}$$

The solution to it can be solved by the Lagrungian form. The process will not be shown in this paper.

4.3 eXtreme Gradient Boosting

eXtreme Gradient Boosting (XGBoost) is developed by Tianqi Chen [26], which uses the gradient descent algorithm to ensemble a set of classification and regression trees (CART). Comparing to only one tree in the CART mode, the tree ensemble model shows stronger predictive ability. These trees will vote for the most promising result. In other words, the gradient descent could minimize the loss by adding new models and correct the errors of existing models [27].

A decision tree is the model that begins with a single non-leaf node that branches into different outcomes. Then the outcomes lead to more additional nodes. Each non-leaf node represents the test on one particular feature, each branch represents the outcome of this feature, and each leaf node stores a classification. Once the split for each feature is made, the one with the minimum loss is viewed as the best split criteria and set it as the rule for that node. The splitting process keeps going until the termination condition is met.

Boosting technique holds the principle that a set of weak classifiers can create a single strong classifier. Weak classifiers are the classifiers only perform slightly better than random classifiers, and strong classifiers are the arbitrarily well-corrected with high accuracy [28].

For the boosting methods, additive training method is applied in each step, which a week classifier is added to the model. In XGBoost, the weak classifier is the new decision tree. Equations below show this hallmark.

$$F_0 = 0 \tag{4.13}$$

$$F_t(x) = F_{t-1}(x) + h(x) \tag{4.14}$$

Here, $\mathbf{h}(x)$ is the new decision tree after $\mathbf{F}_{t-1}(x)$ and $\mathbf{F}_t(x)$ is the full model after $\mathbf{t}-1$ steps. The objective of XGBoost model is to find the tree $\mathbf{F}_t(x)$ that minimizes the equation (4.15) at the $\mathbf{t}^{th}$ step.

$$Obj(F_t) = L(F_{t-1} + F_t) + \Omega(F_t) \tag{4.15}$$

Here, $\mathbf{L}$ is the loss function that decides the predictive power, and $\boldsymbol{\Omega}$ is the regularization function controlling overfitting.

CHAPTER 5

Experiment

5.1 Dataset

The historical stock dataset was gathered from the API of Google Finance by applying the Python library *pandas-datareader*. I chose 20 well-known public companies' stocks traded at both NASDAQ and NYSE, ranging from January 1st, 2010 to December 31st, 2017. There were altogether 1986 trading days. With access to the API, Python can automatically download the daily time series data. The dataset includes daily open price, daily highest price, daily lowest price, daily close price, daily adjusted close price, and daily trading volume.

Figure 5.1 is the candlestick chart with Apple Inc. stock as an example. This figure is used to give a brief depiction of the gathered data. For this figure, with large scale of data, it could be hard to recognize the candle shape. However, we can still figure the trend by noticing that the green means rising and orange means dropping. The red line is the mean price line. The chunk below draws the trading volume of every trading day.

I used the original raw data to calculate the percentage change of price trend, $\mathbf{q}_i$, which was then used as input. The characteristics of stock price is the most direct and intuitive way to understand the trend of stock trend. Comparing to the absolute value of stock price, the percentage change of price trend is more effective in stock prediction. Different stocks have different base prices, which lead to large variation on the absolute value of stock price. By using the percentage change, prediction will be less affected by the price base.

Figure 5.1: Candlestick Chart of Apple Inc. Stock

The percentage change of adjusted price is shown in the following.

$$q_i = \frac{p_i - p_{i-1}}{p_{i-1}} \tag{5.1}$$

After taking the fraction, the adjusted close price sequence was changed to the percentage change sequence.

$$\mathbf{q}_1,\ \mathbf{q}_2,\ \mathbf{q}_3,\ldots,\ \mathbf{q}_T.$$

Besides the historical stock trading data, the dataset also included in the technical indicators of stock trading. In this experiment, three indicators that investors care about are counted in, which are Relative Strength Index (RSI), the Average Directional Movement Index(ADX), and the Parabolic Stop and Reverse (SAR). These three indicators were developed by J. Welles Wilder Jr. [29], and have been widely used in technical analysis softwares. Relative strength index is the momentum oscillator that compares a stock's average recent gains and losses in the certain time period and uses this fraction to measure the variation and speed of price movements. Average Directional Movement Index consists of the positive directional indicator (PDI) and negative directional indicator (NDI). This indicator is used to quantify the strength of trend regardless of whether rising or dropping. Last, the parabolic stop and reverse is the indicator for investors not only to decide the place of trailing stop loss or exit points, but also find the reverse trading position after stopping. Table 5.1 provides the brief introduction of all variables and related formula.

