COMPARATIVE STUDY ON INFLUENCING FACTORS BETWEEN RNN AND LSTM ON STOCK INDEX PREDICTION

ZHANGHAN

XIAMEN UNIVERSITY

MALAYSIA 2021

XIAMEN UNIVERSITY MALAYSIA

厦門大學馬來西亞分校

FINAL YEAR PROJECT REPORT

COMPARATIVE STUDY ON INFLUENCING FACTORS BETWEEN RNN AND LSTM ON STOCK INDEX PREDICTION

NAME OF STUDENT	:	ZHANGHAN
STUDENT ID	:	CST1709486
SCHOOL/ FACULTY	:	SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
PROGRAMME	:	BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND TECHNOLOGY (HONOURS)
INTAKE	:	2017/09
SUPERVISOR	:	DR. SAIF KIFAH ASSISSTANT PROFESSOR

AUGUST 2021

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Xiamen University Malaysia or other institutions.

Signature : 

Name : ____ZHANGHAN_____

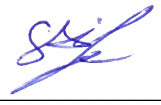
ID No. : ____CST1709486_____

Date : _____10/07/2021_____

APPROVAL FOR SUBMISSION

I certify that this project report entitled "COMPARATIVE STUDY ON INFLUENCING FACTORS BETWEEN RNN AND LSTM ON STOCK INDEX PREDICTION" was prepared by ZHANG HAN has met the required standard for submission in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science and Technology (Honours) at Xiamen University Malaysia.

Approved by,

Signature : 

Supervisor : Dr. Saif Kifah

Date : 10/07/2021

ACKNOWLEDGEMENTS

I would like to thank all who have contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Asst. Prof. Dr. Saif Kifah for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who have helped and given me encouragement.

ABSTRACT

Since the birth of the stock market, people have been thinking about making a profit on stock investment. It is also actively investigating stock price forecasting, continuously employing various data models, machine learning, and data mining to anticipate the future trend of stock prices in order to generate enormous profits. Neural network machine learning methods are frequently utilized among them. This is owing to neural networks' strong self-learning, stability, and abstract simulation capabilities. For compared to mathematical models in statistics and econometrics, neural networks have greater benefits when forecasting financial time series. This thesis determines to use RNN and LSTM to do regression prediction. Guide stock investors by predicting the long-term trend of stocks. After in-depth research on the problems faced by stock price forecasting and research on a variety of stock price forecasting methods, data cleaning, composition analysis, and normalization are used to process data. In the empirical work, RNN and LSTM neural network are used to make short-term predictions on GOOGL stocks and compare the accuracy of the models. Compare the effects of different input variables on models and the differences between models by constructing 4 different models. In view of the problem of excessive stock price factors, only variables such as Open, Close, High, Low, Adj close, Volume were selected in this study. The results show that RNN and LSTM have long-term predictive effects on the stock market and perform similarly, and RNN is more likely to fall into overfitting. PCA technology can perform data dimensionality reduction, but the data after dimensionality reduction has changed, or some information has been lost, and it is impossible to accurately predict the direction of the stock market.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

X_p	Pre-processed data value
X	Unprocessed data value
Min X	Minimum value of X
Max X	Maximum value of X
$\frac{\partial y}{\partial x}$	The partial derivation of y with respect to x
$\nabla_x y$	The gradient of L_i with respect to o^t
A^T	The transposition of A
$diag(A)$	Diagonal matrix
$(\nabla_{\vec{f}^t} L)_i$	The i-th component of $\nabla_{\vec{f}^t} L$.
$\odot$	Hadamard product
η	Learning rate

RNN	Recurrent Neural Network
LSTM	Long-short term memory
EMH	Efficient Market Hypothesis
AR	Auto-Regressive
MA	Moving Average
ARMA	Auto-Regressive Moving Average
ARIMA	Autoregressive integrated moving average
SVM	Support vector machine
LS-SVM	Least square support vector machine
ANN	Artificial Neural Network
TOPIX	Tokyo Stock Prices Indexes
GA	Genetic Algorithm
BPLT	Linear transformation with the backpropagation
GAFD	GA approach to feature discretization
LSTNet	Short Time-series Network
PCA	Principal component analysis

CHAPTER 1

INTRODUCTION

1.1 Background Study

During the Covid-19 pandemic period, the public spend more and more time online activities. As the pandemic has a direct impact on people's consumption and investment behavior and risk appetite form an economical aspect, younger investors are more and more enthusiastic about investing (Susan, 2021). But the stock market is non-linear and unpredictable. This reason has caused many people to fail in investment. Predicting the stock index has become a very important and practical topic.

The stock market is a marketplace where companies' stocks and derivatives may be traded at agreed-upon prices. The stock market is driven by the interplay between supply and demand for equities (Reza, Mahmood, & Hassan, 2010). The stock market is one of the most rapidly growing sectors in any country. Many persons are now connected to this department, either indirectly or directly. As a result, it's critical to be aware of market trends. People are interested in forecasting stock values as the stock market develops. Stock price prediction has become a difficult endeavor due to the dynamic nature and quick fluctuation of stock prices. The stock market is primarily a chaotic system that is non-parametric, non-linear, noisy, and deterministic.

Machine learning approaches such as text analysis technology and neural networks have gotten more mature as statistical science and computer technology have progressed. Text mining and other approaches are increasingly being used by researchers to characterize the influence of monetary policy and market irrationality on stock prices. At the same time, the neural network model, being a very sophisticated nonlinear artificial intelligence system, has practically no constraints for the distribution of input data. This method's self-organization and self-adjustment characteristics make it more suited to dealing with stock price forecasting's numerous influencing elements, random-like complicated nonlinear situations. (Zhou, 2019)

1.2 Research Motivations

Stock price forecasting is the process of predicting future stock price changes based on relevant information from the preceding period, such as historical transaction data, national policy, and so on. Based on contemporary financial ideas about stock price movements and computer technologies, The situation's prognostic behavior.

Effective stock price forecasting may help investors not only make better stock investment decisions and improve the stock market's capacity to service the actual economy, but it can also help them analyze financial risk variations. It has significant guiding and practical implications for the country's stock market macro-control, fostering steady economic and social growth.

The motivation is to build an effective model to predict stock price and analysis factors that influence stock price, to provide practical experience on stock market investment and forecast coming problems of the finance market.

1.3 Problem Statements

Stock prices are a nonlinear, inherently unstable, and unpredictable time series that are influenced by a variety of variables. Predicting such a time series is extremely challenging. When it comes to stock price forecasting, there are numerous aspects to consider.

To begin with, stock price data has a lot of white noise in the time series. If a statistical model is employed to predict the stock price, the time series must first be preprocessed before being fed into the model. The results will be out of order if this is not done. The neural network, on the other hand, does not require preprocessing, ensuring that the data is real and that the results acquired are more accurate.

Additionally, the nonlinearity of stock price time series was then shown by a variety of evidence. Because of this, standard multiple regression and linear regression aren't relevant to this problem, posing significant challenges for many researchers. This nonlinear sequence can only be properly described using complex mathematical models or machine learning approaches.

Finally, when stock investors make their investing decisions is unknown. Many

elements of knowledge may affect a stock investor's investing strategy and portfolio, but buying and selling transactions may also be intuitive. As a result, investors' psychological expectations are unpredictable. (Sun, 2015)

To summarize, numerous factors will influence the stock price projection, and the high noise and nonlinear features make the sequence difficult to describe. At the same time, many investors' psychological expectations, which influence stock prices, are restless.

1.4 Objectives

There are several objectives of this thesis:

a) Provide a high-quality, well-organized database for stock forecasting models.
b) Compare the difference between RNN and LSTM in predicting stock price.
c) Comparison of various input parameters leads to different prediction results.
d) Adjust various parameters to get the best model.
e) Provide some theoretical and practical value for investors.

1.5 Scope

The focus of this thesis is to compare the difference between RNN and LSTM models for stock price prediction on different input data. The research sample is GOOGL stocks from January 1, 2014 to December 31, 2019.Include the following function:

a) Forecast the closing price of GOOGL
b) Dimensionality reduction input data through principal component analysis
c) Compare the prediction effect of different input data
d) Compare the prediction effects of different models

1.6 Challenges

These challenges may be encountered in the process of building a model:

a) Collect data and construct extended data from basic data.
b) Apply principal component analysis to reduce the dimensionality of input data.
c) The influence of different parameters on the model and the prevention of overfitting.
d) Improve the prediction accuracy of the model and determine the different factors that affect the performance of the model.
e) Find the best hyper-parameters for each model.

1.7 Thesis Organization

The following chapters are organized as follows: literature review of different methods in history related to predict the stock market will be placed in the Chapter 2. The method, work flow, data processing, methods adopted to forecast stock index are in Chapter 3. The Chapter 4 mainly includes the process of real experiments, demonstrating the code of PCA approach and models construction. Chapter 5 compares the performance of different models and different variables in stock market prediction. The final conclusion and recommendation will be included in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Stock market forecast has become a popular issue in finance, computer science, and mathematics due to the potential financial benefit. Because it deals with a big amount of money, the stock market is considered a peak investment source. Stock market forecasting also includes whether the financial market is predictable. Since there has been no consensus on the validity of the Efficient Market Hypothesis (EMH), which claims that the market is efficient and there is no possibility for prediction, researchers have attempted to show the predictability of the financial market. (Lawrence, 1997)

Because of quicker computers and the vast amount of information available on the Internet, stock markets have become more accessible to strategic investors and the general public. Because the Internet is the most significant and fastest source of information, stock prices are heavily influenced by data form different source online, therefore using Internet data to generate stock price forecasts has become a major concern. Many academic and business experts have suggested numerous prediction algorithms and models to properly forecast the stock market. Many academic and business experts have suggested numerous prediction algorithms and models to properly forecast the stock market.

For stock price prediction, there are three common methods: technical analysis, traditional time series forecasting, and machine learning. Those approaches will go through these approaches in the following of this chapter.

2.2 Technical Analysis

The analysis of past market data, such as price and volume, is known as technical analysis. Technical analysts combine market psychology, behavioral economics, and quantitative analysis to forecast future market behavior based on previous performance.

2.2.1 Intrinsic Value Analysis Theory

The basic assumption of intrinsic value analysis theory is that at any point in time, the

cleintrinsic value of a single security, or according to economists, the equilibrium price depends on the income potential of the stock. The profit potential of securities again depends on basic factors such as management quality, industry and economic prospects. (Fama, 1995)

Some authorities reckon that the past price behaviour pattern of a single security will tend to reappear in the future. Therefore, the method of predicting stock prices including increasing potential returns is to be familiar with past price behaviour patterns to identify possible repetitions. So in this case, many stock analysts will try to use the knowledge of the past behaviour of the price series to predict the possible future behaviour of the price series.

Overall, stock analysts should be able in principle to determine whether the actual price of the security is higher or lower than its intrinsic value. If the actual price tends to move towards the intrinsic value, then trying to determine the intrinsic value of the security is equivalent to predicting its future price.

2.2.2 Financial Statement Analysis

The financial statement analysis method combines a large number of financial statement items into an overall measurement index to indicate the direction of the profit rate change in the next year. By analysing relevant aspects of financial statements and investment decisions. Get the value of the company. This study assumes that the market price is sufficient to determine the company's value, as the basis of measurement information in accounting measurement.

Accounting attributes are inferred to be value-related because they are also statistically related to stock prices. Stock prices sometimes deviate from these values, only slowly moving closer to the basic value. Therefore, an analysis of the published financial statements can reveal that this value is not reflected in the stock price. The intrinsic value in financial statements does not use price as a value benchmark but is used as a benchmark to compare with prices to determine stocks whose stock prices are too high and too low. Since deviated prices will eventually be attracted by fundamentals, by comparing prices with these fundamental values, investment strategies that generate "abnormal returns" can be discovered. (Jane & Stephen, 1989)

Ball and Brown (1968) concluded the many successive 'information content' papers indicate that accounting earnings and some of its components capture information that is contained in stock prices. Accounting revenue figures account for about half of the net effect of all available information in the 12 months before the announcement; therefore, the annual accounting report is only one of the many sources of information available to investors, and investors use this information to judge the changes in the company's stock price.

Ou and Penman (1989) used financial statement analysis to trade positions between 1973 and 1983, which included the cancellation of long and short positions with zero net investment. The return on long and short positions over the two-year period was approximately 12.5%. After adjusting the "size effect", the return is about 7.0%.

Robert and David (1992) studied the profitability of a trading strategy based on the logit model, which aims to predict the signs of excess returns in the following twelve months based on accounting ratios. During the period from 1978 to 1988, the average annual excess return generated by this trading strategy was between 4.3% and 9.5%, depending on the specific measurement and weighting scheme of the excess return involved. Their overall results support Ou and Penman's (1989) argument that financial statement items can be combined into a summary indicator to gain insight into the subsequent trend of stock prices.

However, their trading strategy is not directly based on a model that predicts excess returns, but a forecast based on the measurement of excess returns. The implementation of Robert and David's trading strategy based on the earnings prediction model seems to have achieved more benefits than Ou and Penman have implemented. Trading strategy based on the profit forecast model. After comparing trading strategies, they believe that the profitability of Ou and Penman's trading strategies is more fragile than what they implied in their tests during the 1973-1983 period. (Robert & David, 1992)

2.3 Time Series Analysis

For time series prediction, traditional statistical models are frequently employed in

economics. These models are capable of simulating linear connections between market-influencing variables and market value. There are two forms of time series forecasting in economics: univariate (simple regression) and multivariate (multivariate regression) (multivariate regression).

The ARMA model, developed by Box and Jenkins and sometimes known as the B–J technique, is a frequently used example of a univariate model. In the recurring equation, there is just one variable. AR model, MA model, ARMA model, and ARIMA are the four main forms of ARMA models . (Ayodele & Aderemi, 2014)

For the lengthy process of stock price prediction, Adebiyi and Adewumi (2014) developed an ARIMA model. The best ARIMA model's experimental findings demonstrate that the ARIMA model has the ability to forecast good short-term stock prices. This can help stock market investors make effective investing selections. According to the findings, the ARIMA model can compete effectively in short-term forecasting with new forecasting methods.

2.4 Machine Learning Methods

The machine learning approach uses past data to train and detect patterns in the data, ultimately achieving the goal of prediction. Machine learning algorithms have been used to anticipate financial time series more and more in recent years. The two fundamental goals of time series analysis are categorization and regression. The rise and fall of stocks is judged by classification, and the price of stocks is predicted using regression. SVM and neural networks are two popular regression techniques.

2.4.1 Support Vector Machine(SVM)

Vapnik and his colleagues created the Support Vector Machine (SVM) in the late 1970s, based on statistical learning theory. Because of its usefulness in classification and regression tasks, notably in time series forecasting and financial applications, it becomes increasingly adopted in stock index prediction research. (Yang, Chan, & King, 2002)

The support vector machine is distinguished from other learning algorithms by the capacity control of the decision function, the use of the kernel function, and the sparsity

of the solution. The structural risk reduction idea, which is a unique theory, is the foundation of the support vector machine. It calculates the function by lowering the upper bound on the generalization error, proving that it is resistant to overtraining and, in the end, attaining outstanding generalization performance. Another important characteristic of SVM is that it is analogous to tackling linear restricted quadratic programming problems when it is being trained. As a result, the SVM solution is generally unique and globally optimum.

One of the most well-known studies on the use of support vector machines for stock market prediction is Kim (2003). For comparative reasoning, he used support vector machines to anticipate financial data and compared them to backpropagation networks and case-based networks (CBR). The support vector machine outperforms the back propagation network CBR, according to the findings of the experiments. The support vector machine offers a stronger generalization impact than previous approaches because it uses the idea of reducing structural risk.

This study by Hengshan and Phichhang (2009) looked at 10 different data mining techniques and how they were used to forecast price movements in the Hong Kong stock market Hang Seng Index. In other models, experimental data demonstrate that SVM and LS-SVM have outstanding prediction ability. SVM is superior than LS-SVM in terms of in-sample prediction, while LS-SVM is superior than SVM in terms of hit rate and error rate criteria for out-of-sample prediction.

2.4.2 Neural Networks

Neural network is a method of processing information similar to the nervous system, based on the relevant knowledge of the network topology as the theoretical basis. Artificial neural networks can handle various problems with large concurrency and complex data. After the neurons in the neural network obtain external information, they can train the neural network through the network structure between the internally connected neurons, and use appropriate learning algorithms to train the neural network., So as to get the required output data. Artificial neural network has the following characteristics: high concurrency, nonlinear and adaptive ability, because of these characteristics of neural network, so ANN is widely used in time series prediction,

image processing, scientific control and other problems.

Instead of assuming the functional form of the relationship, learn the relationship via the facts. Because NN is a general-purpose approximator, any connection may be modelled to any degree of precision when sufficient modelling data is provided. Furthermore, it allows for noise and imperfect data representation to be tolerated to some extent. The neural network, on the other hand, has nonlinear, non-parametric adaptive learning properties and provides the most practical modelling and prediction impact. The nonlinear structure of neural networks suggests that they have a lot of promise for tackling a wide range of issues. Neural networks can be used in the stock market because of the qualities listed above. (Paul & Maria, 2007)

In terms of drawbacks, NN has a black box problem in that it hides the importance of each variable and how the independent variables are weighted (Lawrence, 1997). It is hard to comprehend how the network creates future stock prices since the specific role of each variable cannot be established.

White (1988) attempted to estimate IBM's common stock daily returns using neural networks, but the results were less than ideal. There is no certainty that a global minimum was reached because the optimization approach is basically local. Although the final weight values were chosen to provide the optimum performance over a wide variety of beginning values for our iterations, there is no certainty that a global minimum was obtained.

A Tokyo stock trading time prediction system (TOPIX) based on a modular neural network was created by Takashi and Kazuo (1990) and others. The forecasting system produced accurate predictions, while the stock trading simulation yielded good results.

Hirotaka and Michitaka (2001) presented a neural network model for technical analysis of stock market predictions based on their findings and applied it to the TOPIX Index's trading time forecasting system. A learning approach that aids in improving the accuracy of other, more significant categories' predictions. Using the significance information of each category, this technique limits the amount of learning examples. Experiments using real data demonstrate that the forecasting method

provides buy and sell signals at more opportune times and creates better profits than using each technical indicator individually.

Kyoung and Ingoo (2000) presented a method for predicting stock price indices by utilizing Genetic Algorithms (GAs) to discretize features and calculate the weights of neural networks. The learning method is improved by using a genetic algorithm, which also decreases the complexity of the feature space. At the same time, the evolutionary algorithm optimizes the connection weights between layers and the feature discretization threshold. The results demonstrate that the evolutionary algorithm outperforms the other two traditional models, BPLT and GAFD, on the feature discretization model.

Lawrence (1997) observed that when training NNs for high noisy data is challenging, the networks default to a naive approach, such as always forecasting the most common output. Furthermore, when the input data is highly dimensional, NNs have certain limits in learning patterns. Dash and Liu (1997) emphasized feature selection, claiming that decreasing the number of input variables might enhance model performance for a given data set in some cases. Reducing and transforming unnecessary or duplicate characteristics can speed up the process and produce more generic results.

Hopfield (1983) invented feedforward backpropagation networks with feedback links. Recurrent Neural Network (RNN) is a type of neural network model in which many connections between neurons form a directed loop. The RNN's internal state or memory structure is created as a result of this. The output of the preceding layer, as well as the output of the hidden layer node at the previous instant, are included in the input of the hidden layer node in the RNN neural network. This ring structure is well suited to store historical data in order to accomplish the goal of time series modeling.

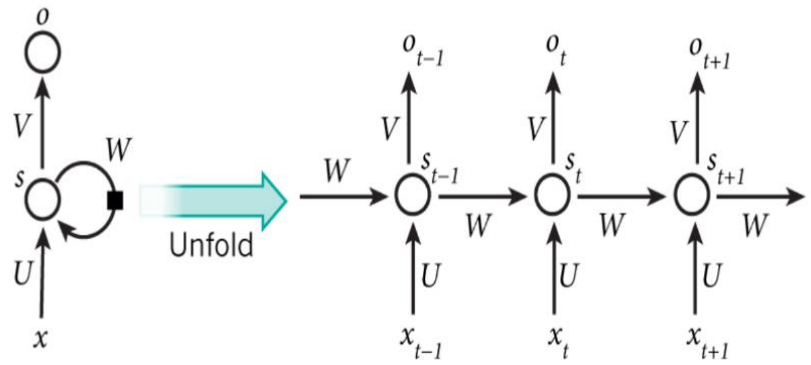


Figure 2.1: An expansion of RNN. (Mao, 2017)

For Chinese stock market predictions, Chen, Zhou and Dai (2015) utilized LSTM. Our LSTM model enhances the accuracy of stock return prediction from 14.3 percent to 27.2 percent when compared to the random prediction technique. Normalization is highly beneficial for boosting accuracy, according to the findings (19.2 percent vs 15.6 percent). The Shanghai Composite Index's accuracy has increased even more (24.1 percent compared. 20.1 percent), which is consistent with the idea that market indexes influence stock returns. As a result, various sets of stocks will impact the forecast's accuracy, therefore different types of stocks must be predicted individually. (Chen, Zhou, & Dai, 2015)

In this study, Chang, Lai, Yang, and Liu (2018) present the Long- and Short Time-series Network (LSTNet), a unique deep learning architecture for forecasting stock prices. Using Convolutional Neural Networks (CNN) and Recurrent Neural Networks, LSTNet identifies short-term local dependence patterns between variables and finds long-term patterns of time series trends (RNN). LSTNet has produced significant performance increases using a variety of state-of-the-art baseline methods. It successfully catches short- and long-term recurring patterns in the data, and it uses a combination of linear and non-linear models to make accurate predictions.

Fischer and Krauss (2018), among others, used the long-term short-term memory network to anticipate the S&P 500 index on a wide scale from December 1992 to October 2015. The results of LSTM are compared to those of random forest, conventional deep net, and basic logistic regression. They discovered that LSTM, which is an intrinsically appropriate approach for this field, outperformed both Standard deep net and logistic regression by a wide margin. Except during the global

financial crisis, random forests outperformed the market for the most part. Their findings demonstrate that the daily statistically and economically significant return of 0.46 percent before transaction expenses offers a clear challenge to the semi-strong form of market efficiency, and that deep learning may have played a role in this sector before 2010. A useful approach for predictive modelling.

2.5 Summary

This chapter looked at recent advancements in stock market prediction models in general. The technical analysis method is used to study the trend of a company's stock price changes by comparing various prediction models, as well as the fundamental aspects of macro national economic policy, the global economic situation, the company's basic profitability, and future industry development prospects. It primarily depends on quantitative indicators of stocks, bonds, and other securities, which are useful for short-term market research, but it is difficult to anticipate the long-term trend of prices. Based on two main theories: a single security's previous price behaviour pattern will tend to recur in the future, and price movements have a tendency to self-influence.

The basic principle of the time series analysis method is to use statistics and econometric models to fit the stock price trend. The main research methods include regression analysis and moving average methods. ARIMA (ARMA) is one of the regression analysis methods. Simulate the dependent variable of the sequence based on the independent variable of time. Non-stationary and nonlinear time series, such as stock prices, are not covered. The data may ultimately be put into the model for prediction since the model pre-processes it. As a result, the stock has fundamentally altered. Because the original data was altered, the projected data is no longer accurate or reasonable. NNs have been compared to the multivariate statistical model in a number of studies. Although multivariate models were formerly frequently employed to forecast stock market movements, they are increasingly being replaced by a variety of machine learning approaches.

For machine learning methods, NNs are more accurate than other existing approaches in predicting market directions. Because NNs can learn nonlinear connections from training input/output pairs, they can more precisely simulate nonlinear dynamic

systems like stock markets. Up to now, Long and short-term memory neural networks (LSTM) and the latest form of recurrent neural network (RNN) have also been shown to be useful for time series forecasting studies. And because these two types of neural networks are based on dynamic neural network theory, which can completely capture the dynamic characteristics of the stock market system, the RNN model prediction is utilized in this study.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter mainly introduces the methodology used in this thesis. The research methodology is conducted 4 stages in total, they are data collection, data pre-processing, construct and evaluate model. The overall flow chart is as shown in Figure 3.1. It will be explained in detail in the following of the thesis.

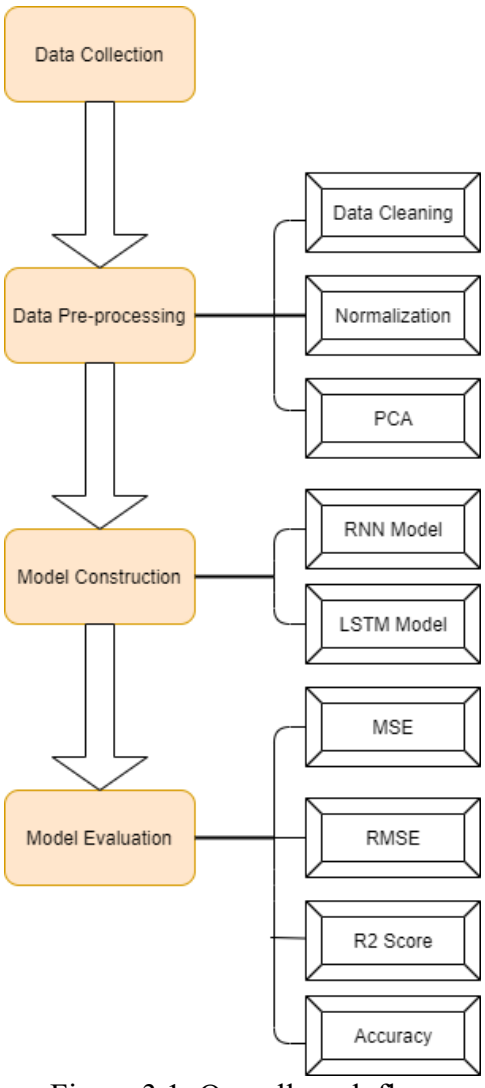


Figure 3.1: Overall work flow

3.2 Data Collection

Since the closing price often reflects the focus of market funds on a certain stock, it has the function of predicting the direction of the next trading day. Therefore,

predicting the stock price is generally predicting its closing price. In this article, the author chooses the closing price of GOOGL stock as the object of our forecast. Download GOOGL's historical data on the yahoo finance website. Data generally include Open, high, low, close, Adj close and volume.

3.3 Data Pre-processing

As the 6 index data are collected, it is necessary to do data pre-processing. For the first step, the author checks whether there are missing values in the dataset. Generally, there are two options leaving, one is deleting variables or filling in the data. For the processing of outliers, various filters can be used to eliminate them in the linear case, but those filters cannot be adopted in the non-linear case, because this may represent a pattern or a precursor to stock price changes. If filters eliminate the outliers, the data may lose some important information.

3.3.1 Normalization

After data pre-processing, data transformation is performed. This is duo to the different indexes in the input data have different dimensions and dimensional units. For example, there is a big difference in the order of magnitude between the close price and the volume. This situation will affect the training of the neural network. And the results of data analysis. In order to eliminate the influence between indicator dimensions, data normalization is required. normalization can also reduce the amount of calculation and speed up the convergence speed of the model training process, the data needs to be normalized before being brought into the model for training. This article uses the minimum and maximum normalization method to normalize the data to range (0,1). The normalization function is:

$$X_p = \left(\frac{X - Min\ X}{Max\ X - Min\ X}\right) \tag{3.1}$$

3.3.2 Principal Component Analysis

After the normalization, the 6 variables (Open, High, Low, Close, Adj close and Volume) are all normalized. Those independent variables often have overlapping information, and too many independent variables will increase the complexity of the problem. Therefore, we hope to reduce the dimensionality of variables while preserving the main information. Therefore, this thesis intends to use principal component analysis to reduce the dimensions of the variables.

Principal component analysis was proposed by Hotelling in 1933. This method uses orthogonal transformation to transform a group of potentially correlated variables into a group of linearly uncorrelated variables, and then according to the built model, the transformed variables, namely principal Select fewer comprehensive indicators from the components to reflect as much detailed information of the principle variables as possible, so as to achieve the effect of dimensionality reduction and elimination of linear correlation effects. (Berni, et al., 2011). The implementation process of Principal component analysis is shown in Table 3.1.

Principal Component Analysis(PCA)
1. Standardize the original index data.
2. Calculate the standardized matrix's correlation coefficient matrix.
3. Determine the correlation coefficient matrix's eigenvalues and eigenvectors.
4. Determine each primary component's variance contribution rate and cumulative variance contribution rate.
5. Choose the principal component.
6. Fit and transform data.
7. Return the principal component.

Table 3.1: procedure of PCA (Nobles, 2021)

3.4 Model Construction

This thesis uses RNN and LSTM networks to make regression models to predict stock prices. LSTM is a special variant of RNN, which is improved on the RNN model. Similar to the BP network, RNN model training has a process of forward propagation and back propagation.

In the forward propagation, the input samples are passed in from the input layer, and after being processed by the hidden layer, processed results turn to the output layer. If the processed results of the output layer does not match the expected output, then it goes to the error back propagation stage.

In the back propagation process, the output error is passed back through some methods, generally refers to the method of gradient descent which is transmitted back to the

input layer through the hidden layer by layer. Afterwards, the error is distributed to all the units of each layer, to obtain the unit error signal of each layer.

This error signal is used to modify the weight of each unit. This forward propagation and back propagation are carried out several times, and the weights are adjusted through the process until the error is reduced to an acceptable level or the number of iterations is completed.

3.4.1 RNN model

RNN has been used in speech recognition or text recognition for a long time due to its unique characteristics. But this thesis uses it in regression problems to predict stock prices. It can recognize that the time data entered earlier has a direct impact on the present time data because it analyses the timing of the input data. Multiple data may be supplied and the matching input and output layer can be configured according to the time point. The number of time nodes is the same as the number of hidden layers in the centre, and the number of neurons in each layer is the number of independent variables. In addition, the hidden layer in the centre will self-loop and provide recursive feedback. The training process adopts the BPTT algorithm. The excitation function in the BPTT algorithm generally adopts the sigmoid function.

3.4.1.1 Forward Propagation

As is shown in Figure 3.2, which is the expanding diagram of forward propagation. Forward propagation begins with a cell state h. The parameters are the bias vectors b and c along with the weight matrices U, V and W, respectively, for input-to-hidden, hidden-to-output and hidden-to-hidden connections. They are shared in the entire RNN network, which is very different from DNN. It is also because it is shared, it embodies the "loop feedback" idea of the RNN model.

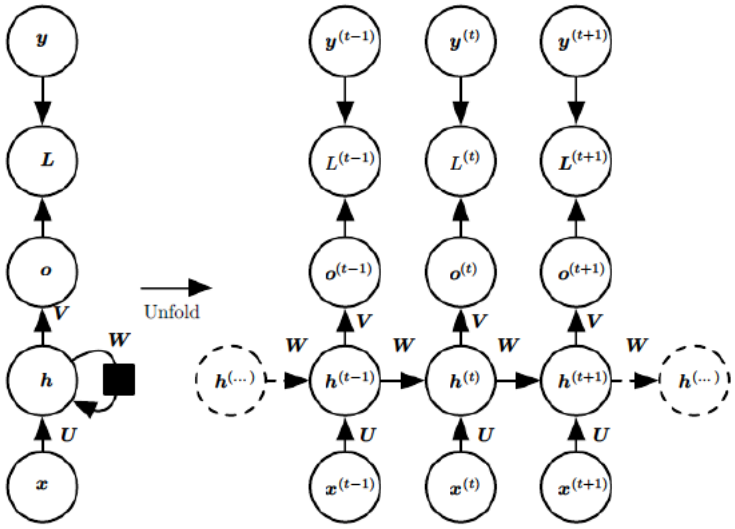


Figure 3.2: Forward pass process of RNN (Yu & Deng, 2014)

The forward propagation formula is presented as following:

$$h^t = f(a^t) = \tanh(Ux^t + Wh^{t-1} + b) \quad (3.2)$$

$$o^t = Vh^t + c \quad (3.3)$$

$$\hat{y} = softmax\ (o^t) \quad (3.4)$$

3.4.1.2 Backward Propagation

The unique structure of the RNN neural network may be readily seen after extending the RNN. Back propagation is accumulated from the last time, while forward propagation is computed in the order of time. The residuals are returned to the user, and the weights are adjusted. For end-to-end training, this type of neural network can employ backpropagation. Backpropagation Through Time is the term for this type of back propagation growth through time steps. The error propagates back along the red line, as seen in Figure 3.3.