Table 5.1: Description and Implication of the Input Variables

Name	**Definition/Implication**
Panel A. Daily Trading Data	
Open Price	The first price of a stock traded at the beginning of a specified trading day.
Close Price	The last price of a stock in the last transaction on a specified trading day.
Adjusted Close Price	The close price adjusted based on the reflection of dividends and splits.
Highest Price	The highest price when a stock traded on a specified trading day.
Lowest Price	The lowest price when a stock traded on a specified trading day.
Percentage Change of Adjusted Close Price	The percentage change of adjusted close price on $\mathbf{i}^{th}$ trading day to the previous $(\mathbf{i}-\mathbf{1})^{th}$ trading day.
Trading Volume	Total amount of shares or contracts of a stock traded on a specified trading day.
Panel B. Technical Indicators	
Relative Strength Index	$RSI = 100 - \frac{100}{1+RS}$ n = number of RSI periods Average Gain $= \frac{TotalGains}{n}$ Average Loss $= \frac{TotalLoss}{n}$ First RS $= \frac{AverageGain}{AverageLoss}$ Smoothed RS $= \frac{(PreviousAverageGain * 13 + CurrentGain)/14}{(PreviousAverageLoss * 13 + CurrentLoss)/14}$

Table 5.2: Description and Implication of the Input Variables (cont.)

Name	**Definition/Implication**
Average Directional Movement Index	$ADX_t = \frac{(ADX_{t-1} * (n-1)) + DX_t}{n}$ where n = smoothing period $DX_t = \frac{100 * (PDI_t - MDI_t)}{PDI_t + MDI_t}$ $PDI_t = MAX(H_t - H_{t-1}, 0)$ $NDI_t = MAX(L_t - L_{t-1}, 0)$
Parabolic Stop and Reverse	$P_t = P_{t-1} + AF * (EP_{t-1} - P_{t-1})$ P_t = current value of the indicator P_{t-1} = previous value of the indicator AF = acceleration factor EP_{t-1} = Extreme point in the previous period

5.2 Data Processing

The figure below briefly illustrates the procedure of data pre-processing.

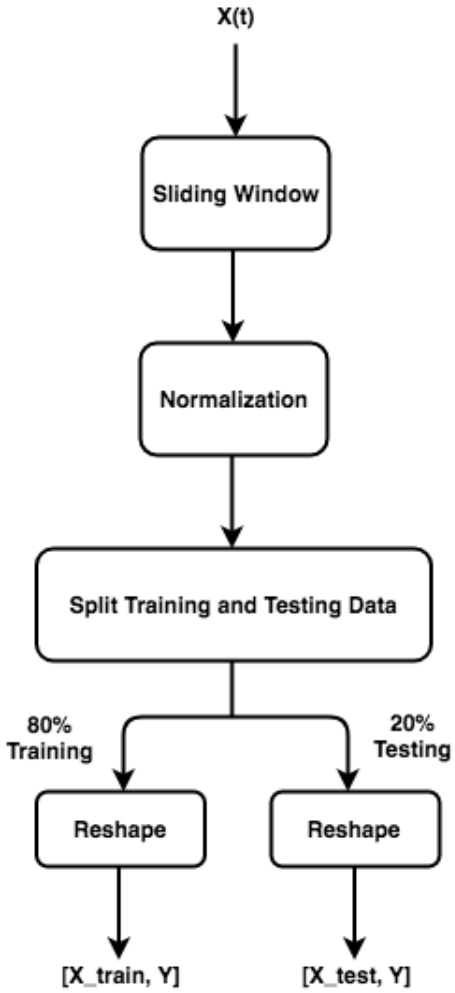

Figure 5.2: Illustration of the Data Pre-processing before Feeding into the Models

1. Sliding window. Window sliding is the strategy of taking the previous time steps to forecast the subsequent time step. Then, with this method, I can transform the data to solve the classification problem. Here I cross-validated the window size, and got the optimal window size of 5 as shown in Figure 5.3. Thus, in this case, I will use a relative small window size to let models relearn and adapt the latest market information. The stock information from the latest week could provide more accurate prediction.