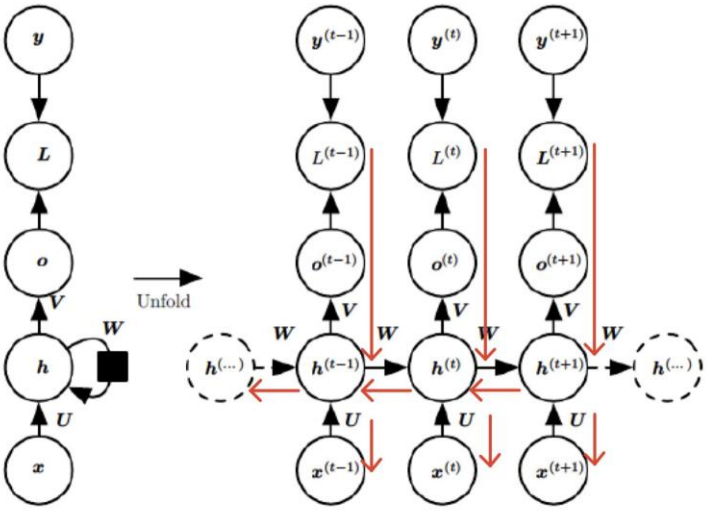


Figure 3.3: Gradient flow of back-propagation

The Backward propagation formula is presented as follows:

$$\nabla_c L = \sum_{t=1}^{\tau} \left(\frac{\partial o^t}{\partial c}\right)^T \nabla_{o^t} L = \sum_{t=1}^{\tau} \nabla_{o^t} L \quad (3.5)$$

$$\nabla_V L = \sum_{t=1}^{\tau} (\nabla_{O^T} L)\ (h^t)^T \quad (3.6)$$

$$\nabla_w L = \sum_{t=1}^{\tau} diag(1-(h^t)^2)\, \nabla_{h^t} L\ (h^{t-1})^T \quad (3.7)$$

$$\nabla_b L = \sum_{t=1}^{\tau} diag(1-(h^t)^2)\, \nabla_{h^t} L \quad (3.8)$$

$$\nabla_U L = \sum_{t=1}^{\tau} diag(1-(h^t)^2)\, \nabla_{h^t} L\ (x^t)^T \quad (3.9)$$

The RNN neural network's training procedure is as follows:

Training Procedure of RNN
1. To get the output vector, the input feature vector reaches the input layer neuron at time t.
2. The input vector of the hidden layer is made up of the output vector of the input layer and the output of the hidden layer at $t-1$.
3. Through the excitation function and threshold, the hidden layer's input vector at time t calculates the hidden layer's output vector.
4. Those output vectors are fed into the output layer, and the activation function is used to get the output layer's result.
5. Determine the difference between the output layer's result and the intended output.
6. Back-propagation based on the mistake, with each layer's weight updated

Table 3.2: Training Procedure of RNN

3.4.1.3 Long-term Dependency

Gradient disappearance and gradient explosion issues will develop as the number of recursive layers grows, which will have a significant impact on the model effect. The LSTM optimizes this from the structure of the neural unit. (Goodfellow , Bengio , & Aaron , 2016). As you can see on Figure 3.4, the gradient gradually disappears in the end.

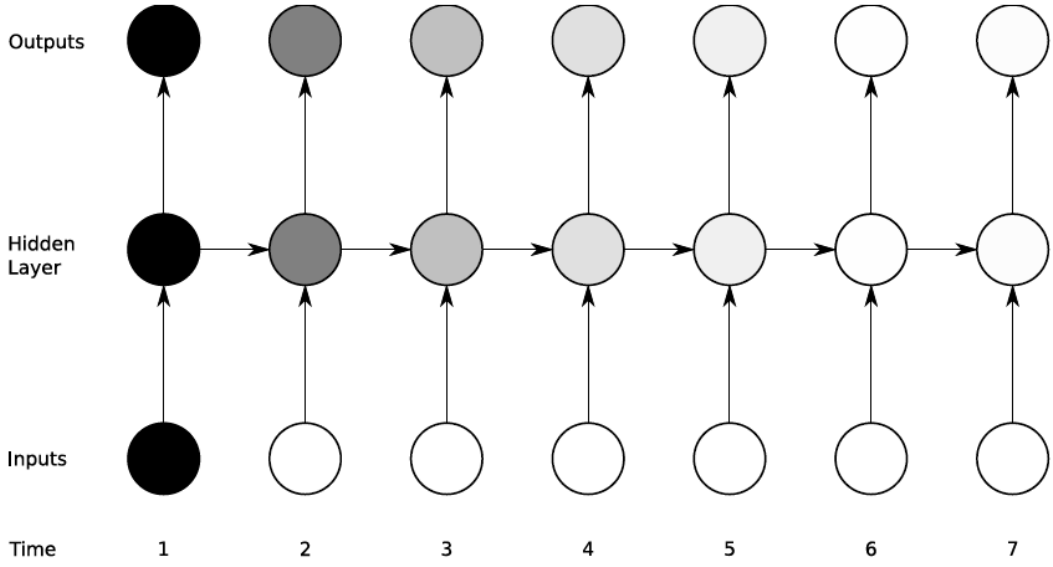


Figure 3.4 Vanishing gradient problem for RNNs. (Graves, 2012)

For the forward propagation, the state of h^t is represented as the equation 3.2:

$$h^t = tanh(Ux^t + Wh^{t-1} + b)$$

Since the amplitude of $\boldsymbol{h^t}$ of each level is limited to range(-1,1) by the tanh function, the forward propagation does not increase exponentially.

For the backward propagation, the partial derivative of equation 3.2 is calculated as follows:

$$Gradient = \frac{\partial h^{t+1}}{\partial h^t} = \ diag(1 - (h^t)^2)W \qquad (3.10)$$

Since the amplitude of h^t is limited to (-1,1) by the tanh function, the gradient reduces W to some certain extent. As the consequence, the larger h^t is, the smaller gradient is.

Eigenvalue of W after such reduction	The value of gradient
much smaller than 1 at every moment	Attenuate to 0 and gradient disappears.
much greater than 1 at every moment	increase exponentially and explosion
sometimes less than 1 and sometimes greater than 1	relatively stable.

Table 3.3: Reason of Long-term Dependency

3.4.2 LSTM model

The LSTM model utilizes a unique LSTM structure to replace the hidden layer neurons of the general RNN mode. Input gates, output gates, forget gates, and memory cells are all part of the LSTM cell. Logic units include input gates, output gates, and forget gates. They do not send their output to other neurons; instead, they are in charge of setting the weights at the edges of the neural network, where the other parts of and the memory unit are connected, in order to selectively memorize the correction parameters of the error function of the feedback with the gradient drop. (Graves, 2012).

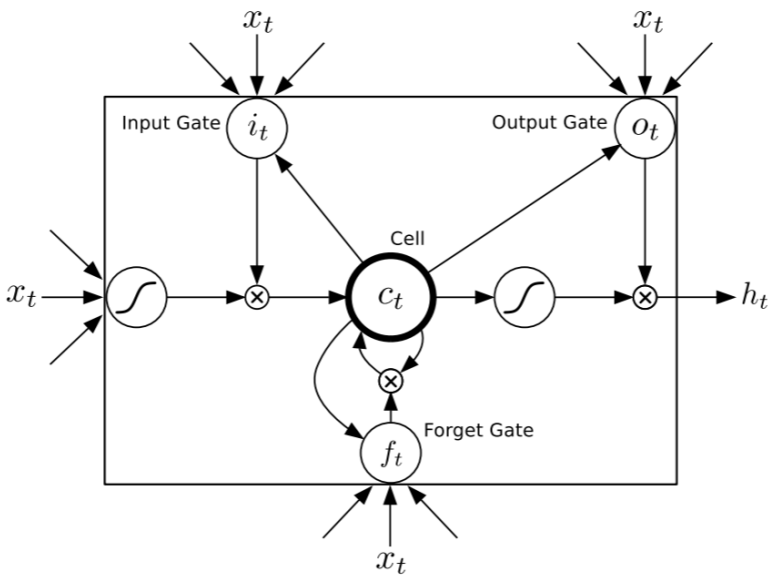


Figure 3.5 Cell structure of LSTM (Graves, 2012)

These gates operate as filters, each with its own function: What information is erased from the cell state is defined by the forget gate. • The input gate determines what data is added to the cell state. • The output gate defines which of the unit state's information

is output. (Fischer & Krauss, 2018)

3.4.2.1 Forward Propagation

The cell of LSTM replaces the hidden unit of the ordinary RNN, and the LSTM is an output of the cell. LSTM introduces a cell cycle to keep the gradient flowing continuously for a long time. One of the key points is: the weight of the cell cycle depends on the context, rather than being fixed. The specific method is to control the weight of the cell cycle through the gate, and the gate is determined by the context.

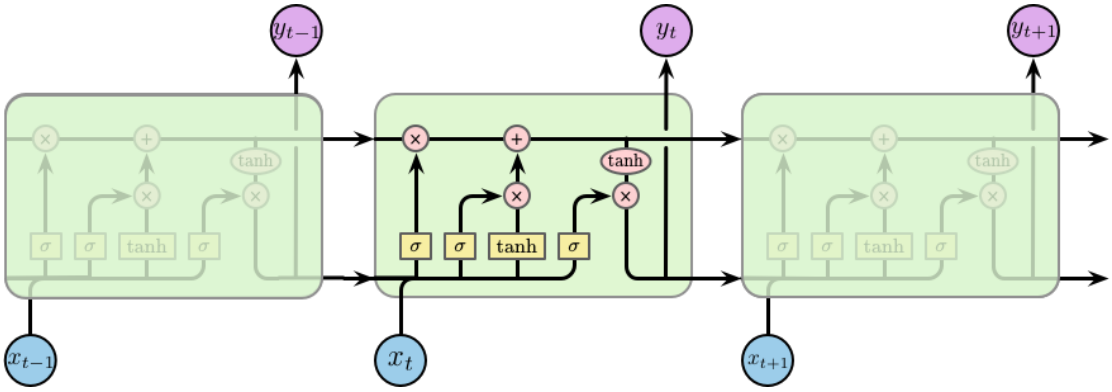


Figure 3.6: The repeating module in an LSTM contains four interacting layers. (colah, 2015)

The cell state, represented by the horizontal line at the top of the diagram, is crucial to LSTMs. The state of the cell resembles that of a conveyor belt in certain ways. There are only a few tiny linear interactions as it travels down the entire chain. It's quite easy for data to simply travel down it without being altered.

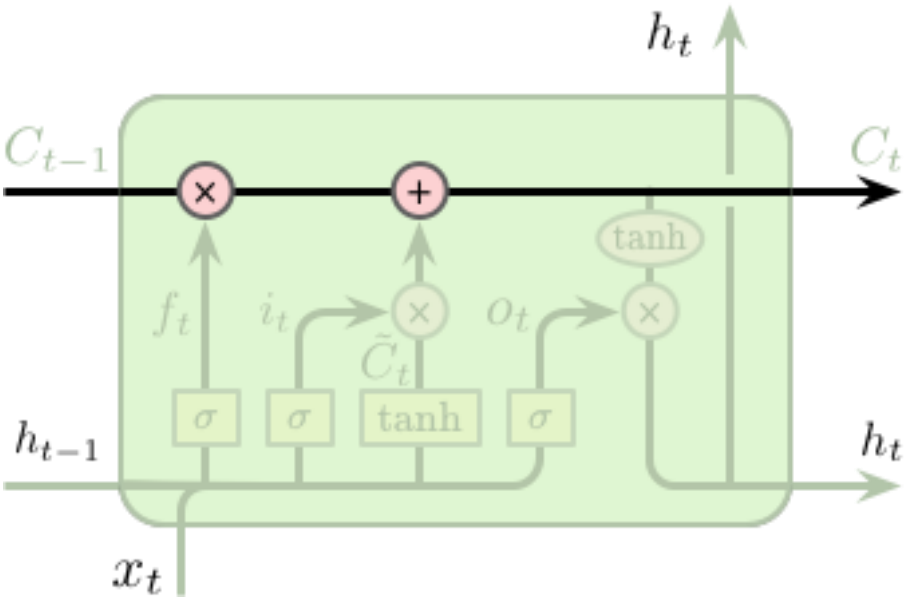


Figure 3.7 Cell (colah, 2015)

The action of the LSTM on the cell state (adding or removing information) is mostly accomplished via valves.

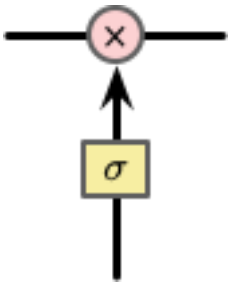


Figure 3.8: Gate (colah, 2015)

The sigmoid layer outputs a number between 0 and 1 describing how much each component should pass. 0 means that don't let anything pass, while 1 means that let everything pass.

Forget gate: What information in the cell state should be discarded and forgotten, choose to forget some information in the past.

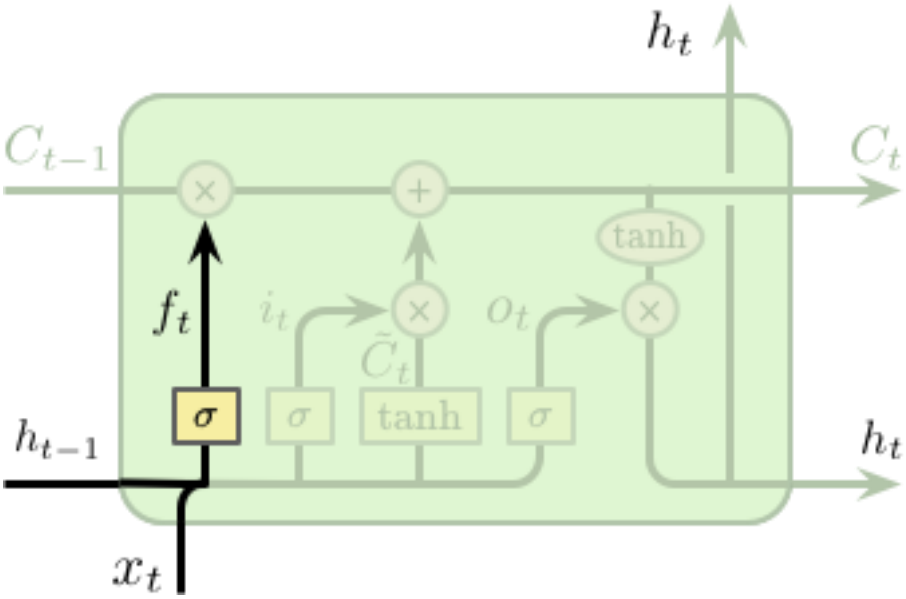


Figure 3.9: Forget gate (colah, 2015)

$$\vec{f}^t = \sigma\left(\vec{b}^f + \vec{U}^f x^t + \vec{W}^f h^{t-1}\right) \quad (3.11)$$

Input gate: Controls how much information in the input x^t enters the current state of the cell C^t.

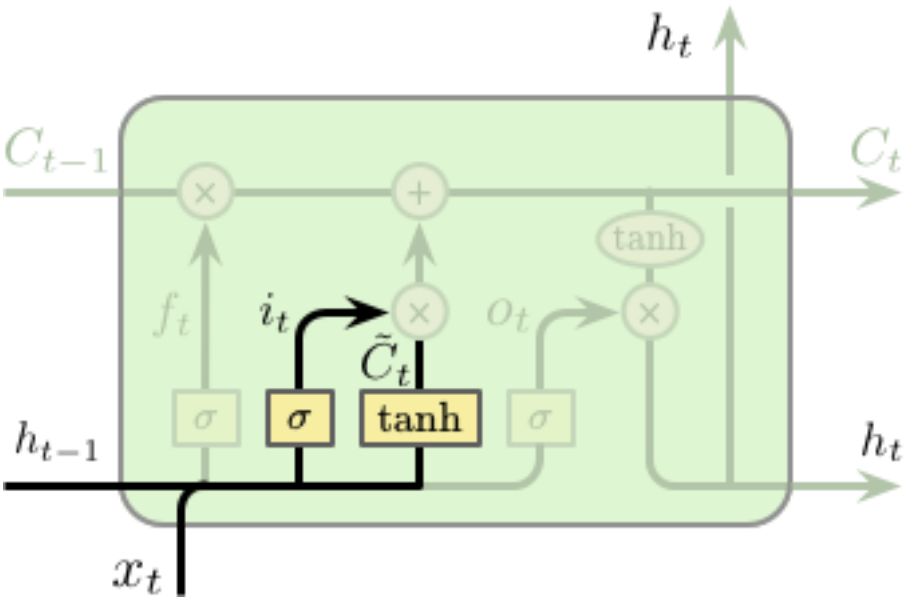


Figure 3.10: Input gate (colah, 2015)

$$\vec{\imath}^t = \sigma\left(\vec{b}^i + \vec{U}^i x^t + \vec{W}^i h^{t-1}\right) \tag{3.12}$$

Output gate: It controls how much information in the input C^t enters the cell and outputs h^t.