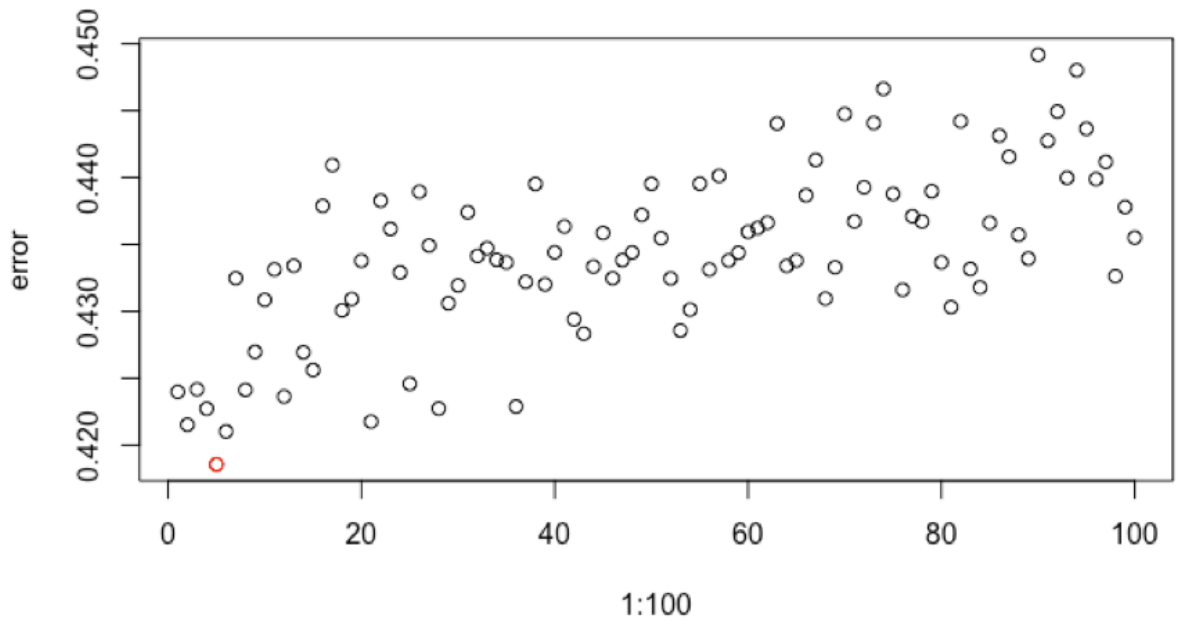

Figure 5.3: Grid Search for Window Size

Figure 5.4 depicts the strategy of sliding window in size of 5. All the inputs with N/A were omitted.

Original		**Slide Window**				
Y_1	X_1	N/A	N/A	N/A	N/A	N/A
Y_2	X_2	X_1	N/A	N/A	N/A	N/A
Y_3	X_3	X_2	X_1	N/A	N/A	N/A
Y_4	X_4	X_3	X_2	X_1	N/A	N/A
Y_5	X_5	X_4	X_3	X_2	X_1	N/A
Y_6	X_6	X_5	X_4	X_3	X_2	X_1
Y_7	X_7	X_6	X_5	X_4	X_3	X_2
...	...	...	...	...	...	...

Figure 5.4: Window Sliding Strategy

2. Normalization.

I used min-max normalization on data, which can scale different indicators of stocks to be comparable. It can be realized by using MinMaxScaler(feature_range=(-1,1)) in Python to transform the original data into range of [-1,1]. That means the minimum value of input is mapped to -1, and the maximum value of input is mapped to 1.

3. Split Dataset.

The dataset was split into chronological sets. I took the first 80% of dataset as the training set, and the rest 20% as the testing set. Afterwards, I reshaped the input to be three-dimensional, [samples, timesteps, features], to fit the network.

5.3 Method

The purpose of this thesis is to predict whether the stock price in the next few days will increase or decrease by using the past few days' price indicators and related technical indicators. For this binary classification problem, I implemented long short-term memory, gated recurrent unit, support vector machine, and eXtreme gradient boosting. In the following sections, I will introduce how I applied these four models to achieve the purpose.

5.3.1 RNN

I used the Python library *Keras* to build and train the models. The *Keras* library can run on top of *TensorFlow* and is the simplified interface to *TensorFlow*. With *Keras*, the built-in function *LSTM* and *GRU* can be used directly. Besides *Keras*, *Pandas* library was included to read and reframe the data.