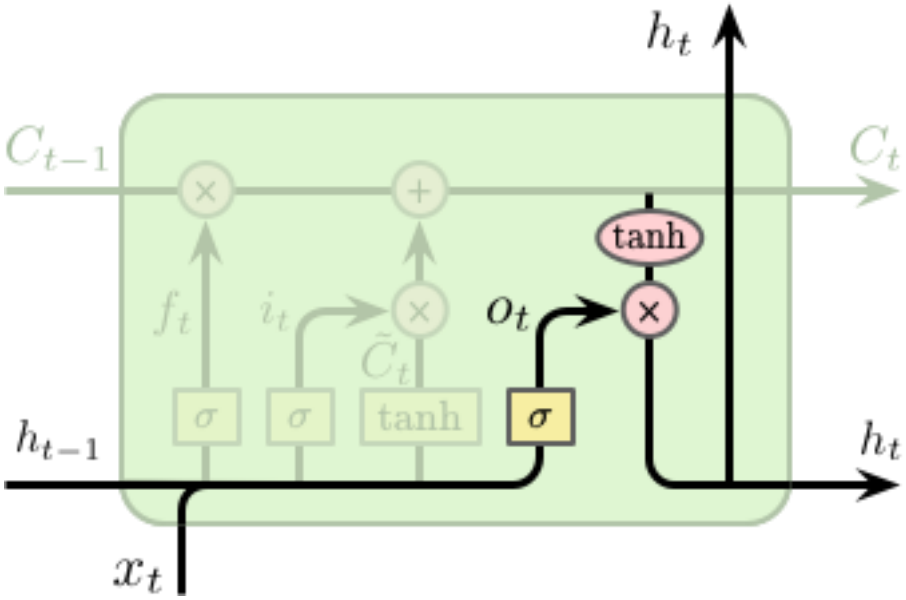


Figure 3.11: Output gate (colah, 2015)

$$\vec{q}^t = \sigma\left(\vec{b}^o + \vec{U}^o x^t + \vec{W}^o h^{t-1}\right) \tag{3.13}$$

Cell status: The status C^tconsists of two parts:

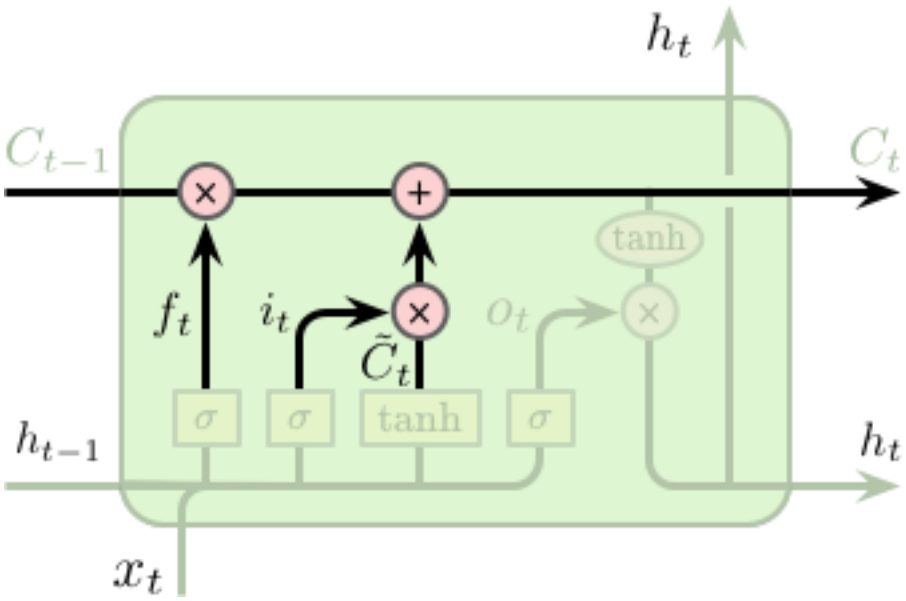


Figure 3.12: Cell state update (colah, 2015)

Part of it comes from the last state C^{t-1}, which has passed the control of the forget gate f^t, so that only part of the state enters the next time. Another part of it comes from the input. After i^t, only part of the input can enter the state update.

$$C^t = \vec{f}^t \odot \vec{C}^{t-1} + \vec{g}^t \odot \sigma\left(\vec{b} + \vec{U}\ x^t + \overrightarrow{W}\ h^{t-1}\right) \tag{3.14}$$

Cell's output update:

$$\vec{h}^t = \tanh\left(\vec{C}^t\right) \odot \vec{q}^t \tag{3.15}$$

Once the output $\vec{h}^t$ of the cell is obtained, the output $\vec{o}$ of the entire RNN unit is the same as the ordinary RNN.

3.4.2.2 Backward Propagation

The back propagation process of LSTM is very similar to the calculation process of RNN. The formula are as follows:

Forget gate:

$$\nabla_{\vec{b}^f} L = \sum_{t=1}^{\tau} \vec{f}^t \odot \left(1 - \vec{f}^t\right) \odot \nabla_{\vec{f}^t} L \tag{3.16}$$

$$\nabla_{\vec{U}_{i,:}^f} L = \sum_{t=1}^{\tau} \left(\nabla_{\vec{f}^t} L\right)_i\ \vec{f}^t \odot \left(1 - \vec{f}^t\right) \odot \vec{x}^t \tag{3.17}$$

$$\nabla_{\overrightarrow{W}_{i,:}\ f} L = \sum_{t=1}^{\tau} (\nabla_{\vec{f}^t} L)_i \ \ \vec{f}^t \odot \left(1 - \vec{f}^t\right) \odot \vec{h}^{t-1} \tag{3.18}$$

Where $(\nabla_{\vec{f}^t} L)_i$ represents the i-th component of $\nabla_{\vec{f}^t} L$.

Input gate:

$$\nabla_{\vec{b}^i} L = \sum_{t=1}^{\tau} \vec{i}^t \odot (1 - \vec{i}^t) \odot \nabla_{\vec{i}^t} L \tag{3.19}$$

$$\nabla_{\vec{U}_{i,:}\ i} L = \sum_{t=1}^{\tau} (\nabla_{\vec{i}^t} L)_i \ \ \vec{i}^t \odot (1 - \vec{i}^t) \odot \vec{x}^t \tag{3.20}$$

$$\nabla_{\overrightarrow{W}_{i,:}\ i} L = \sum_{t=1}^{\tau} (\nabla_{\vec{i}^t} L)_i \ \ \vec{i}^t \odot (1 - \vec{i}^t) \odot \vec{h}^{t-1} \tag{3.21}$$

Where $(\nabla_{\vec{i}^t} L)_i$ represents the i-th component of $\nabla_{\vec{i}^t} L$.

Output gate:

$$\nabla_{\vec{b}^o} L = \sum_{t=1}^{\tau} \vec{q}^t \odot (1 - \vec{q}^t) \odot \nabla_{\vec{q}^t} L \tag{3.22}$$

$$\nabla_{\vec{U}_{i,:}\ o} L = \sum_{t=1}^{\tau} (\nabla_{\vec{q}^t} L)_i \ \ \vec{q}^t \odot (1 - \vec{q}^t) \odot \vec{x}^t \tag{3.23}$$

$$\nabla_{\overrightarrow{W}_{i,:}\ o} L = \sum_{t=1}^{\tau} (\nabla_{\vec{q}^t} L)_i \ \ \vec{q}^t \odot (1 - \vec{q}^t) \odot \vec{h}^{t-1} \tag{3.24}$$

Where $(\nabla_{\vec{q}^t} L)_i$ represents the i-th component of $\nabla_{\vec{q}^t} L$.

Cell state:

$$\nabla_{\vec{b}}\ L = \sum_{t=1}^{\tau} \vec{i}^t \left(1 - \tanh^2\left(\vec{b}\ + U\vec{x}^{\ t} + W\vec{h}^{\ t-1}\right)\right)\ \odot \nabla_{\vec{C}^t} L \tag{3.25}$$

$$\nabla_{\vec{U}_{i,:}\ o} L = \sum_{t=1}^{\tau} (\nabla_{\vec{q}^t} L)_i \ \ \vec{q}^t \odot (1 - \vec{q}^t) \odot \vec{x}^t \tag{3.26}$$

$$\nabla_{\overrightarrow{W}_{i,:}}\ L = \sum_{t=1}^{\tau} (\nabla_{\vec{C}^t} L)_i \ \ \vec{i}^t \left(1 - \tanh^2\left(\vec{b}\ + U\vec{x}^{\ t} + W\vec{h}^{\ t-1}\right)\right)\ \ \odot \vec{h}^{t-1} \tag{3.27}$$

Where $(\nabla_{\vec{C}^t}L)_i$ represents the i-th component of $\nabla_{\vec{C}^t}L$.

Output gate:

$$\nabla_{\vec{C}}\ L = \sum_{t=1}^{\tau} \nabla_{\vec{s}^t}L \tag{3.28}$$

$$\nabla_{\vec{V}_{i,:}{}^o}L = \sum_{t=1}^{\tau} (\nabla_{\vec{s}^t}L)_i \odot \vec{h}^t \tag{3.29}$$

Where $(\nabla_{\vec{s}^t}L)_i$ represents the i-th component of $\nabla_{\vec{s}^t}L$.

Training Procedure of LSTM
1. Input the data features at time t into the input layer, and use the activation function to output the result.
2. Input the input layer's output result, the hidden layer's output at $t-1$, and the information stored in the Cell unit at $t-1$.
3. The output data to the next hidden layer or output layer is processed in the nodes of the LSTM structure by the Input Gate, Output Gate, Forget Gate, and Cell units.
4. The result is produced after the LSTM structure node's output is transmitted to the output layer neuron.
5. The error is transmitted back to the weights, which are then updated

Table 3.4: Training Procedure of LSTM

3.4.2.3 Mitigating Long-term dependency

when $t = \tau$, $\vec{h}^t$ is the last node and subsequent nodes are $\vec{s}^t$:

$$\nabla_{\vec{h}^t}L = \left(\frac{\partial \vec{s}^t}{\partial \vec{h}^t}\right)^T \nabla_{\vec{s}^t}L = V^T\nabla_{\vec{s}^t}L \tag{3.30}$$

when $t < \tau$, $\vec{h}^t$ is not the last node and subsequent nodes are $\vec{s}^t, \vec{f}^{t+1}, \vec{i}^{t+1}, \vec{q}^{t+1}, \vec{C}^{t+1}$:

$$\begin{aligned}\nabla_{\vec{h}^t}L = {} & V^T\nabla_{\vec{s}^t}L + \vec{f}^{t+1}\odot\left(1-\vec{f}^{t+1}\right)\odot(W^f)^T\vec{C}^t\odot\ \nabla_{\vec{C}^t}L \\ & + \vec{i}^{t+1}\odot(1-\vec{i}^{t+1})\odot\left(W^i\right)^T\tanh(\vec{b}\ + U\vec{x}^{\ t+1} + W\vec{h}^{\ t})\nabla_{\vec{C}^{t+1}}L \\ & + \vec{q}^{t+1}\odot(1-\vec{q}^{t+1})\odot(W^o)^T\tanh(\vec{C}^{t+1})\odot\ \nabla_{\vec{h}^{t+1}}L) + \vec{i}^{t+1}\odot(1 \\ & - \tanh^2\left(\vec{b}\ + U\vec{x}^{\ t+1} + W\vec{h}^{\ t}\right))(W\ )^T\nabla_{\vec{C}^{t+1}}L\end{aligned} \tag{3.31}$$

Since there is a constant part $V^T\nabla_{\vec{s}^t}L$, LSTM can alleviate the disappearance of the gradient. Due to the existence of various gates, the non-constant part in $\nabla_{\vec{h}^t}L$ will be

reduced, so the gradient explosion can be alleviated.

Since the RNN is converted into an ultra-long conventional neural network, the use of BP backpropagation may gradually reduce the error, but because the expansion is too long, the error must be attributed to each layer. This will cause the error to disappear after the gradient calculated by each neuron is halved. Due to the absence of gradient, the training weight change will only be somewhat updated. As a result, the entire training process will be unable to exclude the local best answer.

LSTM solves this problem. It is based on the modification of RNN, and each neuron in each layer is set with three gates, which are input gate, output gate and forget gate. According to the feedback weight correction number, it is possible to selectively forget and partially or totally accept, ensuring that no neuron is changed, the gradient is not lost several times, and the weights of preceding layers are also retrieved. Modifications to the error function, with the error function falling quicker with the gradient.

In the case of time series problems, any RNN may be represented, however LSTM will converge to the best solution faster and easier. RNN might not be able to find the best option. The LSTM provides a more flexible learning method for the RNN neural network's feedback error attribution, so that it does not immediately reach the local optimal solution while dropping with the gradient. (Mao, 2017)

3.5 Train Method of RNNS

The training procedure for RNN and LSTM neural networks is an important consideration. When it comes to addressing optimization issues, there are a variety of algorithms to select from, but gradient descent is the most popular. The optimization challenge is to determine which iterative approach is utilized to repeatedly improve the learning rate so that the network may get the greatest results in the shortest amount of time and avoid overfitting. As a result, selecting a suitable technique can improve the model's performance.

Set the loss function for sample i to $C_i(w)$ to get the total loss function:

$$C(w) = \sum_{t=1}^{n} \nabla C_{ij}(w) \tag{3.32}$$

The core of the optimization methods is to obtain w through iteration methods, so as to minimize the loss function.

3.5.1 SGD

The current Stochastic Gradient Descent (SGD) generally represents mini-batch gradient descent (Tseng, 1998). The formula is as follows:

$$w_{i+1} = w_i - \eta \sum_{t=1}^{m} C_{ij}(w) \tag{3.33}$$

Because the batch gradient descent technique updates each parameter using all training data, the training process will become unreasonably sluggish as the number of samples grows. To address the limitations of the batch gradient descent technique, the stochastic gradient descent method is suggested. Each sample is updated repeatedly using stochastic gradient descent (SGG).

3.5.2 RMSprop

The Root Mean Square Prop(RMSProp) technique employs the differential square weighted average for the gradient of the weight W and the bias b to further improve the problem of excessive swing amplitude in the update of the loss function and to speed up the convergence speed of the function. This method is useful for eliminating the direction of excessive swing amplitude and for correcting the swing amplitude so that it is less in each dimension. On the other side, it accelerates the convergence of network functions.

$$G_t = \gamma G_{t-1} + (1-\gamma)\nabla C(w)_t^2 \tag{3.34}$$

Use G to constrain the learning rate, and the update formula:

$$w_{t+1}^d = w_t^d - \frac{\eta}{C_{t+1}^d} \nabla C(w) \tag{3.35}$$

where d represents the dimension.

3.5.3 Adam

Adam (Adaptive Moment Estimation) algorithm is a combination of Momentum algorithm and RMSProp algorithm, which dynamically adjusts the learning rate of each parameter using first-order and second-order moment estimation of the gradient.

The major benefit of Adam is that, after bias correction, each repetition of the learning rate has a defined range, resulting in more stable parameters.