Models is the core of *Keras* neural network. It represents the neural network that we defined has layers, activation function and so on. The training and testing were also based on this *models*. The *Sequential* model represents that the network is built on a linear stack of layers, which is the basic structure. Dense, Activation, and Dropout are the core layers of neural network. Dense is actually the fully-connected layer. Activation is the activation

function, which uses ReLu, Softmax, or sigmoid function to amend on the results from last layer. When there are too many neurons, the effects will be worsening due to overfitting. Then Dropout will be used to randomly exclude neurons from last layer while training the network. This is a regularization method that can reduce the problem of overfitting effectively and improve the performance of model.

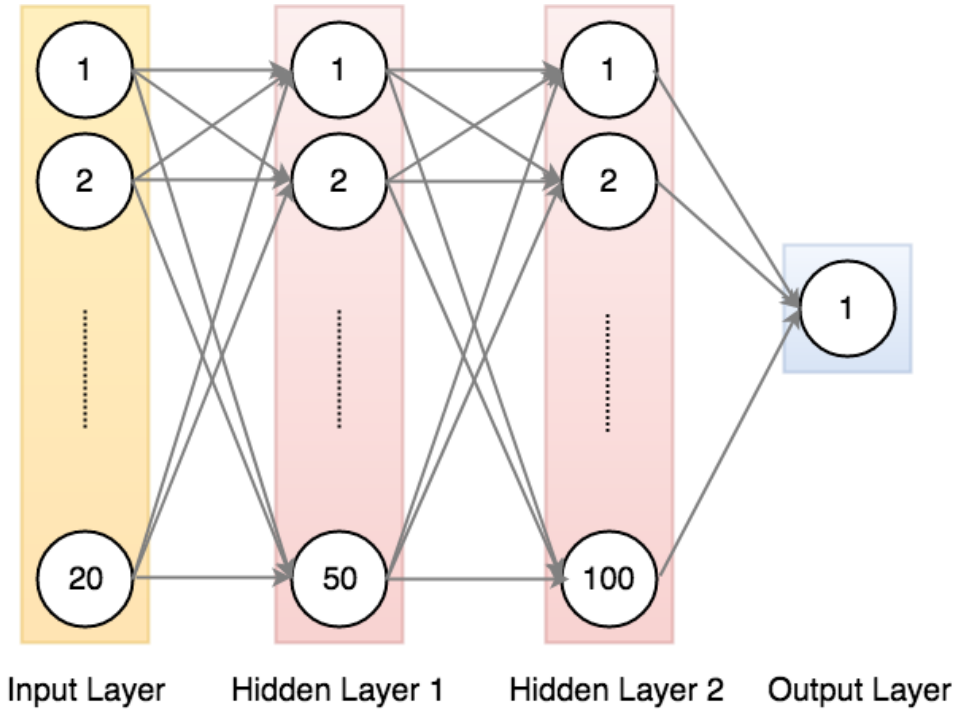

Figure 5.5: RNN Architecture

In this experiment, I used the prior 5 values as timesteps to predict the $\mathbf{6}^{th}$ value, and then compared the predicted and the real $\mathbf{6}^{th}$ value. So, there were altogether 20 units as inputs. Thus, I defined the RNN with 20 neurons in the input layer, 50 neurons in the first hidden layer, 100 neurons in the second hidden layer, and 1 neuron in the output layer. Two 20% dropout layers were added after the first and the second layer. I selected Root Mean Square Error (RMSE) to calculate error, and optimize through *adam*.

The model fit for 50 training epochs with batch size equaling to 80. At last, I applied the

fit function to set the validation data argument in order to record the history of the training and testing loss. The training and testing loss were plotted and is shown in Figure 6.1.

The next step after defining the model was to forecast the entire test set. According to whether the predicted $\mathbf{q}_i$ is larger or smaller than 0, we can turn the problem into classification. After that, I calculated the error for the model based on the prediction and real values in the initial scale.

Above is the methodology for the recurrent neural network. In this experiment, I applied two special cases of RNN, long short-term memory and gated recurrent units, to the hidden layers.

5.3.2 SVM

Support vector machine can be achieved by using *sklearn* in Python. The model is less complex to achieve comparing to the recurrent neural network. With the built-in function, I tuned the parameters to perfect the model.

There are three parameters we need to take care of, which are kernel, regularization, and gamma.

Kernel decides the type of svm. There are different kernel functions that are available to choose, including radial basis function kernel (RBF), linear, sigmoid and so on. Depending on the application and cross-validation method, I could determine which kernel type performs best. In this experiment, I chose linear kernel, which has lower error rate comparing to other kernel types. The equation of linear kernel is listed as follows:

$$K(x_i, x_j) = x_i^T x_j. \tag{5.2}$$

The next parameter needs to be tuned was the regularization, 'C' in *sklearn* library. This term determines how much to avoid the error of misclassifying in the training set and controls the ability of generalization when optimizing. With larger C, the optimization will make the margin of hyperplane smaller, which probably will lead to lower generalization. Low generalization means the model only works well on the training set but poorly on new

sample. It could be caused by the problem of overfitting. With smaller C, the optimization will set the margin larger and reduce the risk of overfitting. Here, after grid search on different C, I selected 100 for the regularization parameter.

The last parameter is gamma. It determines how far the training sample can influence. Higher gamma means close that only points near hyperplane will be considered. Lower gamma means far that far away points will also be counted. However, since I implemented linear kernel, it is not necessary to tune because the default gamma is 1 for linear.

5.3.3 XGBoost

For XGBoost, there exists python package named *xgboost*. After splitting training and testing sets, following parameters need to be modified. For tree booster, there are several types are available, such as *gbtree*, *gblinear* and *dart*. Since the target problem is the classification, it would be better to use *gbtree*. The number of rounds determines the maximum iteration times. It can be tuned by cross-validation and the result showed 1000 performs the best. Then for the objective parameter, I chose binary: logistic, which is used for binary classification. The eta parameter determines the learning rate of grasping the patterns of data and it ranges from 0.01 to 0.3. Higher eta can make computation quicker. I selected 0.3 in this case. The last parameter is the regularization term, gamma, which controls the problem of overfitting. The default = 0 means that there is no regularization. The higher gamma, the higher regularization. I began tuning it with 0 and checked the cross-validation error rate, then took gamma = 5. Besides the parameters above, all others were set as default.

CHAPTER 6

Results

The results are divided into two parts. The first part is the recurrent neural network training results. Since for the 20 stocks, the plots demonstrate similar information, I only illustrated the LSTM method for Apple Inc. as example. The second part is accuracy rate comparison among four models.

Figure 6.1 shows the RMSE of training and testing sets over 50 epochs. We can notice that errors reduce over more epochs. Training set is colored as blue and testing set is colored as orange. Testing set error is higher than training set error in most cases, which is expected.

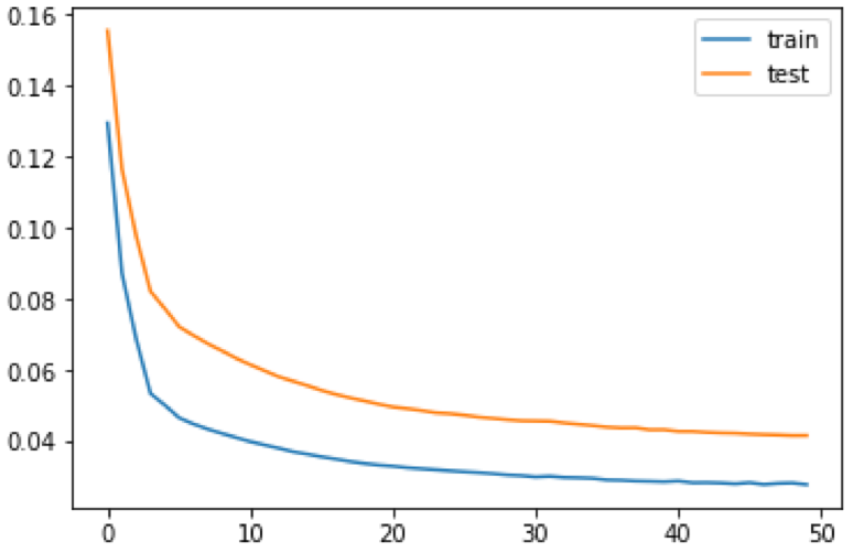

Figure 6.1: LSTM: Classification Error of Training and Testing Performance Over Number of Epochs. Case Study of Apple Inc.

Figure 6.2 gives the predicted and real percentage change of adjusted close price on training set for Apple Inc. It is based on the method of LSTM. The orange one refers to predicted $\mathbf{q}_i$ and the blue one refers to the real $\mathbf{q}_i$. The x-coordinate represents the days. 0 is the middle line and the wave bounces in between -1 and +1. We can notice that not all the time the model can predict correctly on the trend. We will see how it performs by taking a look at the accuracy rate.