First-order momentum:

$$m_t = \alpha m_{t-1} + (1-\alpha)\nabla C(w) \tag{3.36}$$

Second-order momentum:

$$v_t = \beta m_{t-1} + (1-\beta)\nabla C(w)^2 \tag{3.37}$$

Because m and v are both set to 0 at the start, utilize the power of t to increase their size in the first few iterations:

$$\widehat{m} = \frac{m_i}{1-\alpha^t} \ , \hat{v} = \frac{v_i}{1-\beta^t} \tag{3.38}$$

Finally, the update formula is:

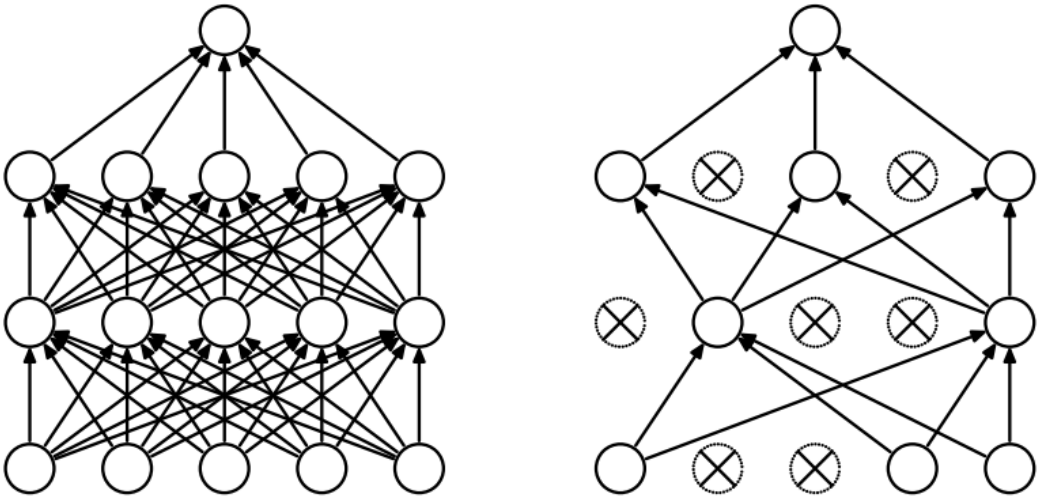
$$w_{t+1} = w_t - \eta \frac{\widehat{m}_t}{\sqrt{\hat{v}_i}} \tag{3.39}$$

3.6 Overfitting Problem

The number of parameters in a deep neural network grows at a rapid rate as the number of layers and neurons in each layer grows. Overfitting is an issue that arises when the number of parameters is high. The term "over-fitting" refers to a model that fits well on the training data but has a poor effect on the validation data, implying that the model's generalization ability is weak. Overfitting is a problem with deep neural network models. The following two approaches are now the most commonly utilized to tackle the problem of over-fitting.

3.6.1 Dropout

Hinton introduced the dropout approach in 2012 to prevent deep neural network models from overfitting. As demonstrated in Figure 3-1, dropout causes the model to "drop" hidden neurons at random with probability p each time it is trained (b). This training will not update the weights of these deleted neurons, but when the model is utilized, all neurons will be utilised.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.13: Dropout Neural Net Model. a) A standard neural net, with no dropout. b) Neural net with dropout applied. (Adrian, 2018)

3.6.2 Generation

Overfitting is a problem in deep learning that happens when there are too many features. As a result, we can lower the weight of traits or punish those that aren't significant. By adding an extra term to the cost function, the L2 regularization approach minimizes overfitting and enhances the model's generalization capabilities. (Shubham, 2020)

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2 \tag{3.40}$$

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} \; w \tag{3.41}$$

$$w = w - \eta \frac{\partial C_0}{\partial w} - \frac{\eta\lambda}{n} \quad w = \left(1 - \frac{\eta\lambda}{n}\right) w - \eta \frac{\partial C_0}{\partial w} \tag{3.42}$$

The L2 regularization term reduces the weight w, which implies that the network complexity is reduced and the data fit is optimizing. In conclusion, the LSTM neural network based on the RNN recurrent neural network efficiently considers the time of the input data and solves the problem of Long-term dependance in standard recurrent neural networks. However, owing to the intricacy of deep neural networks' structure, their training frequently necessitates the use of numerous optimization approaches to speed up the training process while also addressing common over-fitting issues.

3.6.3 Early Stop

The length of time it takes to train neural networks is one of the most significant problems. The model will not fit the training and test sets if there is insufficient training. If the model receives too much training, it will overfit the training data set and perform badly on the test data set. The compromise technique is to train on the training data set but stop when the validation data set performance starts to deteriorate. Early stopping is a simple, successful, and commonly utilized way of training neural networks. (Prechelt, 1998)

Procedure of Early Stop
1. The training dataset is first separated into the training dataset and the validation dataset, both of which must be trained.
2. At the end of each epoch (or every N epoch): Obtain the test results on the verification set and note the best verification set accuracy so far; if the test error on the verification set grows as the epoch advances, then cease training.
3. The network's final parameter is the weight of the greatest accuracy rate in the preceding processing of the test set.

Table 3.5: Procedure of Early Stop

3.7 Model Evaluation

After completing the model construction, we need to evaluate the effectiveness of the model. In this thesis, I constructed 4 evaluation indicators, namely MSE, RMSE, R2 score and Accuracy. MSE is the square of the difference between the true value and the predicted value, and then the sum average. RMSE is the square root of MSE.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{3.43}$$

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{3.44}$$

The coefficient of determination, or R^2 score, shows the amount of the dependent variable's fluctuation that can be explained by the independent variable through the regression connection. The R^2 score ranges from 0 to 1, and its value indicates the

relative degree of regression contribution, or the proportion of the total variance of the dependent variable Y that the regression relationship can explain. When the R^2 score is 1, the anticipated and true values in the sample are identical, implying that the independent variable in the regression analysis can better explain the dependent variable. The R^2 score is zero. At this point, the numerator equals the denominator, and each sample's projected value equals the mean. The prediction impact is shown to be quite low.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \quad \bar{y} \ )^2} \tag{3.45}$$

Accuracy represents the direct gap between the true value and the predicted value.

$$\text{Accuracy} = 1 - \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2} \tag{3.46}$$

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

The empirical experiment will be conducted in this chapter. Follow the procedure of chapter 3 and demonstrate the code.

4.2 Programming Environment

In the implementation chapter, the programming language used is python. The deep learning framework Tensorflow and Keras are used. The integrated environment uses jupyter notebook to program, test code and visualize. Many packages are imported to complete specific tasks.

Tools	Version
Python	3.6.13
Tensorflow	2.4.1
Keras	2.4.3
Numpy	1.17.0
Pandas	1.1.5
Matplotlib	3.1.1
Scikit-learn	0.19.0

Table 4.1: Version of tools

4.3 Data Collection and Description

Google's stock price from january 1, 2014 to December 31, 2019 are collected on the yahoo finance website. The data collected only includes various basic transaction data related to stock trading. They are: open price, High price, Low price, Close price, Adj close, Volume. These indicators are directly derived from the specific trading situation of the stock market on the day, and the calculation process does not include subjective factors, and can well represent the daily Price and volume information.The input of the neural network should be the data collected, and the price of the stock price is the output, So the output items should be the closing index.

Date	Open	High	Low	Close	Adj Close	Volume
2014-01-02	558.288269	559.434448	554.684692	557.117126	557.117126	3639157
2014-01-03	558.058044	559.024048	553.018005	553.053040	553.053040	3330067
2014-01-06	557.062073	559.989990	553.773804	559.219238	559.219238	3535061
2014-01-07	563.063049	570.415405	561.141113	570.000000	570.000000	5100095
2014-01-08	573.573547	574.234253	567.212219	571.186157	571.186157	4480515

Table 4.2: Top 5 records of GOOGL

Figure 4.1: Close price of GOOGL

	Open	High	Low	Close	Adj Close	Volume
count	1510.000000	1510.000000	1510.000000	1510.000000	1510.000000	1.510000e+03
mean	867.258911	874.421186	859.469559	867.176322	867.176322	1.997124e+06
std	245.749267	248.054443	243.899328	246.101995	246.101995	1.121032e+06
min	499.239990	500.279999	490.910004	497.059998	497.059998	5.206000e+05
25%	606.753021	609.039841	602.178436	606.509033	606.509033	1.346575e+06
50%	821.674988	825.345002	813.845001	820.160003	820.160003	1.679950e+06
75%	1090.052460	1103.847504	1078.585022	1090.942505	1090.942505	2.250775e+06
max	1364.000000	1367.050049	1353.000000	1362.469971	1362.469971	1.285810e+07

Table 4.3: History index description

In the historical transaction information data, the data of the valid samples of the six proxy indicators are all 1510, indicating that there are no missing values. The minimum close price is about 497USD, but the highest close price is 1362USD, indicating that

during this period, GOOGL stock has experienced a big rise and is a stock with good returns. The dataset do not include the information during weekends.

	Open	High	Low	Close	Adj Close	Volume
Close	0.999074	0.999539	0.999630	1.000000	1.000000	-0.263872

Table 4.4: Correlation of close price

In table 4.4, Open, High, Low, Close, Adj Close have a high correlation with each other, Over 99.9%. While correlation between Close and Volume is weak negative relationship, only about -0.2638. This may be that the relationship between Close and Volume are complicated and non-linear. They all represents trading signals. For example, While prices are dropping on higher volume, the negative trend is strengthening. The stock price has the ability to revert when it achieves a new high (or no low) owing to a reduction in volume.

4.4 Data Pre-processing

As it is shown in the Figure 4.2, the 1510 data we collected will be divided into training data, verification data and test data. The training data includes records from January 1, 2014 to December 31, 2018, the verification data includes records from January 1, 2019 to December 31, 2019, and the test data includes records from January 1, 2019 to December 31, 2019.

```
# Train Val Test Split
train_start = dt.date(2014, 1, 1)
train_end = dt.date(2018, 12, 31)
train_data_X = Train_series_X.loc[train_start:train_end]
train_data_Y = Train_series_Y.loc[train_start:train_end]

val_start = dt.date(2019, 1, 1)
val_end = dt.date(2019, 12, 31)
val_data_X = Train_series_X.loc[val_start:val_end]
val_data_Y = Train_series_Y.loc[val_start:val_end]
test_start = dt.date(2019, 1, 1)
test_end = dt.date(2019, 12, 31)
test_data_X = Train_series_X.loc[test_start:test_end]
test_data_Y = Train_series_Y.loc[test_start:test_end]
print(train_data_X.shape, val_data_X.shape, test_data_X.shape, train_data_Y.shape, val_data_Y.shape, test_data_Y.shape)
```

```
(1258, 2) (252, 2) (252, 2) (1258, 1) (252, 1) (252, 1)
```

Figure 4.2: Split dataset

4.4.1 Normalization

After data splitting, data normalizztion is performed. Call the MinMaxScaler function from sklearn.preprocessing to complete this operation.

```
scaler_X = MinMaxScaler()
train_X = scaler_X.fit_transform(train_data_X)
val_X = scaler_X.transform(val_data_X)
test_X = scaler_X.transform(test_data_X)
print(train_X.shape, val_X.shape, test_X.shape)
```

```
(1258, 6) (252, 6) (252, 6)
```

```
scaler_Y = MinMaxScaler()
train_Y = scaler_Y.fit_transform(train_data_Y)
val_Y = scaler_Y.transform(val_data_Y)
test_Y = scaler_Y.transform(test_data_Y)
print(train_Y.shape, val_Y.shape, test_Y.shape)
```

```
(1258, 1) (252, 1) (252, 1)
```

Figure 4.3: Normalization

4.4.2 Dimensionality Reduction by PCA

In this section, PCA's dimensionality reduction of 6 variables is completed and two comprehensive indexes are constructed. They are PCA1 and PCA2. Call the StandardScaler function from sklearn.preprocessing to standardize data. After standardization, call the PCA function from sklearn.decomposition to fit the data.

```
scalar = StandardScaler()
data = scalar.fit_transform(data)
pca = PCA() # instantiation
pca.fit(data)
print(pca.explained_variance_)
print(pca.explained_variance_ratio_)
print(pca.components_)
```

Figure 4.4: Data fitted by PCA

Since the contribution rate of the first feature and the second feature reached 99%, the data dimension after PCA dimensionality reduction is 2.As you can see in Figure 4.7, we use pca.fit_transfrom to do dimensionality reduction, and the final data set is 2 dimensions.

Variance	Varience_ratio
5.0856	0.8470
0.9165	0.1526
0.0012	0.0002
0.0004	0.00007
0.0001	0.00002
0.0000	0.0000

Table 4.5: Variance and contribution

Then PCA is executed and come out two comprehensive indexes

```
pca = PCA(n_components=2).fit(data)
new_data = pca.fit_transform(data)
```

```
print("orginal dataset shape:", data.shape)
print("after PCA dimensionality reduction:", new_data.shape)
```

```
orginal dataset shape: (1510, 6)
after PCA dimensionality reduction: (1510, 2)
```

Figure 4.5: Data after dimensionality reduction

After splicing with the original data, a new dataset is found.

Date	Open	High	Low	Close	Adj Close	Volume	PCA1	PCA2
2014-01-02	558.288269	559.434448	554.684692	557.117126	557.117126	3639157	-2.996552	1.049453
2014-01-03	558.058044	559.024048	553.018005	553.053040	553.053040	3330067	-2.976129	0.773870
2014-01-06	557.062073	559.989990	553.773804	559.219238	559.219238	3535061	-2.978648	0.958204
2014-01-07	563.063049	570.415405	561.141113	570.000000	570.000000	5100095	-3.095663	2.352180
2014-01-08	573.573547	574.234253	567.212219	571.186157	571.186157	4480515	-2.975991	1.810907
...	...	...	...	...	...	...	...	...
2019-12-24	1350.209961	1352.010010	1344.170044	1344.430054	1344.430054	673400	4.488255	-0.548886
2019-12-26	1346.550049	1363.199951	1345.510010	1362.469971	1362.469971	1183100	4.504307	-0.087043
2019-12-27	1364.000000	1367.050049	1353.000000	1354.640015	1354.640015	1160600	4.530889	-0.103320
2019-12-30	1356.810059	1357.000000	1337.839966	1339.709961	1339.709961	999700	4.439151	-0.261240
2019-12-31	1335.790039	1340.660034	1332.130005	1339.390015	1339.390015	975700	4.363655	-0.294381

1510 rows × 8 columns

Table 4.6: New variables

It can be shown in the Table 4.8 that the two newly constructed comprehensive variables PCA1 and PCA2 have a good correlation with the original variables. Therefore, it can be considered that the comprehensive index constructed in this article is more appropriate and can effectively express the information of the 6 variables.

```
results.corr()
```

	Open	High	Low	Close	Adj Close	Volume	PCA1	PCA2
Open	1.000000	0.999618	0.999460	0.999074	0.999074	-0.259560	9.976223e-01	6.365514e-02
High	0.999618	1.000000	0.999395	0.999539	0.999539	-0.253548	9.974127e-01	6.993149e-02
Low	0.999460	0.999395	1.000000	0.999630	0.999630	-0.270655	9.984968e-01	5.222824e-02
Close	0.999074	0.999539	0.999630	1.000000	1.000000	-0.263872	9.980940e-01	5.925452e-02
Adj Close	0.999074	0.999539	0.999630	1.000000	1.000000	-0.263872	9.980940e-01	5.925452e-02
Volume	-0.259560	-0.253548	-0.270655	-0.263872	-0.263872	1.000000	-3.206122e-01	9.472105e-01
PCA1	0.997622	0.997413	0.998497	0.998094	0.998094	-0.320612	1.000000e+00	3.326066e-17
PCA2	0.063655	0.069931	0.052228	0.059255	0.059255	0.947210	3.326066e-17	1.000000e+00

Table 4.7: correlation table

4.5 Model Building and Training

The regression model adapted deep learning framework Tensorflow, and Keras library. TensorFlow is the world's most widely used open source machine learning framework. It's quick, versatile, and well-suited to product-level, large-scale applications, allowing any developer or researcher to quickly apply artificial intelligence to a variety of problems.

To forecast the GOOGL daily closing index, the RNN and LSTM techniques are adopted, which transforms the time series analysis problem into a regression problem in supervised learning. The algorithm can learn how to predict the output mode from the input mode in the supervised learning issue, which consists of input mode (X) and output mode (Y). The problem becomes a regression problem based on one-dimensional features if the closing index x_t at the present time is used to forecast the closing index x_{t+1} at the future moment. The problem becomes a regression problem with two-dimensional features if it is based on the prior time x_{t-1} and the present time closing index x_t to forecast the future time closing index x_{t+1}, and so on.

Procedure of Model Construction
1. Convert the input data $x_1, x_2, \dots x_i$ into the characteristics of the closing index of x_{i+1}.
2. As much as feasible, feed the training set data into the set model in batches, and update the weights in the network structure as often as possible.
3. Reduce the difference between the fitted and actual values to learn about the nonlinear features of the input and output variables.
4. The test period's input variables are inserted into the trained model to produce the anticipated result.