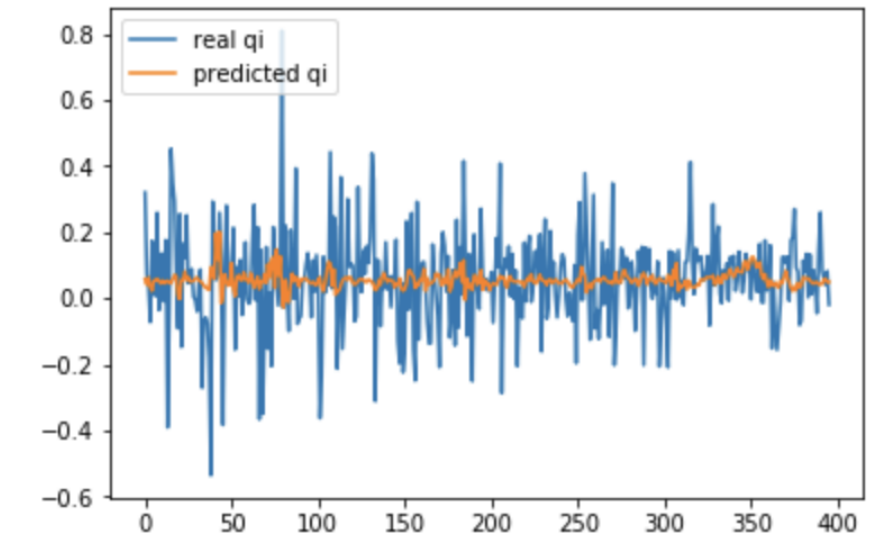

Figure 6.2: LSTM: Predicted and Real Percentage Change of Adjusted Close Price on Training Set. Case Study of Apple Inc.

Figure 6.3 shows the predicted and real classification for Apple Inc. The range is from 0 to 1 where 0 means the trend going down and 1 means the trend going up. The x-coordinate represents the days. We can see several times the models can predict precisely.

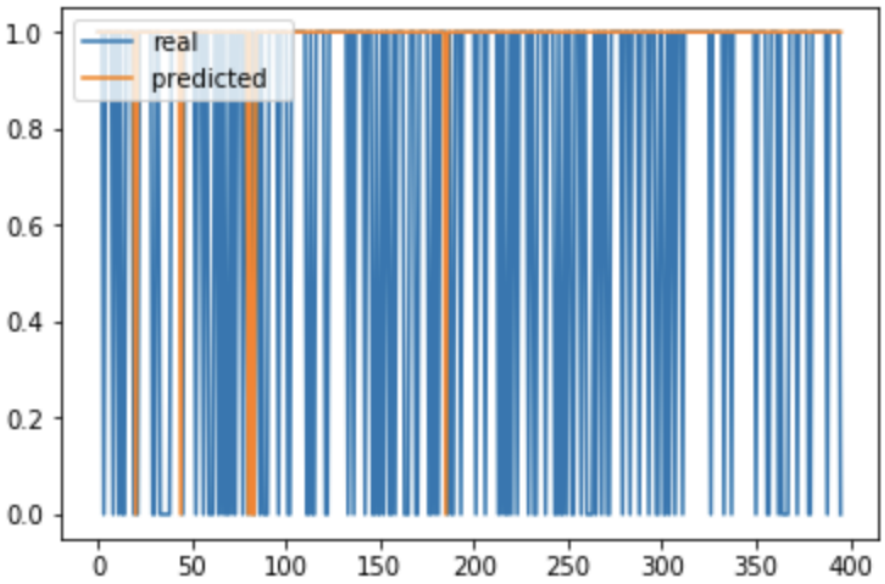

Figure 6.3: LSTM: Predicted and Real Classification. Case Study of Apple Inc.

The following Table 6.1 gives detailed accuracy rate of each stock with different models. Accuracy rate is calculated as following formulas.

Error Rate: The error rate is the percentage of the testing sets that are incorrectly classified.

Error Rate = (Number of Incorrectly Classified Test Sample/Test Sample Size) * 100

Accuracy Rate: The accuracy rate is the percentage of correct classifications.