Table 4.8: Procedure of Model Construction

4.5.1 Reshape Data

Before inputting data into the model, it is necessory to reshape the input data. Since the parameter of the input model is a three-dimensional variable (sample, timestep, variable_number). Call the reshape function from the numpy library to complete this work. It is shown in the Figure 4.6 below.

```
X_train = []
Y_train = []
X_val = []
Y_val = []

# Loop for training data
for i in range(timesteps,train_X.shape[0]):
    X_train.append(train_X[i-timesteps:i])
    Y_train.append(train_Y[i][0])
X_train,Y_train = np.array(X_train),np.array(Y_train)

# Loop for val data
for i in range(timesteps,val_X.shape[0]):
    X_val.append(val_X[i-timesteps:i])
    Y_val.append(val_Y[i][0])
X_val,Y_val = np.array(X_val),np.array(Y_val)
```

Figure 4.6: Reshape data

4.5.2 Model Building

This section uses Keras library to build LSTM network structure. Keras is a high-level neural network API framework that provides a model data structure for more efficient network layer organization. The Sequential model is Keras' primary model. The add function technique allows users to build a model by stacking the network layers they require.

```
# Adding Layers to the model
model = Sequential()
model.add(LSTM(X_train.shape[2],input_shape = (X_train.shape[1],X_train.shape[2]),return_sequences = True,activation = 'relu'))
Dropout(0.04)
for i in range(len(hl)-1):
    model.add(LSTM(hl[i],activation = 'relu',return_sequences = True))
    Dropout(0.04)

model.add(LSTM(hl[-1],activation = 'relu'))
model.add(Dense(1))
model.compile(optimizer = optimizers.Adam(lr = lr), loss = 'mean_squared_error')  #lr learning rate
#print(model.summary())

# Training the data
history = model.fit(X_train,Y_train,epochs = epochs,batch_size = batch,validation_data = (X_val, Y_val),verbose = 0,
                    shuffle = False, callbacks=callbacks_list)
model.reset_states()
return model, history.history['loss'], history.history['val_loss']
```

Figure 4.7: LSTM model construction

As illustrated in Figure 4.7, the network structure of the prediction model in this study is to add a layer of LSTM network to take input and a layer of fully connected Dense network for output, for time series issues such as stock index forecasting. The hidden lay setted is len(hl) layer of LSTM network with row(hl) unites. In this experiment, the activation function of Dense is set to Linear, the activation function of the hidden layer is set to relu, the loss function of the model is set to mean absolute error (hence referred to as MSE), and the optimization method is set to Adam. The prediction effect evaluation of the model adopts MSE, RMSE, R2, Accuracy, and Error. The model.compile function used to build the model and model.fit to fit training dataset.

Finally, return the model and the loss function of training dataset and validation dataset.

As shown in Figures 4.8, this thesis adopts Dropout and early stop to solve the problem of model overfitting.

```
# Setting up an early stop
earlystop = EarlyStopping(monitor='val_loss', min_delta=0.0001, patience=80,  verbose=1, mode='min')
callbacks_list = [earlystop]
```

Figure 4.8: Early stop

The Dropout setted is 0.04, and for an early stop, the monitor set is a loss function. the model will stop at loss function <=0.0001 and if this loss will not change during the next 80 epochs. The training process will stop.

4.6 Model evaluation

```
# Evaluating the model
def evaluate_model(sc, model, test_X, test_Y, timesteps):
    X_test = []
    Y_test = []

    # Loop for testing data
    for i in range(timesteps, test_X.shape[0]):
        X_test.append(test_X[i-timesteps:i])
        Y_test.append(test_Y[i][0])
    X_test, Y_test = np.array(X_test), np.array(Y_test)
    #print(X_test.shape, Y_test.shape)

    # Prediction Time !!!!
    Y_hat = model.predict(X_test)
    Y_hat= Y_hat.reshape(1, -1)
    Y_test= Y_test.reshape(1, -1)
    Y_test_true = sc.inverse_transform(Y_test)
    Y_hat_true = sc.inverse_transform(Y_hat)
    Y_test_true = Y_test_true[0]
    Y_hat_true = Y_hat_true[0]
    mse = mean_squared_error(Y_test_true, Y_hat_true)
    rmse = sqrt(mse)
    r = r2_score(Y_test_true, Y_hat_true)
    accuracy = 1 - np.sqrt(np.mean(np.square((Y_hat_true - Y_test_true) / Y_test_true)))
    accuracy=accuracy * 100
    return mse, rmse, r, Y_test_true, Y_hat_true, accuracy
```

Figure 4.9: Evaluation

As shown in Figure 4.9, in this thesis, model.predict is used to predict the function. Call the inverse_transformation to get correct prediction index and use the function in the math package as the calculation evaluation index. Restore the normalized value through inverse_transform. Finally, return to MSE, RMSE, R2_score, Accuacy, and test and predicted value.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, the results of the experiment will be represented and give a brief discussion of why the results are. The data uses the data from January 1, 2014 to December 31, 2018 for training, and predicts the close index of GOOGL from January 1, 2019 to December 31, 2019. MSE, RMSE, R2, Accuracy, and Error are calculated as the predictive effect evaluation index of the model. At the same time, use MinMaxScaler to reduce the complexity, applied Dropout, early stop to avoid over-fitting to improve the prediction accuracy of the model. Empirical analysis of different models and different input features on the impact of stock market forecast.

5.2 Influencing Model Parameters

Stock prediction modelling based on deep neural networks is to fit the non-linear mapping between input and output through sample learning, and then use the functional relationship between input and output to give new input, and the output obtained is the prediction result.

At this stage, 4 models are constructed as shown in Table 5.1. They are RNN-1, RNN-2, RNN-3, LSTM-1.

Model	Neural Network	Input Variable
RNN-1	RNN neural network	Close price
RNN-2	RNN neural network	6 variables
RNN-3	RNN neural network	Variables processed by PCA
LSTM-1	LSTM neural network	6 variables

Table 5.1: Model information

In the process of model construction and training, there are many parameters that affect the performance of the model. As shown in Table 5.2, there are about 7 of them. They are Timesteps, Hidden layer, Batch, Epochs, Learning rate, Early stopping, Dropout. These parameters are the values that have been adjusted to achieve good results for the model. The blanks are not set or not desired.

Name	Meaning	RNN-1	RNN-2	RNN-3	LSTM-1
Timesteps	Time steps	30	30	30	40
Hidden layer	The number of hidden layer	50	50	50	10
Node	Nodes in hidden layer	45	45	45	120
Batch	Batch size	32	32	32	64
Epochs	Number of iteration	100	250	100	500
Learning rate	Learning rate of model compiling	1e-3	1e-3	1e-3	1e-3
Early stopping	Epochs stopped		156		446
Dropout	Avoid overfitting				0.04

Table 5.2: Parameters Comparison

5.3 Model Performance

In this section, models are executed and get their results, and then compare the differences to further explore the reasons that affect the stock forecast.

5.3.1 RNN-1 Model

a) Model Training

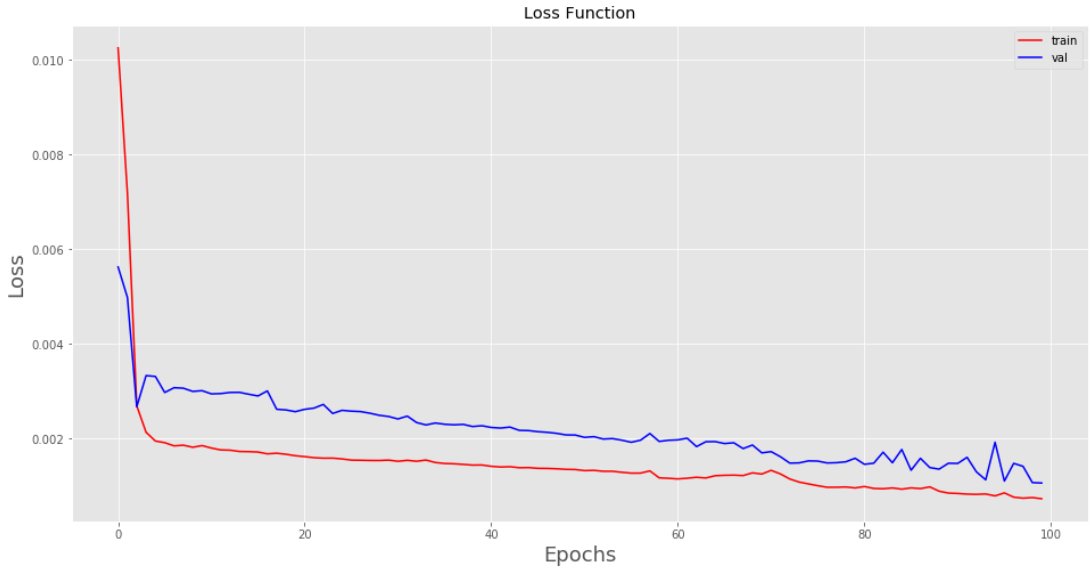


Figure 5.1: Loss function of RNN-1

Figure 5.1 shows the loss of the training dataset and the validation dataset after each iteration of the model. It can be observed that in the initial experiment of the RNN-1 model. The model's fitting effect and generalization level are considerably enhanced when all of the training data is repeatedly computed within 5 times. When epoch=20, the MSE value of the training set is 0.00017, the MSE value of the test set is 0.00023, indicating that the fitting effect of this model is relatively good. As the increment of epochs, the loss function of train and validation both decreases. The training error and validation error of the model have become stable, and the difference is small, indicating that the model's fitting effect and generalization level are relatively good at this time, and there is no under-fitting and overfitting.

b) Model Prediction

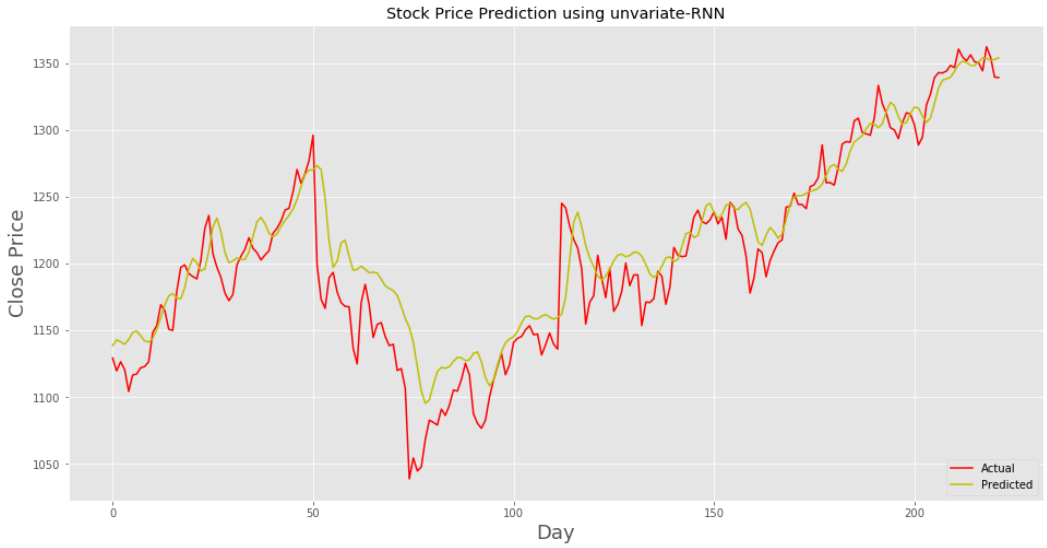


Figure 5.2: Prediction of RNN-1

Figure 5.3: Error Function of RNN-1

Figure 5.2 shows the prediction effect of the model, and Figure 5.3 shows the difference between the model's true value and the predicted value. It can be seen that the predicted value of 150-250 days during the test period has a small difference from the true value, and the prediction effect is better. From the 50th trading day to the 120th trading day of the test period, there are two large outliers and the error fluctuates between -100 and 75.

	MSE	RMSE	R^2 score	Accuracy
RNN-1	796.7150	28.2261	0.8585	97.5311

Table 5.3: Evaluation of RNN1

5.3.2 RNN-2 Model

a) Model Training

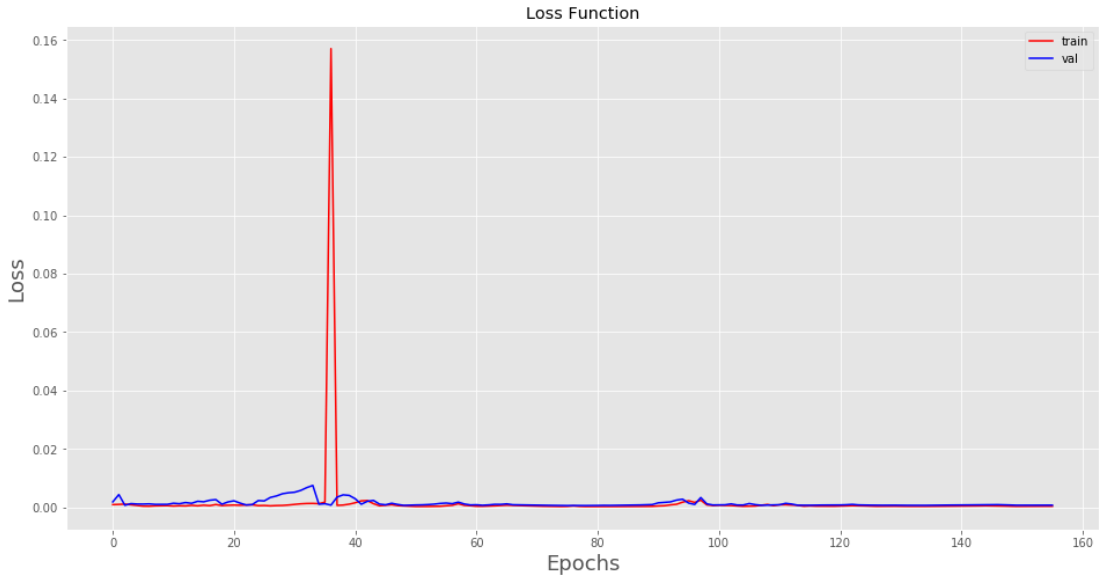


Figure 5.4: Loss function of RNN-2

Figure 5.4 shows that in the RNN-2 model. It seems that the model fits well from the beginning. But in fact, when epochs = 35, there is a divorce point loss=0.16 that increases the entire ordinate order of magnitude. It can be considered that after epochs = 37, the model has reached a relatively good generalization level. In the subsequent training process, due to early stop, it stopped at epochs = 156.

b) Model prediction

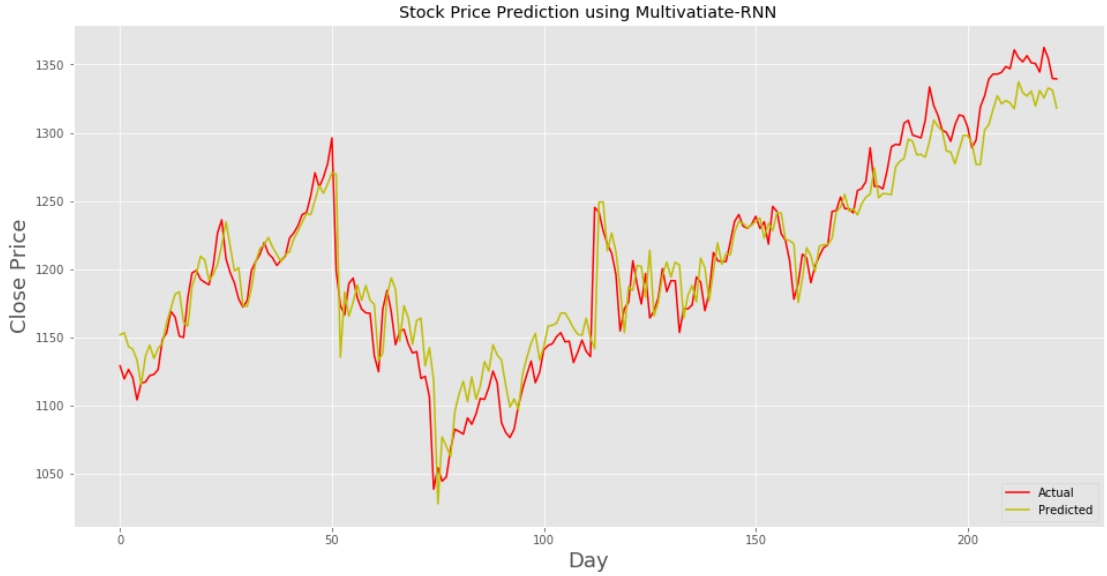


Figure 5.5: Prediction of RNN-2

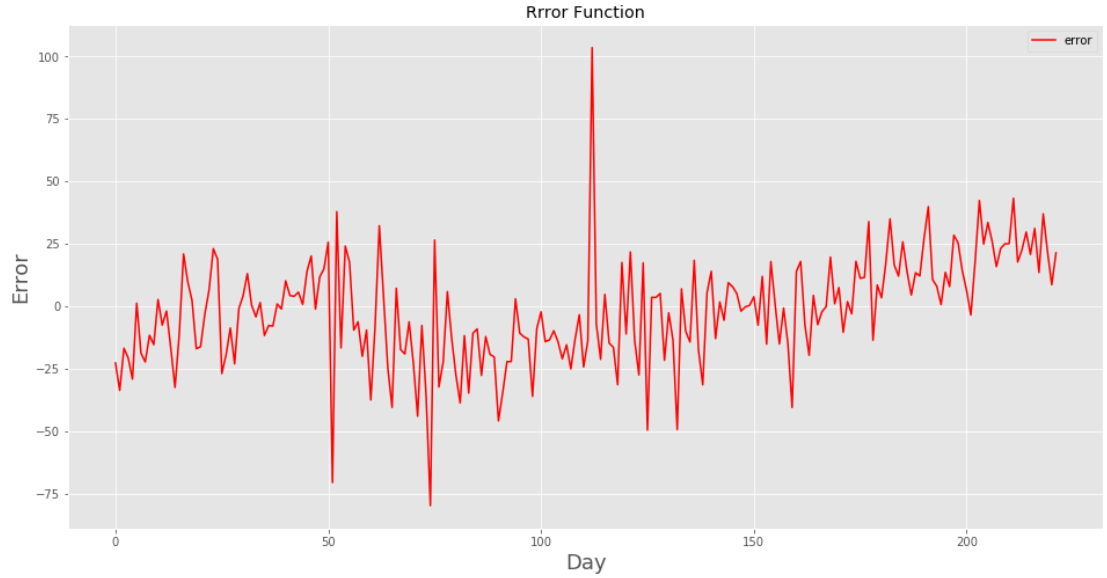


Figure 5.6: Error function of RNN-2

Figure 5.5 shows the prediction effect of the model, and Figure 5.6 shows the difference between the model's true value and the predicted value. It can be seen that the overall simulation effect is good, But the simulation did not work well after 150 days. There are also 2 divorce point greater than the average value for the error function.

	MSE	RMSE	R^2 score	Accuracy
RNN-2	469.38	21.6652	0.9166	98.1702

Figure 5.4: Evaluation of RNN-2

5.3.3 RNN-3 Model

a) Model training

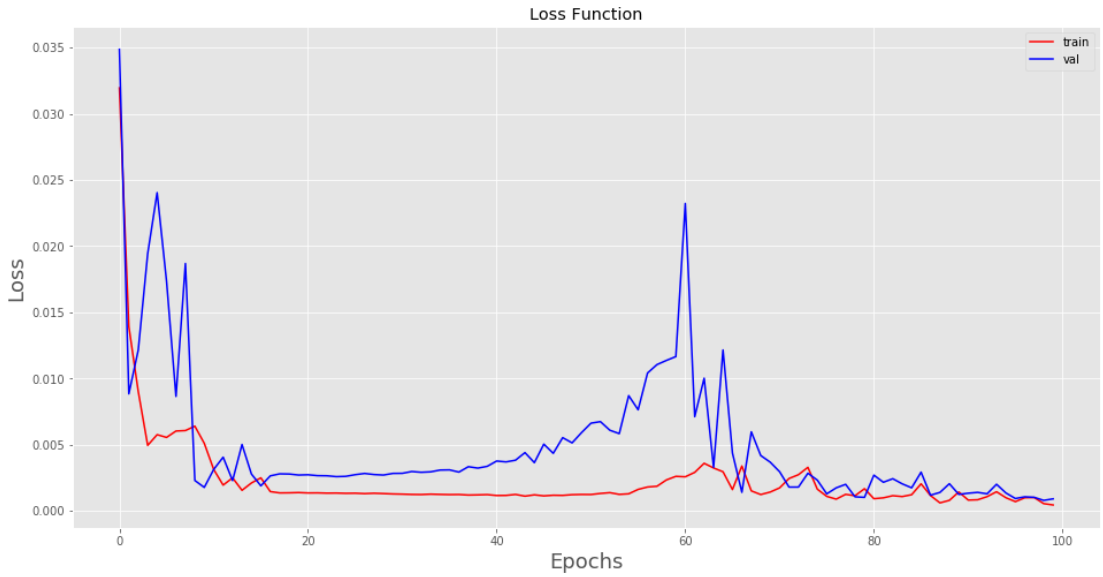


Figure 5.7: Loss function of RNN-3

Figure 5.7 shows that in the RNN-3 model, when epochs=17, the MSE value of the model in the training set and the validation set drops drastically with fluctuation, indicating that the model's fitting effect and generalization level are significantly improved at this time, but still remain at a relatively high point. At epochs=60, the training error of the model tend to be stable, but the validation error of the model experience an increase, then falling down. In the following training process, the fitting effect of the model and the generalization level are relatively good at this time, and there is no under-fitting and over-fitting.

b) Model prediction

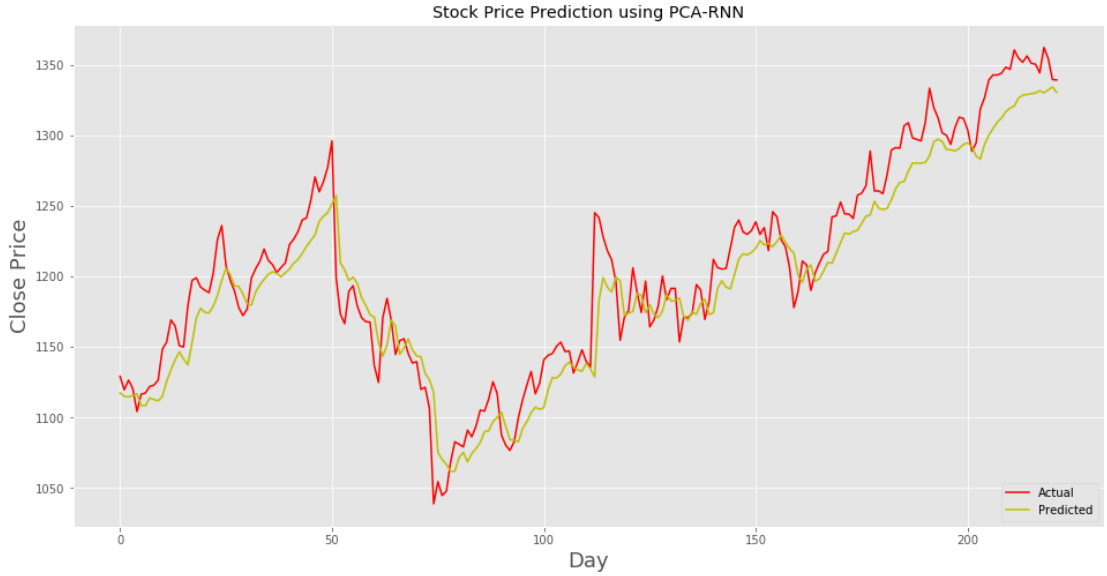


Figure 5.8: Prediction of RNN-3

Figure 5.9: Error function of RNN-3

Figure 5.8 shows the prediction effect of the model, and Figure 5.9 shows the difference between the model's true value and the predicted value. It can be seen that the overall simulation effect is not particularly good, and the overall error ranges -50 and 50. There are also 2 divorce points, one of them is smaller than -75 and the other is bigger than 100.

	MSE	RMSE	R^2 score	Accuracy
RNN-3	562.3328	23.7135	0.9001	98.0458

Table 5.5: Evaluation of RNN-3

5.3.4 LSTM-1 Model

a) Model training

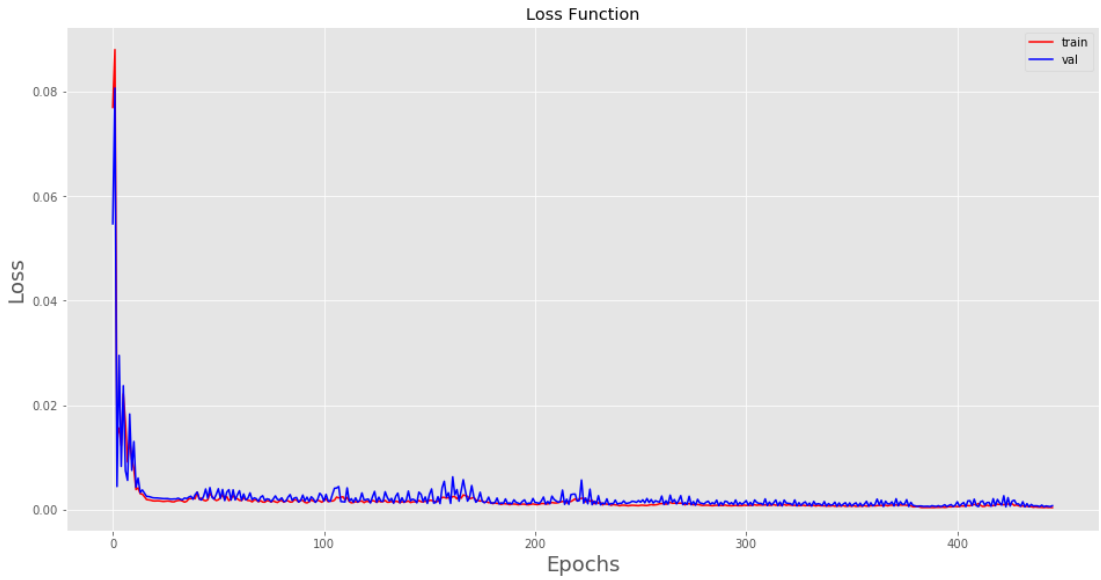


Figure 5.10: Loss function of LSTM-1

Figure 5.10 shows that in the LSTM-1 model, when epochs=20, the MSE value of the model in the training set and the test set drops drastically, indicating that the model's fitting effect and generalization level are significantly improved at this time. During the training process, training error and test error of the model tend to be stable, and the difference is small. The loss function of Validation dataset always fluctuates until the end of training process.

b) Model prediction

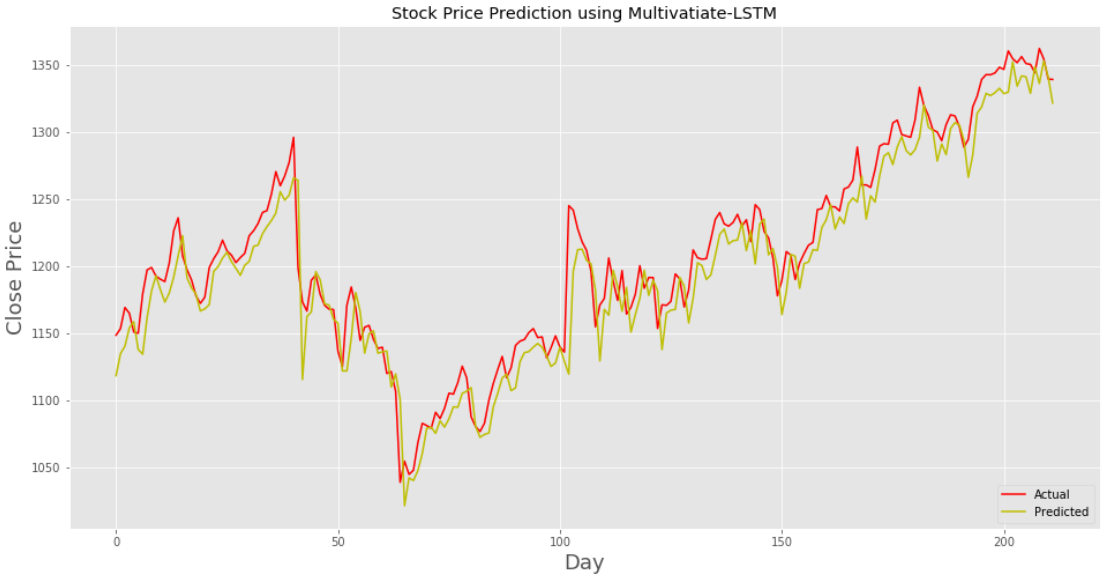


Figure 5.11: LSTM-1 model prediction diagram

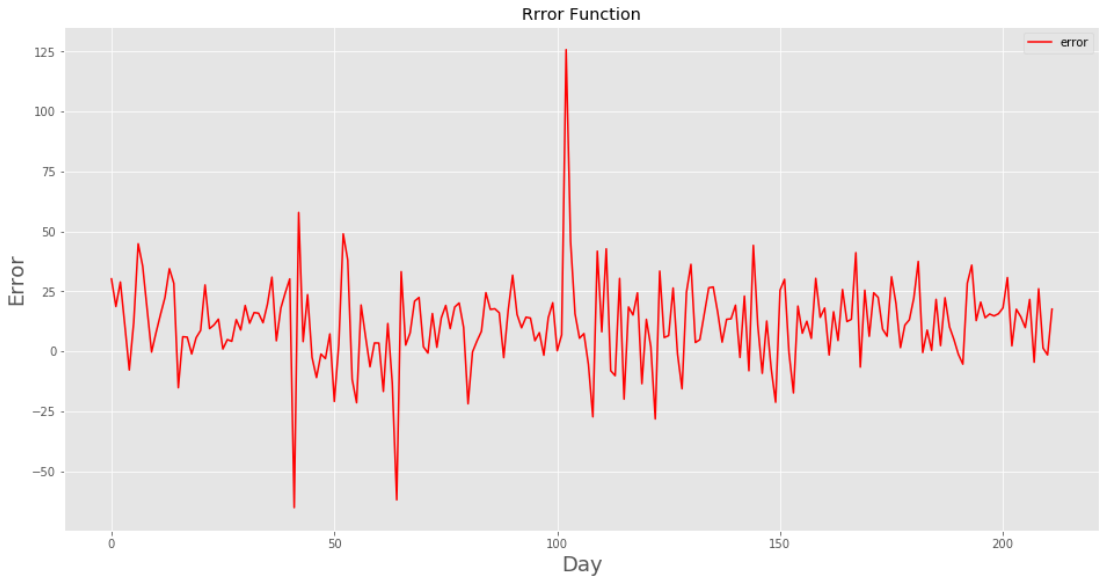


Figure 5.12: LSTM-1 Error function diagram

Figure 5.11 shows the prediction effect of the model, and Figure 5.12 shows the difference between the model's true value and the predicted value. It can be seen that the overall simulation effect is good. There are also 3 outliers greater than the average value for the error function. During the day 130 to day 200, the most part of error function ranges form -25 to 25, which indicate that its simulation works well.

	MSE	RMSE	R^2 score	Accuracy
RNN-1	467.2807	21.6166	0.9157	98.2071

Table 5.6: Evaluation of LSTM-1

5.4 Model Comparison

Model	RNN-1	RNN-2	RNN-3	LSTM-1
MSE	796.7150	469.38	562.3328	467.2807
RMSE	28.2261	21.6652	23.7135	21.6166
Accuracy	0.8585	0.9166	0.9001	0.9157
R^2 score	97.5311	98.1702	98.0458	98.2071

Table 5.7: Model performance

Overview, in table 5.7, It can be observed that RNN-2 and LSTM-1 perform better than Others. The RNN-3 model performs better than RNN-1 model.

RNN-1 model can be considered that the prediction effect of the model is poor and does not have practical significance. At the same time, to a certain extent, it shows that there are many factors affecting the fluctuation of stock prices, and the historical closing price of the GOOGL stock cannot fully reflect the relevant information of the stock market. Therefore, its performance is the worst.

By contrast, The RNN-2 which contains all 6 variables demonstrates better in all evaluation indexes compared with RNN-1 model. It is suggested that more data related to changes to stock prices should be collected, this will make our model work better.