Accuracy Rate = 1 - Error Rate

From the table, we can notice the accuracy rate is still higher than random guess (50%). Both LSTM and GRU show higher accuracy rate than other two classification methods, which encourages us to dig more into applying recurrent neural network to financial times series problems.

Table 6.1: Accuracy Rate of Four Models

	LSTM	GRU	XGBoost	SVM
AAPL	62.93%	63.43%	59.70%	57.58%
AIG	66.97%	67.47%	59.2%	58.79%
AMZN	61.67%	62.17%	60.76%	56.21%
BA	60.66%	60.40%	59.24%	60.15%
COF	63.43%	63.18%	59.24%	57.12%
EBAY	63.69%	63.43%	59.39%	59.70%
FDX	64.19%	64.44%	60.76%	59.55%
GE	68.48%	68.48%	57.88%	61.21%
GOOG	61.41%	61.92%	59.85%	58.33%
HD	65.45%	65.96%	59.24%	59.70%
JNJ	64.19%	64.44%	58.33%	56.52%
JPM	65.96%	66.21%	62.12%	61.67%
KO	62.17%	62.17%	61.67%	59.24%
MSFT	63.69%	64.19%	60.61%	59.09%
NKE	65.71%	65.45%	61.67%	60.61%
ORCL	63.94%	64.44%	58.03%	58.94%
PEP	66.46%	65.45%	61.36%	59.39%
T	67.22%	66.97%	61.82%	58.64%
WMT	64.44%	64.19%	63.94%	60.91%
XOM	62.42%	62.17%	60.15%	60.45%
AVERAGE	**64.25%**	**64.33%**	**60.25%**	**59.19%**

CHAPTER 7

Conclusion and Future Work

In this thesis, it has shown how four different machine learning methods: long short-term memory, gated recurrent units, support vector machine and eXtreme gradient boosting performed on financial time series trend prediction. In the experiment, 20 U.S. stocks trading at NASDAQ and NYSE were selected to compare the accuracy rate yielded by various models. The results turned out that recurrent neural network did the best comparing to SVM and XGBoost. RNN can have around 5% improvement of accuracy rate. This may suggest that in the time series related prediction, recurrent neural network outstands among other methods.

In RNN, gated recurrent unit performed slightly better than long short-term memory. However, due to limited stock number and relative small data size, we could not arbitrarily conclude that gated recurrent unit is better than long short-term memory in stock prediction. It would be better to compare these two models if data scale is large enough.

In the future, more works can be done to improve the performance of stock prediction.

1. Import more Indicators. In this experiment, I only included three indicators raised by J. Welles Wilder Jr. Actually, there are more indicators to evaluate stocks while trading. Including more indexes could possibly improve the accuracy for models to analyze.

2. Stack Models. Here, I only compared single models with each other. We could discover more by stacking models to see if it can improve the prediction ability.

3. Import Textual Information. Some scholars have used sentiment analysis on Twitter [30] to predict the stock market movement. Besides breaking down social media information, other qualitative indicators like news, international and domestics policy changes

can also be used as inputs to predict price trend.

4. Predict Exact Price Value. John Hull [31] in his book *Options, Futures Other Derivatives* introduced mathematical finance solutions, such as Ito's Lemma, Black-Scholes-Merton model, to study the pricing of financial products. In the future, it could be possible to combine the mathematical finance ways and machine learning methods to determine and predict the exact price value.

Although most people hold negative view on using machine learning to predict stock price as they believe stock market is too volatile to predict, scholars and traders from big companies are still trying to explore more indicators and models to accurately forecast the economic market. We can speculate that in the near future, the machine learning methods will provide us a bright future in learning and mastering the market.

REFERENCES

[1] Fama, E. F. *The Behavior of Stock-Market Prices*. Journal of Business, 1965; 38(1), 34105.

[2] Malkiel, B. G. *A Random Walk Down Wall Street*. New York: Norton, 1973.

[3] Wang, B., Huang, H., Wang, X. *A Novel Text Mining Approach to Financial Time Series Forecasting*. Neurocomputing, 2012; 83(6):13645.

[4] Markowitz, H. *Portfolio selection[J]*. The Journal of Finance, 1952; 7(1): 77-91.

[5] Fung, G. P. C., Yu, J. X., Lam, W. *News Sensitive Stock Trend Prediction[M]*. Advances in knowledge discovery and data mining. Springer Berlin Heidelberg, 2002; 481-493.

[6] Cavalcante, R.C., Brasileiro, R.C., Souza V.L.F., Nobrega, J.P., Oliveira, A.L.I. *Computational Intelligence and Financial Markets: A Survey and Future Directions.*. Expert Systems with Applications, 2016; 55:194211.

[7] Maciel, L.S., Ballini, R. *Design A Neural Network for Time Series Financial Forecasting: Accuracy and Robustness Analysis*. Instituto de Economia, Universidade Estadual de Campinas, Sao Paulo-Brasil, 2008.

[8] Wilson R. L., Sharda R. *Bankruptcy Prediction Using Neural Networks.* Decision support systems, 1994; 11(5), pp. 545-557.

[9] Hsieh, T., Hsiao, H., Yeh, W. *Forecasting Stock Markets Using Wavelet Transforms and Recurrent Neural Networks: An Integrated System Based on Artificial Bee Colony Algorithm*. Applied Soft Computing, 2011; 11(2), pp.2510-2525.

[10] Kamijo, K., Tanigawa, T. *Stock Price Pattern Recognition: A Recurrent Neural Network Approach*. In Proceedings of the international joint conference on neural networks (IJCNN), 1990; (pp. 215221), Washington DC.

[11] Hochreiter. S., Schmidhuber, J. *Long short-term memory*. Neural Computation, 1997; 9(8):17351780

[12] Cho, K., Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwen, H., Bengio, Y. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. in Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 2014; arXiv:1406.1078.

[13] Akita R., Yoshihara A., Matsubara T., Uehara K. *Deep learning for stock prediction using numerical and textual information[C]*. Computer and Information Science (ICIS). IEEE/ACIS 15th International Conference on. IEEE, 2016: 1-6.

[14] Persio, L., Honchar, O. *Artificial Neural Networks architectures for stock price prediction: comparisons and applications*. International journal of circuits, systems and signal processing, 2016; 10.

[15] Chen, W, Zhang, Y, Yeo, C.K., Lau, C.T., Lee, B.S. *Stock market prediction using neural network through news on online social networks[C].* Smart Cities Conference (ISC2), 2017 International, 2017: 11-2.

[16] Prasaddas, S., Padhy, S. *Support Vector Machines for Prediction of Futures Prices in Indian Stock Market.* International Journal of Computer Applications, 2012; 41(3):226.

[17] Tay, F.E.H., Cao, L. *Application of support vector machines in financial time series forecasting.* Omega, 2001; Vol. 29, No. 4, pp.309317.

[18] Jain A., Menon M.N., Chandra S. *Sales Forecasting for Retail Chains.* India, 2015.

[19] Sawon, M., Hosen, M. *Prediction on large scale data using extreme gradient boosting.* [Thesis Report] Thesis Report, BSc (Computer Science and Engineering), BRAC University, Dhaka, Bangladesh, 2016.

[20] Menkveld, A. J. *High frequency trading and the new market makers[J].* Journal of Financial Markets, 2013; 16(4): 712-740.

[21] Hinton, G.E.: Recurrent neural networks, `https://www.cs.toronto.edu/ hinton/csc2535/notes/lec10new.pdf`

[22] Palangi, H., Ward, R., Deng, L. *Distributed Compressive Sensing: A Deep Learning Approach.* IEEE Transactions on Signal Processing, 2016; 64(17):450418.

[23] Olah, C.: Understanding LSTM Networks `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

[24] Vapnik, V. N. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[25] Boser, B.E., Guyon, I., Vapnik, V.N. *A training algorithm for optimal margin classifiers.* In Proceedings of the Fifth Annual Workshop of Computational Learning Theory, 1992; 5, 144152. Pittsburgh, ACM.

[26] Chen, T.Q. *Introduction to Boosted Trees.* University of Washing Computer Science. University of Washington, 22 Oct, 2014. Web.

[27] Brownlee, J.: A Gentle Introduction to XGBoost for Applied Machine Learning - Machine Learning Mastery `https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/`

[28] Freund, Y., Schapire, R. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.* Journal of Computer and System Sciences, 1997;55(1), pp.119-139.

[29] Wilder, J. *New concepts in technical trading systems.* Greensboro, N.C.: Trend Research, 1978.

[30] Yang, S., Mo, S., Liu, A. *Twitter financial community sentiment and its predictive relationship to stock market movement.* Quantitative Finance, 2015; 15(10), pp.1637-1656.

[31] Hull, J. *Options, Futures, and Other Derivatives, Global Edition*, 2017; Pearson Education UK.