For the model RNN-3, compared with RNN-2, it performs bad in 4 evaluation indexes. This might be that in the PCA dimensionality reduction process, a part of the amount of information may be lost, which will have a certain impact on the results of the model prediction. It is suggested that the original data which are processed by PCA method loses some information.

In this experiment, the LSTM-1 performs slightly better than the RNN-2 model, but this may be due to incorrect parameter selection. In the early stop, for the RNN model, I set 250 epochs and it stopped at 156. While for the LSTM model, I set 500 epochs and it stops at 446. It seems that LSTM needs more epochs and it is not easy to be over-fitted duo to bigger epochs. If I set the epochs for the RNN model and do not set the early stop. The RNN model will easily over-fit. Generally, their performance in predicting stock price is the same. However, in terms of the model's graph, the LSTM-1 model outperforms the RNN-2 model since it accurately depicts the stock price trend.

CHAPTER 6

CONCLUSION AND RECOMMENDATION

6.1 Overview

Changes in the stock market have a significant influence on the country's economic development in the current climate. Stock index projections play a critical role in national economic growth as well as scientific research. This thesis uses RNN model and LSTM model to predict the closing price of GOOGL. Comparing different input features and characteristics between models, four different models were constructed.

This thesis combs the relevant literature in the field of stock price forecasting in detail, starting from three aspects: the influencing factors of stock price, stock price forecasting methods and the actual combat of GOOGL forecasting closing price. Find out two directions worthy of improvement in the existing research. One is the limitations of the indicators selected for stock index price prediction, and the other is that new attempts are needed in the method of stock index price prediction.

Six indications, including the Open, Close, High, Low, Adj close, and Volume, were chosen as indicators based on current research, starting with the authenticity, comprehensiveness, and trustworthiness of the data. Collect and organize the required data, clean and process the data for abnormal situations such as missing values. Due to the high correlation between the indicators, the principal component analysis method is used for the original indicators.

Through experimental analysis, the use of a single variable close price to predict the stock price is less effective, because the stock price is affected by many factors, and the close price cannot reflect all the circumstances. However, in multivariate forecasting of stock prices, due to the high correlation between various indexes, most of the information can be retained after dimensionality reduction through PCA, but the prediction effect is not good. The reason may be that part of the information is confirmed in the process of dimensionality reduction, which makes it impossible to accurately reflect the situation of the stock market. The prediction effects of RNN-2

and LSTM-1 are relatively good, but the RNN model is more prone to overfitting and cannot be iterated multiple times. But the prediction effects of the two models are similar.

For stock price prediction, experiments presented a multivariate deep learning-based method. The multivariate ANN model(RNN-3) clearly exceed the best univariate ANN model(RNN-1). According to the findings, it's also found that multivariate models make better use of the input data and increase the stock prediction task's performance and efficiency. The methodology we proposed can only be solidified by comparing it to other stock prediction algorithms. In the future work, the comparison with other machine learning methods can be made and to see which one is more effective at predicting stock prices.

6.2 Limitation

There are some limitations of models designed in this thesis. First of all, the data input to the model is limited, and the variables are limited to only 6 trading indicators and the length of time is also very short. In the future work, if sufficient time and resources are given, comprehensive information collection can be carried out, so that the model takes into account various factors that affect stock prices.

The second restriction is that the models developed in this thesis cannot assist you in making money from short-term stock trading, such as daily trading. These models make predictions based on knowledge from prior lags. (Selvin, R, E.A, Vijay, & K.P, 2017) It is difficult to assist you in developing a short-term investing plan due to the huge number of factors impacting the stock price and the necessity to update the model's input variables on a timely basis. The architecture of models is mostly to blame. Only the models developed in the thesis can help investors make long-term stock market investments. The models can demonstrate the Long-term trend.

6.2 Future Work

To enhance the accuracy of the forecast, you can evaluate alternative models or change the model used in this article in future work. As previously stated, the model is not

appropriate for real-time analysis owing to structural issues. Real-time prediction can be accomplished by modifying existing models and integrating the model's structure with other technologies.

Only six transaction data are utilized as input variables in this article: Open, High price, Low and Close, Adj close, and Volume. However, numerous characteristics that represent price fluctuations, such as short-term price patterns, would be missing if those variables are used alone. As a result, researchers have not only studied these trading indications in recent studies. There will also be some technical indications included. Singh (2016) used OHLC basic data to create several indicators that may monitor the short-term trend of stock prices in addition to technical indicators. As a result, by altering the input indications, the future work may be better forecasted. As a result, it may be considered in future study to use trading indicators and technical indicators created from trading indicators as model input parameters.

REFERENCES

Adrian, G. (2018, June). *A review of Dropout as applied to RNNs - Adrian G*. Retrieved June, 2021, from Medium: https://adriangcoder.medium.com/a review-of-dropout-as-applied-to-rnns-72e79ecd5b7b

Ayodele, A. A., & Aderemi, A. O. (2014). Stock Price Prediction Using the ARIMA Model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*. doi:10.1109/uksim.2014.67

Berni, A., Giuliani, A., Tartaglia, F., Tromba, L., Sgueglia, M., Blasi, S., & Russo, G. (2011). Effect of vascular risk factors on increase in carotid and femoral intima media thickness. Identification of a risk scale. *Atherosclerosis, 216*(1), pp. 109 114.

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. *2015 IEEE International Conference on Big Data*.

colah. (2015, August). *Understanding LSTM Networks*. Retrieved from http://www.huaxiaozhuan.com/%E6%B7%B1%E5%BA%A6%E5%AD%A6%4%B9%A0/chapters/6_RNN.html

Dash, M., & Liu, H. (1997). Feature Selection for Classification. *Intelligent Data Analysis*.

Fama, E. F. (1995). Random Walks in Stock Market Prices. *Financial Analysts Journal*. doi:10.2469/faj.v51.n1.1861

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), pp. 654-669. doi:10.1016/j.ejor.2017.11.054

Goodfellow , I., Bengio , Y., & Aaron , C. (2016). *Deep Learning*. Retrieved June, 2021, from https://www.deeplearningbook.org/

Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks. *Studies in Computational Intelligence*.

Hiemstra, Y. (1995). Modeling structured nonlinear knowledge to predict stock market returns. *Evidence and Applications*.

Jane, o. A., & Stephen, P. H. (1989). Financial statement analysis and the prediction of stock returns. *Journal of Accounting and Economics*. doi:10.1016/0165 4101(89)90017-7

Kim, K. j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*, pp. 307-319. doi:10.1016/s0925-2312(03)00372-2

Kim, K. j., & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural. *Expert Systems with Applications*.

Kimoto, T., & Asakawa, K. (1990). Stock Market Prediction System with Modular Neural Networks. *1990 IJCNN International Joint Conference on Neural Networks*.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. doi:10.1145/3209978.3210006

Lawrence, R. (1997). Using Neural Networks to Forecast Stock Market Prices.

Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the. *Expert Systems with Applications*.

Mao, J. H. (2017, April). Research on Influencing Factors of Stock Market Time Series Prediction Accuracy Based on LSTM Deep Neural Network.

Mizuno,, H., Kosaka,, M., & Yajima, H. (2001). Application Of Neural Network To Technical.

Nobles. (2021, June). *Understanding Principle Component Analysis(PCA) step by step.* Retrieved from Medium: https://medium.com/analytics vidhya/understanding-principle-component-analysis-pca-step-by-step e7a4bb4031d9

Paul, Y. D., & Maria, K. H. (2007). Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*. doi:10.1109/cimca.2005.1631572

Phichhang, O., & Hengshan, W. (2009). Prediction of Stock Market Index Movement by Ten Data Mining Techniques. *Modern Applied Science, 3*(12). doi:10.5539/mas.v3n12p28

Prechelt, L. (1998). Early Stopping - But When? *Lecture Notes in Computer Science*. doi:10.1007/3-540-49430-8_3

RAY, B., & PHILIP, B. (1968). An Empirical Evaluation of Accounting Income Numbers. *Journal of Accounting Research*. doi:10.2307/2490232

Reza, A. G., Mahmood, Y., & Hassan, P. (2010, February). The Comparison of Methods Artificial Neural Network with Linear Regression Using Specific Variables for Prediction Stock Price in Tehran Stock Exchange. *International Journal of Computer Science and Information Security*.

Robert, W. H., & David, L. F. (1992). The prediction of stock returns using financial statement information. *Journal of Accounting and Economics*. doi:10.1016/0165 4101(92)90025-w

Schierholt, K. (1996). Stock Market Prediction Using Different Neural Network.

Selvin, S., R, V., E.A, G., Vijay, M. K., & K.P, S. (2017). Stock price prediction using LSTM, RNN AND CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*.

Shubham, J. (2020, May). *An Overview of Regularization Techniques in Deep Learning (with Python code)*. Retrieved June, 2021, from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning regularization-techniques/

Sun, R. Q. (2015, December). Research on trend prediction model of stock index price based on LSTM neural network.

Susan, T. (2021, February). *Why new investors bought stock during the COVID-19 pandemic*. Retrieved June, 2021, from Detroit Free Press: https://eu.freep.com/story/money/personal-finance/susantompor/ 2021/02/05/ how-invest-stock-market/4360276001/

Tsaih, R., Hsu, Y., & Lai, C. (1998). Forecasting S&P 500 stock index futures with a hybrid AI system. *Decision Support Systems*. doi:10.1016/s0167-9236(98)00028 1

Tseng, P. (1998). An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*.

Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Transactions of Neural Networks, 10*(988-99).

White, H. (1988). Economic prediction using neural networks: the case of IBM daily stock returns. *IEEE International Conference on Neural Networks*. doi:10.1109/icnn.1988.23959

Yang, H., Chan, L., & King, I. (2002). Support Vector Machine Regression for Volatile

Stock Market Prediction. *Lecture Notes in Computer Science*, p. 391.

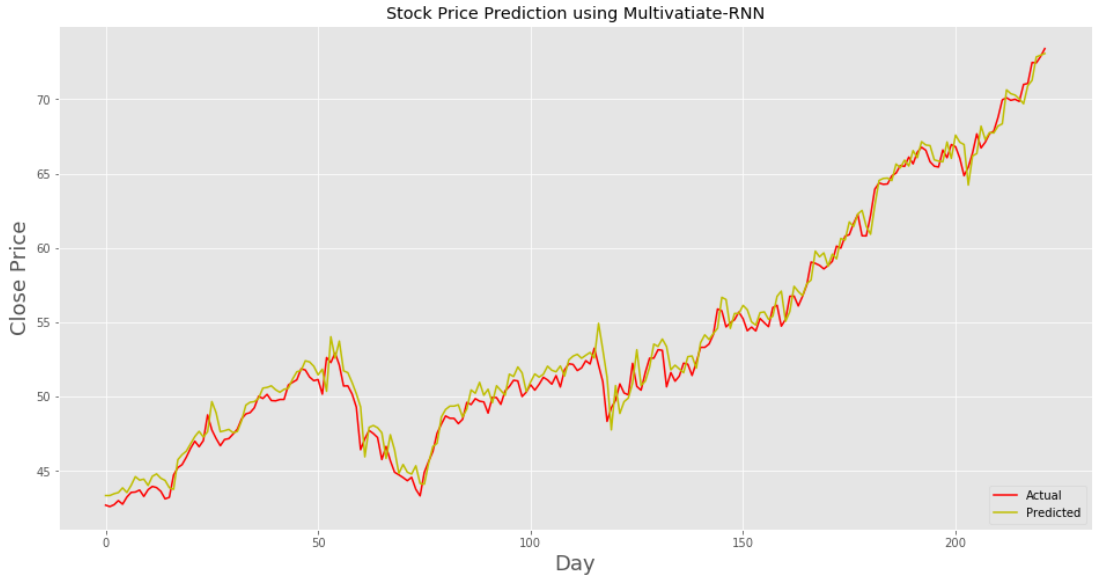
Yu, D., & Deng, L. (2014). Recurrent Neural Networks and Related Models. *Automatic Speech Recognition*, pp. 237-266.

Zhou, Y. (2019, April). Research on Price Forecast of Shanghai Composite Index Based on LSTM Model.

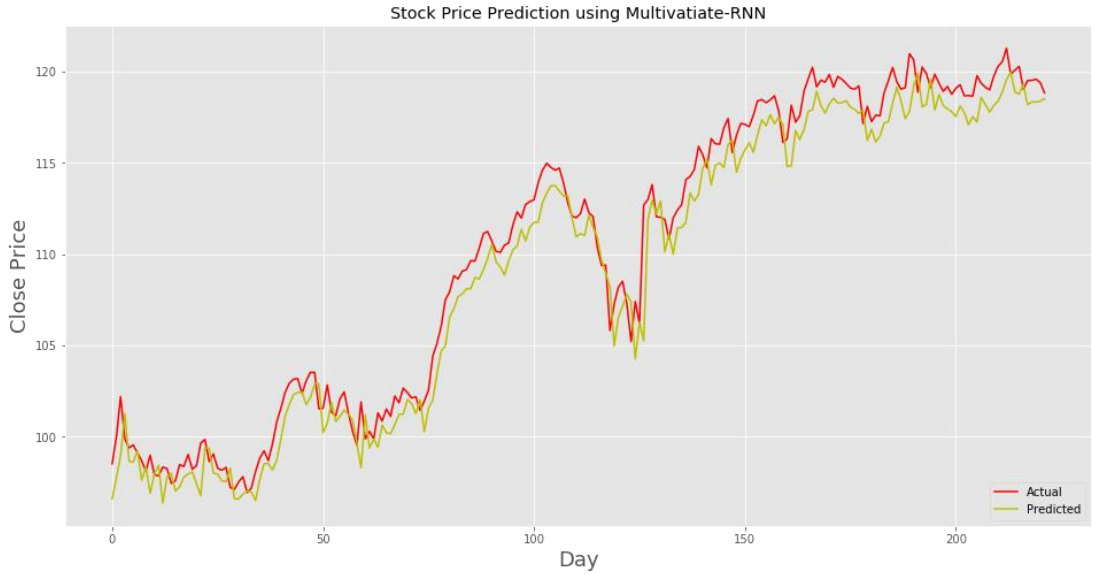
APPENDIX

Demonstrate more examples:

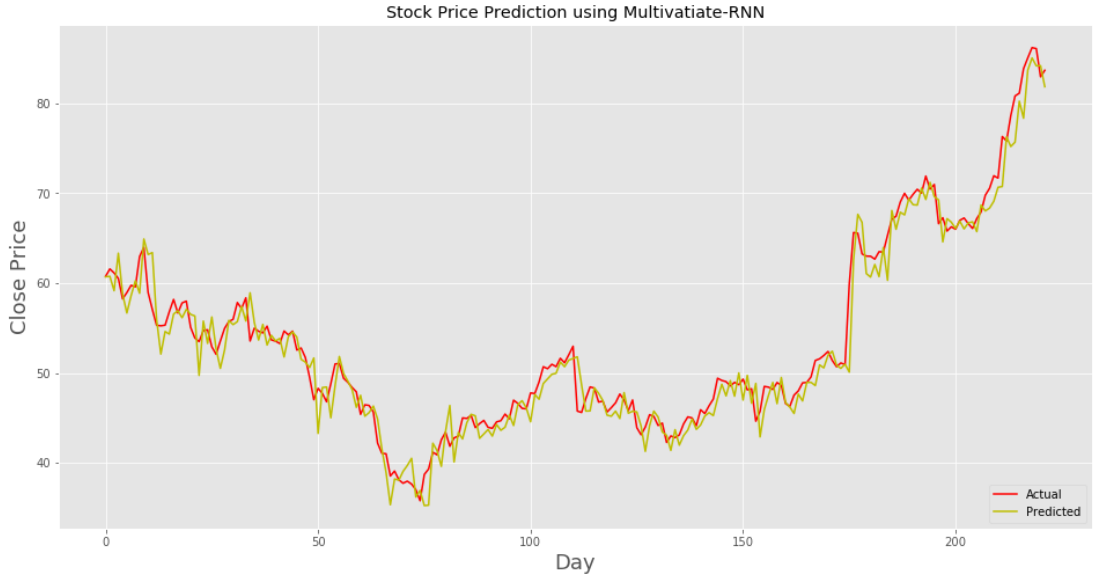
AAPL stock:



WMT stock:



TSLA stock:



MSFT stock: