HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION RESEARCH

Application of artificial intelligence in detecting and predicting the change of cash flow in the securities sector.

Nguyễn Ngọc Đăng
dang.nn200149@sis.hust.edu.vn

Major: Global ICT
Specialization: AI and BigData

Supervisor: Associate Professor Cao Tuan Dung

Department: Computer Engineering

School: Information and Communications Technology

HANOI, 03/2023

ACKNOWLEDGMENTS

During the process of researching and implementing the project, I have developed my self-study ability, cultivated many skills and learned many new technologies. Although difficulties are inevitable, thanks to the help of brothers and sisters, friends and teachers, I have finally completed this project. In particular, I would like to express my sincere thanks to Assoc. Prof.Cao Tuan Dung, Department of Computer Engineering, School of Information and Communication Technology, who directly guided, enthusiastically instructed and gave extremely useful advice throughout the process of making the project. Help me to complete the project in the best way and build the basis knowledge for the next projects.

ABSTRACT

Securities is a field that is not unfamiliar to each of us. Especially in the last ten years, this field has attracted a large number of investors around the world. Since then, a series of new technologies, algorithms, applications as well as support tools were born to provide maximum support for users. Artificial intelligence is a relatively new technology that has not been applied much in securities but has great potential. Seeing that, I chose the topic "Application of artificial intelligence in detecting and predicting the change of cash flow in the securities sector" with the desire to develop and apply AI technology as a facilitator for investors.

The topic gives me the opportunity to learn and self-study about AI, and also helps me gain more understanding of technologies when building mobile apps and websites for users.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Chapter 1 will give practical issues leading to the reason for choosing the topic, an overview of the stock index prediction system. Then give the goal and scope of the project along with the solution orientation and presentation layout of the project.

1.1 Motivation

Making accurate predictions about the market in general as well as stocks in particular is always the goal of experts as well as investors. The stock market is always volatile day by day and extremely unpredictable with many influencing factors such as GDP, CPI, Inflationary,... Therefore, to serve decision making, there are many mathematical formulas. , strategies, charts and graphs have been built such as RSI, Top-down method, Bottom-up strategy, scalping strategy,.. However, a fairly new application is Intelligence technology. Recently, artificial intelligence has been studied and partially applied to securities applications in parallel with traditional support tools and achieved certain effects.

In the Vietnamese market, AI applications are still too new to investors, partly limited by the volatile Vietnamese market and not too many studies on this application. Therefore, I chose the topic "Application of artificial intelligence in detecting and predicting the change of cash flow in the securities sector" to build an AI system to predict stocks, measure and evaluate the effectiveness of the model to support for future investors.

1.2 Objectives and scope of the graduation thesis

Based on the information I learned, I found that traditional support methods such as Moving Average indicators, McClellan Oscillator, .. do not help users automate making predictions and next trends. across a wide range of stocks as well as large number of biases. I want to apply AI technology to learn on large amount of data and make the user's prediction operation happen automatically and most conveniently.

I will rebuild a browser-based charting platform to display the price history of each stock, traditional indicators to assist users, and apply AI indicators to display live predictions stock parameters on the screen graph.

In this project, I will only apply the prediction range on the stock indexes for each day

such as opening price, closing price, volume, cash flow, etc.

1.3 Tentative solution

In the field of machine learning, there have been many models and methods to predict stock indices such as Linear Regression, Support Vector Machine, Ensemble Learning, etc. However, in order to give results with low expectation error. As much as possible and can improve based on the size of the data, I choose a deep learning algorithm Recurrent Neural Network, an extremely important algorithm specializing in string information processing. Specifically, a variant of RNN is LSTM, an All-To-One model that makes predictions based on a combination of long- and short-term data.

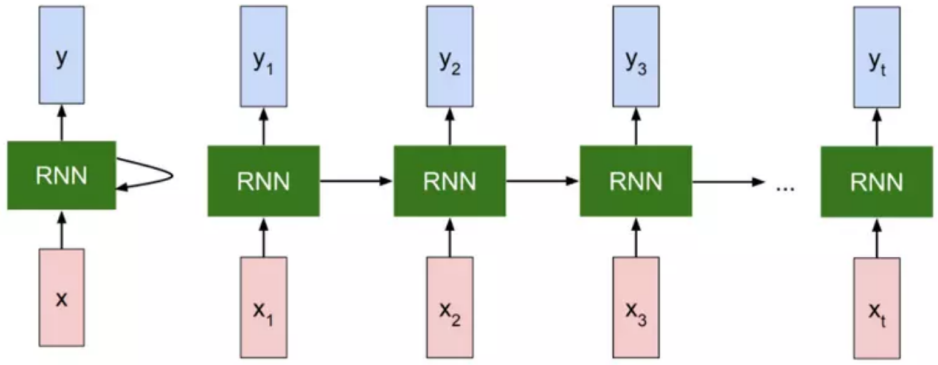


Figure 1.1 General RNN structure

As for building a website, I will build a candlestick chart interface similar to the Trading-View interface - the most popular user support platform today. Use some formulas in the stock field to design traditional indicators, apply library chartjs, ReactJS, HTML, CSS and Flask.

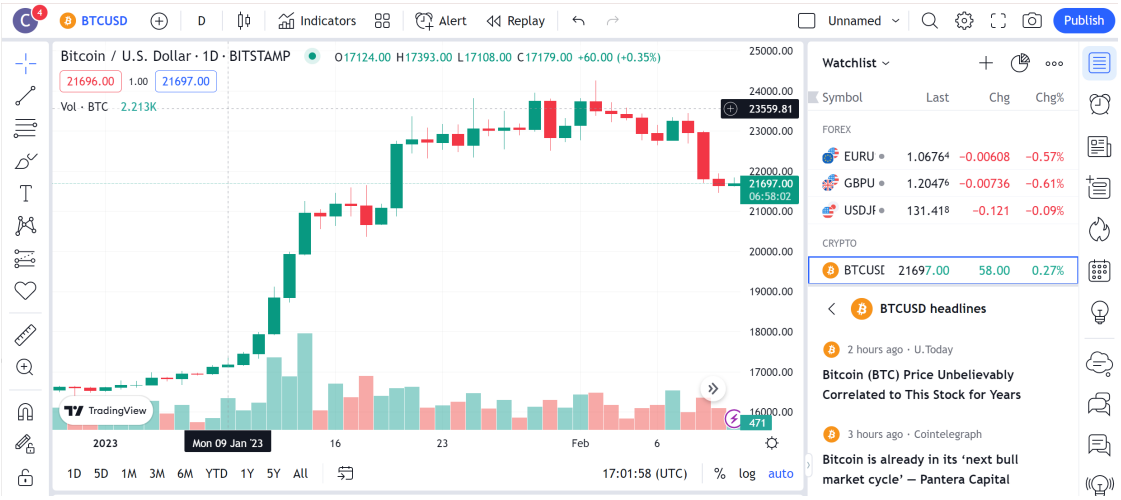


Figure 1.2 Trading View interface

1.4 Thesis organization

The remainder of this graduation project report is organized as follows:

In chapter 2, you will build a theoretical basis and introduce the technologies used.

With the above requirements, in chapter 3 you will analyze website design.

In Chapter 4, I present the implementation and the results obtained.
In chapter 5, you present about the evaluation of results, comparison with available applications.

CHAPTER 2. Theoretical foundations and applied technology

In Chapter 1, I gave an overview of the topic. And in chapter 2, I will present the survey and analysis of market demand, then conduct detailed analysis and design requirements of the website, tools to support the display and prediction of market data.

2.1 Status survey

During the project, I did 2 surveys. Firstly, about the current support tools, most of them do not have AI Stock Trading Bots. These include Trading View, StockRover, .. Besides, some tools have been at the forefront of testing and applying AI in market analysis and achieved certain success. Topping our list of best AI stock trading bots is Trade Ideas, the global market leader in providing AI and machine learning strategies for financial instruments, announced that it has been named as Best Machine Learning Development by Fund Technology and WSL in their annual 2019 Awards. Trade Ideas' HOLLY is an artificial intelligence tool that generates active strategies each trading day optimized and tested to capture alpha. Over the last year HOLLY's performance was 94.1% after commissions in Risk-On mode. The portfolio's gross return, before commissions and fees, measured 111%. These results compare to the S&P 500, measured by the $SPY index, which fell 4% over the same period. Risk-on performance reflects a bias towards staying in the positions beyond reward targets while strictly to risk targets for each trade. Unlike risk-off mode, where reward actions do not deviate from trade plan parameters and risk management rules permit exits earlier if existing profits erode significant, risk-on mode carries all trades the entire day until the close. This mode is often present during days of momentum which typify the majority of trading days in 2018. Note that Holly's risk-off performance for CY 2018 was 32%. In addition, some other prominent AI chatbots on the market such as TrendSpider, Tickeron,...

The second survey of algorithms, AI models achieved the highest efficiency. The 2022 article on "Machine Learning Approaches in Stock Price Prediction" released by the UK Institute of Physics (IOP) reviewed several studies that focused on various techniques for stock prediction.Even though these traditional algorithms are not

certainly flawed, they have been found to be relatively more accurate, especially when working with large datasets, and even more so when integrated into hybrid models. This combination of various ML algorithms can enhance their potential as some perform better at handling historical data, while others excel at processing sentiment data. However, these algorithms can also be highly sensitive to outliers and may not be able to effectively identify anomalies and exceptional cases. Researchers have evaluated several machine learning techniques and algorithms, including:

- Random Forest: This algorithm is particularly effective at achieving high accuracy with large datasets and is commonly used in stock prediction for regression analysis, which involves identifying relationships among multiple variables.
- Naive Bayesian Classifier: A simple yet efficient option for analyzing smaller financial datasets and determining the likelihood of one event impacting another.
- Support Vector Machine: An algorithm that uses supervised learning, which is trained by providing actual examples of inputs and outputs. It is highly accurate with large datasets but may struggle with complex and dynamic scenarios.
- K-nearest Neighbor: This algorithm uses a computationally expensive, distance-based approach to predict the outcome of an event based on the records of the most similar historical situations, referred to as “neighbors.”
- ARIMA: A time series technique that excels at forecasting short-term stock price fluctuations based on past trends such as seasonality but may not perform well with non-linear data and making accurate long-term stock predictions.

In additions, Deep learning (DL) can be viewed as an advanced version of machine learning, as it employs complex sets of specialized algorithms called artificial neural networks (ANN) to replicate the functioning of the human brain, resulting in a higher level of analysis and understanding compared to traditional ML systems. In particular, I am interested in the potential of using deep learning algorithms for stock prediction, with a particular focus on the highest performing algorithm, long-term short-term memory (LSTM).

Category	Algorithm	Metrics Used	
Traditional Machine Learning	Partial Least Squares Classifier (PLS Classifier) [15]	Average Error Value = 0.81225	
	Sequential Minimal Optimization (SMO) [15]	Average Error Value = 0.79656	
	Sentiment Analysis of Tweets using SVM and Extremely Randomized trees (ExtRa) [23]	R2 similarity and RMSE of the Ensemble model with respect to the SVM regressor. The model best performed for a 5-month dataset with difference in R^2 values given as:	
		SVM Meta-Regressor: +0.003	ExtRa tree Meta Regressor: -0.02
Deep Learning	LSTM coupled with sentiment analysis [22]	Matthews Correlation Coefficient (MCC) = 0.04092	Accuracy = 52.27%
	Attention-based LSTM with sentiment analysis [22]	Matthews Correlation Coefficient (MCC) = 0.04780	Accuracy = 54.58%
	LSTM [24]	RMSE = 0.0091, MAE=63 , MAPE=2.23%	
	CNN [24]	RMSE = 0.0087, MAE=61, MAPE=2.16%	
	Conv1D-LSTM [24]	RMSE = 0.0081, MAE= 56, MAPE=1.98%	

Figure 2.1 Comparation among some algorithms

2.2 Introduce to Long-Short Term Memory Model

Long short-term memory is a variant of the RNN model to overcome the problem of RNNs that is not able to remember results from very long ago. This makes it possible to use RNNs for problems with very long input such as the problem of writing sentences in a paragraph, the problem of translating a paragraph, or even predicting stock prices based on 100, 200 ,.. the previous day became very difficult. Let's explain through this problem in mathematical form:

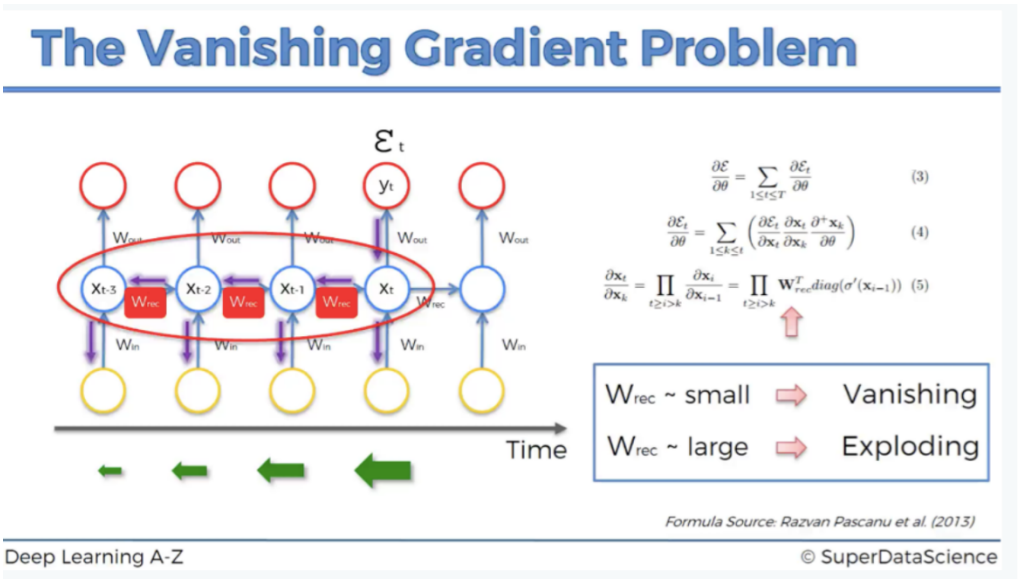


Figure 2.2 Explanation about vanishing gradient problem in RNN

As we all know the backpropagation process is one of the very important processes of each neural network that helps to change the parameters to align the output results closer and closer to the labeling results. Moreover, observing the above figure, usually

the parameters will change after each learning according to the formula:

$$Wrec = Wrec - \frac{\partial \varepsilon}{\partial \theta}$$

Which the above derivative depends directly on W_{rec} as shown in the figure. That leads to 2 problems. If $W_{rec}<1$ then the derivative will get lower over time and go to 0. If $W_{rec}>1$ then the derivative will explode and become very large. When the derivative goes to 0, that leaves the parameters almost unchanged after each learning. Scientists have devised an LSTM neural network based on RNN to avoid the vanishing derivative and the model has the ability to store memory for a long time.

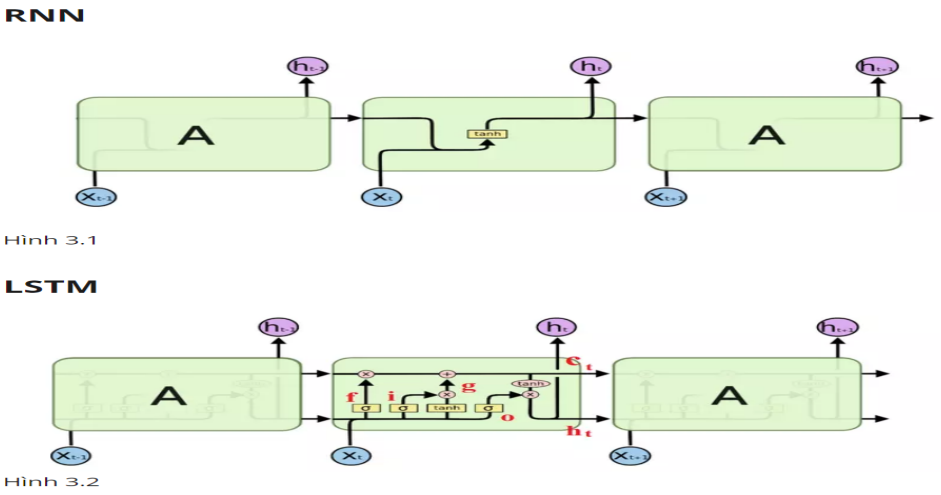


Figure 2.3 Difference between RNN and LSTM

LSTM network configuration:

Each cell of the LSTM will include 3 inputs that are:

- Long-memory (c): Stores all previously learned information from far away.
- Short-memory(h): Đầu ra(thông tin dự đoán) của ô trước đó.
- Input(x): Đầu vào(kết quả gán nhãn) của ô hiện tại.

In which 3 input information will be combined and processed through 3 main gates:

- Input gate(i): The input gate helps to decide how much of the input will affect the new state. Decide how, through the characteristic of the sigmoid function (the output is in the range [0,1][0,1]), so that when an information vector passes through here, if multiplied by 0, the vector will be completely annihilated. If multiplied by 1, most of the information will be retained.
- Forget gate(f): The gate decides how much information to discard from the previous state.

- Output gate(o): The gate regulates the amount of information that can go out y_t and the amount of information transmitted to the next state.

Initially put the long-memory stream into the forget gate first to calculate how much of that long-term information can be retained and how much can be deleted. Next, the input gate will provide the current and updated information to the long-memory. Finally, the output gate calculates the output prediction $h_{(t+1)}$ based on long-memory, short-memory(previous prediction) and input(current label result).

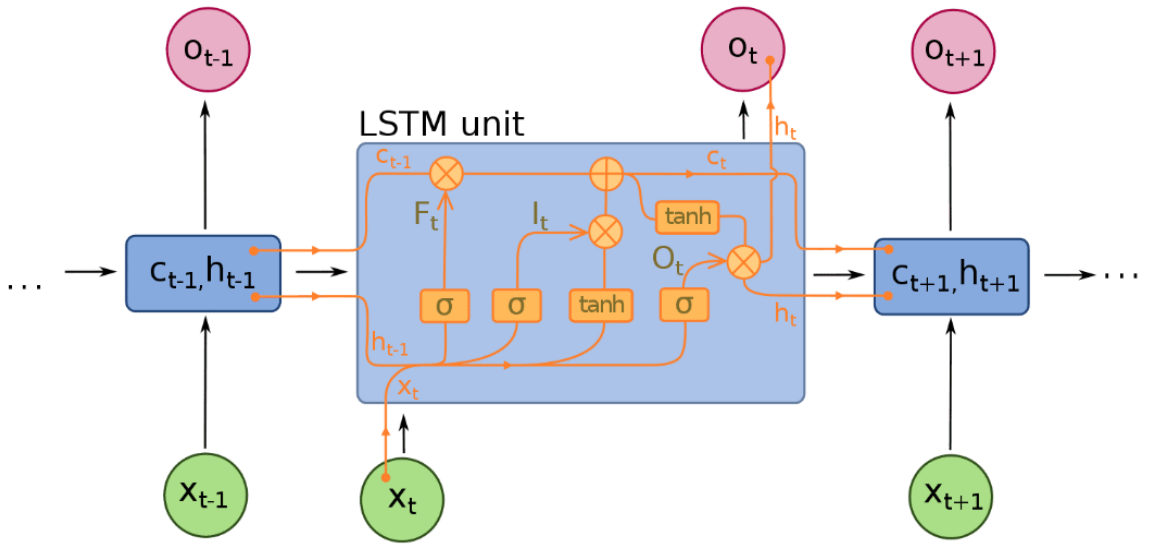


Figure 2.4 Visualization of a LSTM unit

Formulas for the LSTM feedforward process:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right) \quad (1)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \quad (3)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \quad (4)$$

$$h_t = o_t \odot tanh\left(c_t\right)$$

Why LSTM can solve the problem of vanishing gradient?:

Backpropagation in LSTM:

The error of gradients:

$$\frac{\partial \epsilon_k}{\partial W} = \frac{\partial \epsilon_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\cdots\frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

$$= \frac{\partial \epsilon_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W}$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \acute{\sigma}\left(W_f\left[h_{t-1}, x_t\right]\right) W_f o_{t-1} \otimes \dot{tanh}\left(c_{t-1}\right) c_{t-1}$$
$$+ f_t + \dot{\sigma}\left(W_i\left[h_{t-1}, x_t\right]\right) W_i o_{t-1} \otimes \dot{tanh}\left(c_{t-1}\right) \tilde{c}_t$$

$$+ \dot{\sigma}\left(W_c\left[h_{t-1}, x_t\right]\right) W_c o_{t-1} \otimes \dot{tanh}\left(c_{t-1}\right) i_t$$

Observed during backpropagation , the component that directly causes vanishing gradient in RNN is:

$$\frac{\partial s_{t+1}}{\partial s_t} = \left(1 - s_t^2\right) * W, \ in\ which\ s\_\{t\},\ W < 1$$

In LSTM that component has been transformed and depends not only on the parameter W, but also has an additional component f_t:

$$\frac{\partial c_t}{\partial c_{t-1}} = f_t$$

Since $0 < f_t < 1$, the LSTM basically still has a vanishing gradient but less than that of the RNN. Moreover, when carrying information on long-memory, it is seldom necessary to forget the old cell value, so $f \approx 1$ and it can avoid vanishing gradient. Therefore, LSTM is more commonly used than RNN for string information operations.

2.3 Traditional indicators

This section will cover common indicators that aid in candlestick chart analysis that will be applied to this project.

2.3.1 Moving Average

In finance, a moving average (MA) is a stock indicator commonly used in technical analysis. The reason for calculating the moving average of a stock is to help smooth out the price data by creating a constantly updated average price.

By calculating the moving average, the impacts of random, short-term fluctuations on the price of a stock over a specified time frame are mitigated. Simple moving averages (SMAs) use a simple arithmetic average of prices over some timespan, while exponential moving averages (EMAs) place greater weight on more recent prices than older ones over the time period.

There are 2 types of moving average:

- **Simple Moving Average:**

A simple moving average (SMA), is calculated by taking the arithmetic mean of a given set of values over a specified period. A set of numbers, or prices of stocks, are added together and then divided by the number of prices in the set. The formula for calculating the simple moving average of a security is as follows:

$$SMA = \frac{A_1 + A_2 + \cdots + A_n}{n}$$

where:

A=Average in period n

n=Number of time periods

- **Exponential Moving Average (EMA)**

The exponential moving average gives more weight to recent prices in an attempt to make them more responsive to new information. To calculate an EMA, the simple moving average (SMA) over a particular period is calculated first.

$$EMA_t = \left[V_t\left(\frac{s}{1+d}\right)\right] + EMA_y\left[1 - \left(\frac{s}{1+d}\right)\right]$$

Where:

EMAt=EMA today

Vt=Value today

EMAy=EMA yesterday

s=Smoothing

d=Number of days

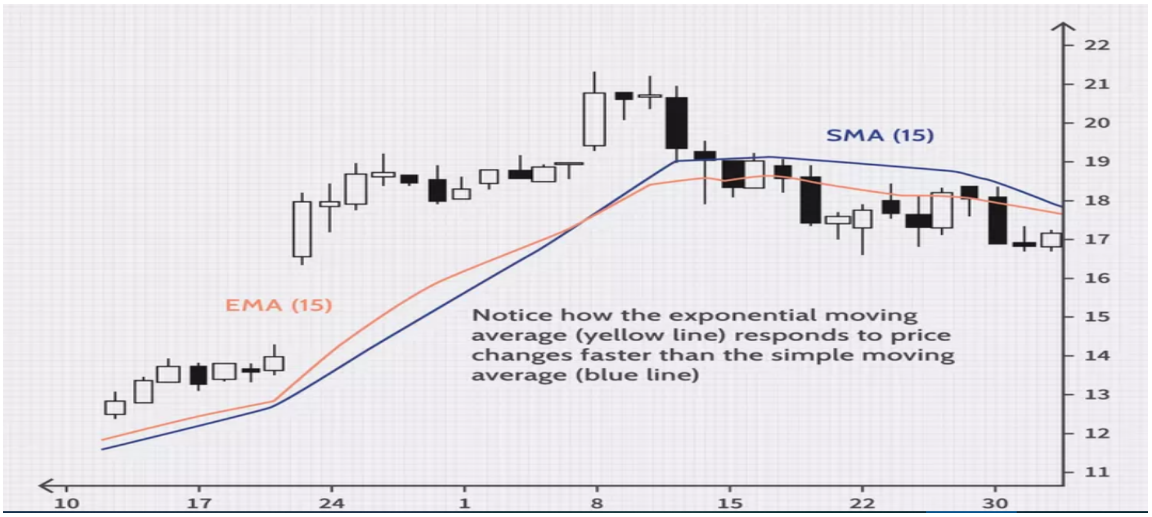


Figure 2.5 Visualization of Moving Average indicator

2.3.2 Bollinger Bands

Bollinger Bands are indicators that are plotted at standard deviation levels above, and below a simple moving average. Since standard deviation is a measure of volatility, a large standard deviation indicates a volatile market, and a smaller standard deviation indicates a calmer market. Bollinger Bands are a good way to compare volatility against relative price levels, over a period of time.

$$MA = \frac{\sum_{i=1}^{n} y\_\{i\}}{n}$$

$$UpperBB = MA + D\sqrt{\frac{\sum_{i=1}^{n}\left(y_i - MA\right)^2}{n}}$$

$$LowerBB = MA - D\sqrt{\frac{\sum_{i=1}^{n}\left(y_i - MA\right)^2}{n}}$$

$$MiddleBB = MA$$

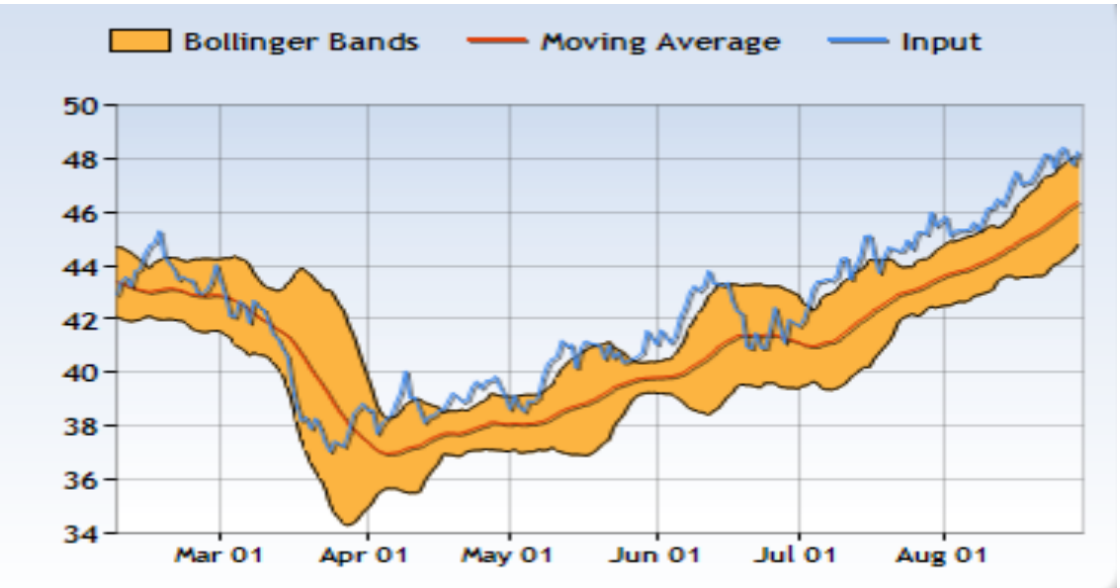


Figure 2.6 Visualization of Bollinger Bands indicator

2.3.3 Relative Strength Index (RSI)

The relative strength index (RSI) is a momentum indicator used in technical analysis. RSI measures the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security.

$$RS = \frac{Average\ Gain}{Average\ Loss}$$

$$RSI = 100 - \frac{100}{1+RS}$$

The average gain or loss used in this calculation is the average percentage gain or loss during a look-back period. The formula uses a positive value for the average loss. Periods with price losses are counted as zero in the calculations of average gain. Periods with price increases are counted as zero in the calculations of average loss.

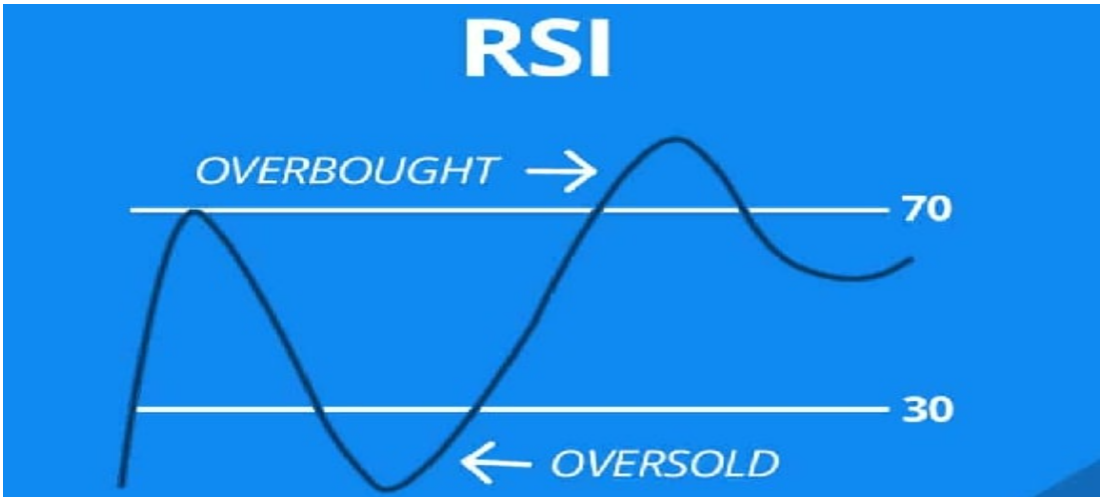


Figure 2.7 Visualization of RSI indicator

2.3.4 Moving Average Convergence/Divergence(MACD)

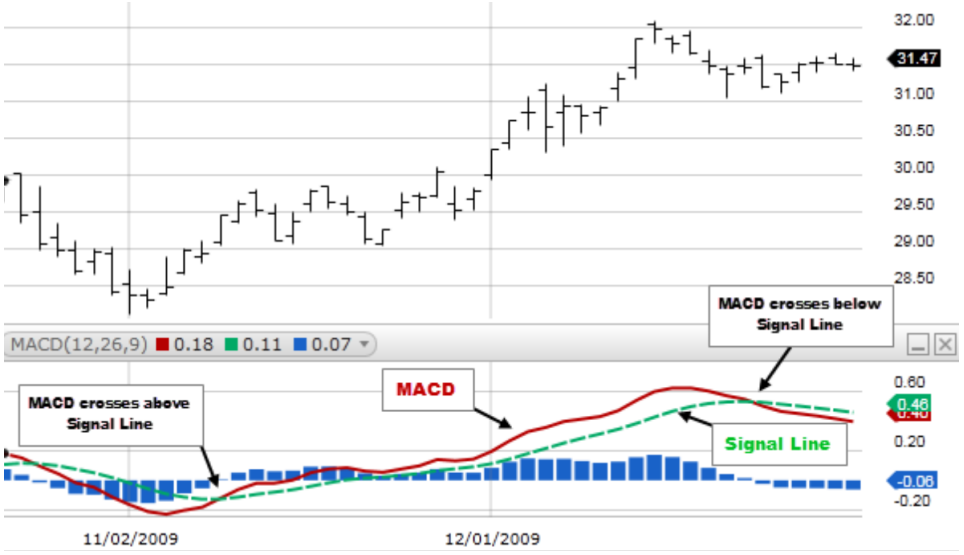
The Moving Average Convergence/Divergence indicator is a momentum oscillator primarily used to trade trends. Although it is an oscillator, it is not typically used to identify over bought or oversold conditions. It appears on the chart as two lines which oscillate without boundaries. The crossover of the two lines give trading signals similar to a two moving average system.

MACD crossing above zero is considered bullish, while crossing below zero is bearish. Secondly, when MACD turns up from below zero it is considered bullish. When it turns down from above zero it is considered bearish.

Figure 2.8 Visualization of MACD indicator

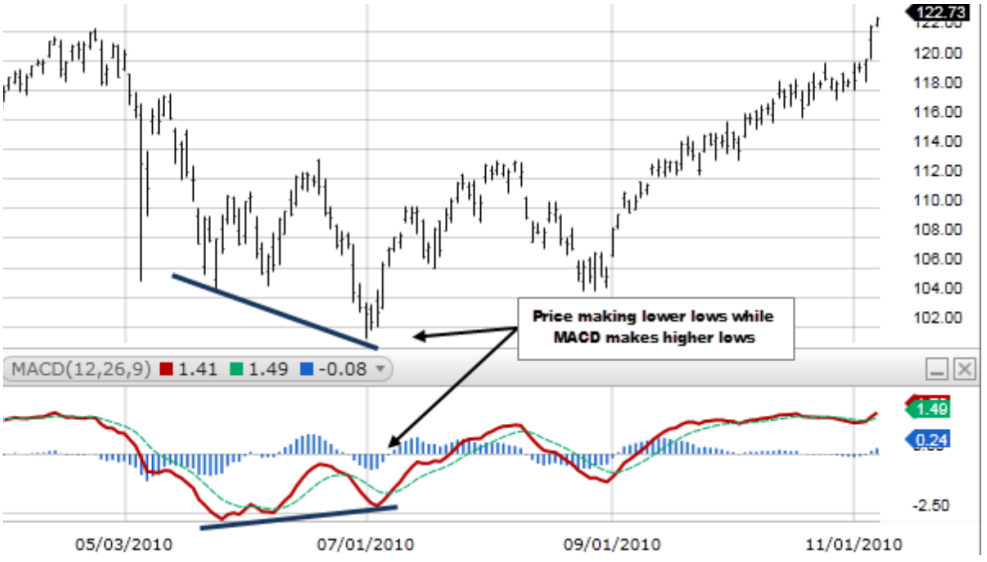
When the MACD line crosses from below to above the signal line, the indicator is considered bullish. The further below the zero line the stronger the signal.

When the MACD line crosses from above to below the signal line, the indicator is considered bearish. The further above the zero line the stronger the signal.



During trading ranges the MACD will whipsaw, with the fast line crossing back and forth across the signal line. Users of the MACD generally avoid trading in this situation or close positions to reduce volatility within the portfolio.

Divergence between the MACD and the price action is a stronger signal when it confirms the crossover signals.



Calculation: An a*pproximated* MACD can be calculated by subtracting the value of a 26 period Exponential Moving Average (EMA) from a 12 period EMA. The shorter EMA is constantly converging toward, and diverging away from, the longer EMA. This causes MACD to oscillate around the zero level. A signal line is created with a 9

period EMA of the MACD line.

CHAPTER 3. METHODOLOGY

In Chapter 2, I surveyed and analyzed the theoretical basis of the AI model and important indicators. In Chapter 3, I presented about the technologies used to develop websites, tools to display data and support investors. The system is divided into three blocks: AI block, database block and software block. The software block uses the ReactJS library to build the website interface, the open source library chartjs to build the features of the chart. The em database block uses .csv files to store data. And for the AI block, I use the pytorch library and sublibraries to build models, process data, and train models.

3.1 Data Block

In this graduation research 1, I just built a static database, haven't used the real-time data scraping API from lightning price lists. I directly download 10 csv files containing information about 10 stock codes from website cophieu68. The data of each code is updated from 2010 to present.

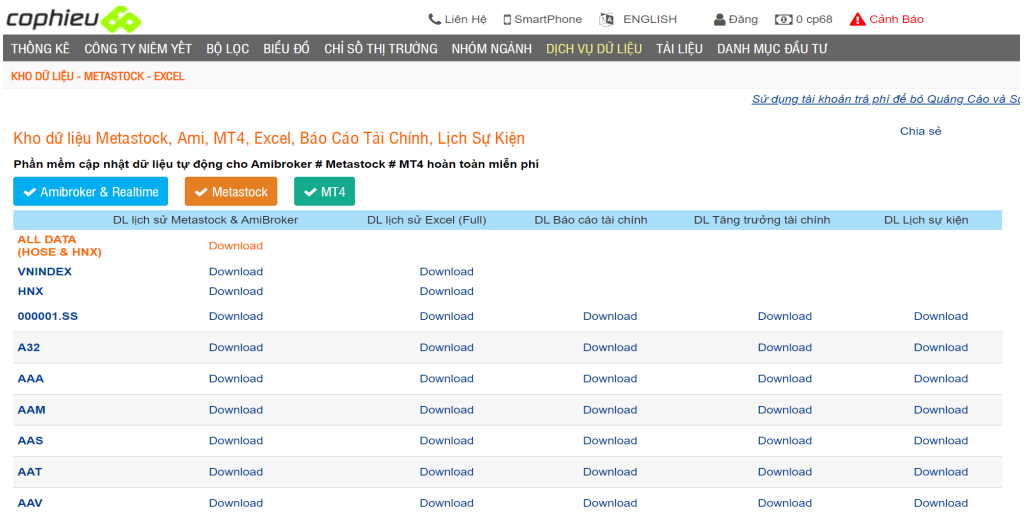


Figure 3.1 Database on website cophieu68

3.2 AI Block

3.2.1 Python language

First, what is Python? Python is a high-level computer programming language commonly used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of programs and is not specialized for any particular problem with:

- Optimal support frameworks for data processing such as os, pandas, numpy
- Interpreted languages: Python is handled at runtime by the Python Interpreter.
- Object-oriented language: It supports object-oriented programming features and techniques.
- Interactive programming languages: Users can directly interact with the python interpreter to write programs.
- Languages that are easy to learn: Python is easy to learn, especially for beginners.
- Simple Syntax: Python syntax formulation is very simple and straightforward, which also makes it popular.
- Easy to read: Python source code is clearly defined and visible to the eye.
- Portable: Python code can run on multiple hardware platforms with the same interface.
- Extensible: Users can add low-level modules to the Python interpreter.
- Possible improvement: Python provides an improved structure to support large programs then shell-script.

3.2.2 Pytorch libraby

Pytorch is an open source framework for Deep Learning developed by Facebook. Along with Tensorflow and Keras, it is one of the most popular frameworks used in artificial intelligence problems.

Possessing a large number of users and a strong community, Pytorch has risen to the 2nd position in terms of the number of users after Tensorflow in 2019.

In the study of artificial intelligence applications, Pytorch is often preferred because it can help implement problems easily. Pytorch provides powerful support for developers to implement Debug (debug) and Data Visualize (data visualization) with the Dynamic Computation Graph mechanism.

Benefits of using PyTorch:

- Open source helps PyTorch build a strong community with quality resources.
- Powerful graphics processing for clear CPU & GPU control.
- Collection of many natural Pythonic.
- Easily handle code when encountering bugs.

- Having TouchScript helps deploy applications to production scale to scale.
- Basic functions and syntax in Pytorch help solve AI problems quickly

Basic Features of PyTorch:

- Tensor:

  Tensor may be a view or matrix that is multidimensional and represents data types. All values in a tensor will have the same data type. The shape of the data is also the size of the array or matrix.

  Tensor can also be processed by CPU or GPU to make it work faster. There are different types of Tensor like Float Tensor, Double Tensor, Half Tensor, Int Tensor and Long Tensor. PyTorch uses 32-bit Float Tensor as default.

- Some math activities:

  The code for performing math operations in PyTorch is the same as in Numpy (a popular Python math library). The user needs to start up 2 Tensors and perform math operations like addition, subtraction, multiplication and division with them.

- Dynamic Calculation Graph:

  Calculation schemes in PyTorch allow the framework to compute gradient values for Neural Networks to be built. PyTorch uses Dynamic Computation Graph which allows users to build interlaced and value graphs. In addition, this type of graph is Debug-friendly because it allows code to be executed line by line.

  In summary, Dynamic Compute Graph is a key feature that makes PyTorch the preferred choice in the industry.

- Datasets and DataLoader

  Working with large datasets requires loading all data into memory in a single pass to save time. This causes memory fullness and slow programs.

  PyTorch provides two initial data areas, DataLoader and Dataset, allowing users to use their own data as well as previously loaded data files.

- Restart matrix

  To initialize a matrix with random numbers in PyTorch, you would use the randn() function and provide a tensor filled with random numbers. The basic matrix operations and position transformations in PyTorch are similar to NumPy.

Popular PyTorch modules:

- Autograd

  Autograd is PyTorch's automatic segregation module. This module generates a

directed ac graph with an input tensor and an output tensor.

- Optimal

  Optim is a package with pre-written algorithms for optimizers, which can be used to build Neural Networks (Artificial Neural Networks in AI)

- nn

  nn includes different classes that help build Neural Network models. All modules in PyTorch are subclasses of nn.

3.3 Software Block

3.3.1. ReactJS

ReactJS is an opensource developed by Facebook, launched in 2013, itself is a Javascript library used to build interactions with components on the website. One of the most outstanding features of ReactJS is that rendering data can not only be done on the Server layer, but also below the Client.

Advantages of ReactJS:

- Suitable for a variety of website types: ReactJS makes website creation easier because you don't need to code as much as when creating a website using only JavaScript, HTML and it already provides you with all kinds of "toys". ” so you can use it for many situations.
- Reusing Components: If you build components well enough, flexible enough to be able to meet the "requirements" of many different projects, you only spend time building and reusing almost all of them. in the following projects. Not only ReactJS, but frameworks today also allow us to do that, for example Flutter.
- Can be used for mobile applications: Most of us know that ReactJS is used for website programming, but in fact it was born for more than just that. If you need to develop more Mobile applications, then use React Native - another framework developed by Facebook itself, you can easily "share" components or reuse Business Logic in the application.
- SEO-friendly: SEO is an integral part of putting your website's information on the top of Google search. ReactJS is essentially a JavaScript library, Google Search Engine has now crawled and indexed JavaScript code, but you also need a few other libraries to support this!
- Debug easily: Facebook has released a Chrome extension for debugging during application development. That helps speed up your product release process as well as your coding process.

- Hottest web development tool right now: If you look at the statistics from Google Trends in Vietnam in the image below, browse through the top job sites in Vietnam like Topdev, Itviec, etc. The number of vacancies for the position of React Developer is extremely large along with the extremely attractive salary and current popularity of ReactJS in the Vietnamese market.

3.3.2. Chartjs

Chart.js is one of the open source projects that makes it easy and beautiful for anyone to draw charts that can display data on the web. This project now has more than 41,000 stars and 2600 commits on Github and is regularly updated.

The 4 strongest points of Chart.js are:

- Open source project: whole community develop and bug fix.
- Good compatibility with HTML 5 this is almost mandatory nowadays
- More than 8 most popular chart styles today
- Responsive: best display in all devices from Desktop, Tablet, Mobile

3.3.3. Flask

Flask is a very lightweight Python Web Framework that makes it easy for beginners to learn Python to create small websites. Flask is also extensible to build complex web applications.

Flask Framework is a repository that makes it easier for developers to create web pages that are scalable, efficient, and maintainable by providing reusable code or extensions for web pages. popular tasks.

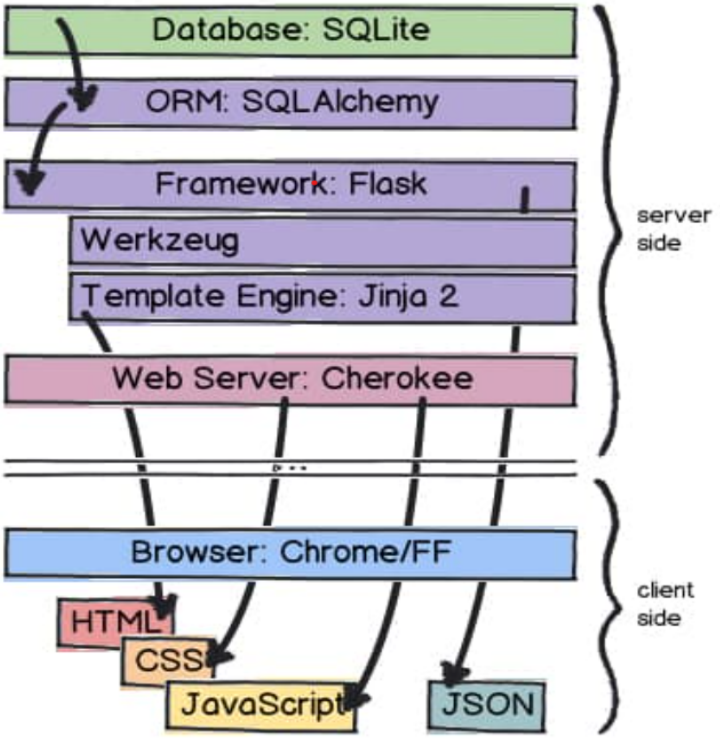


Figure 3.2: How Flask works

Flask is suitable for building small and medium-sized web applications, APIs and web services:

- Building web applications is very similar to writing standard Python modules, the structure is neat and clear.
- Instead of providing everything, Flask provides users with the most commonly used core components of a web application framework such as URL routing, request & response objects, templates, etc.
- With Flask, it's up to us to choose which components for our application. This is great, because each web application has its own characteristics and features, and it doesn't have to contain components it doesn't use.

CHAPTER 4: Design analysis

In this chapter 4, I present from the overview design and detailed design of each system.

4.1 Architecture design

4.1.1 Overall design

Figure 4.1 describes the overall architecture of the website displaying stock data. The components in that architecture are: a database block containing csv files and checkpoints, a software block that includes an application that displays candlestick data and contains prediction tools, an AI block that includes models, data processing, training process.

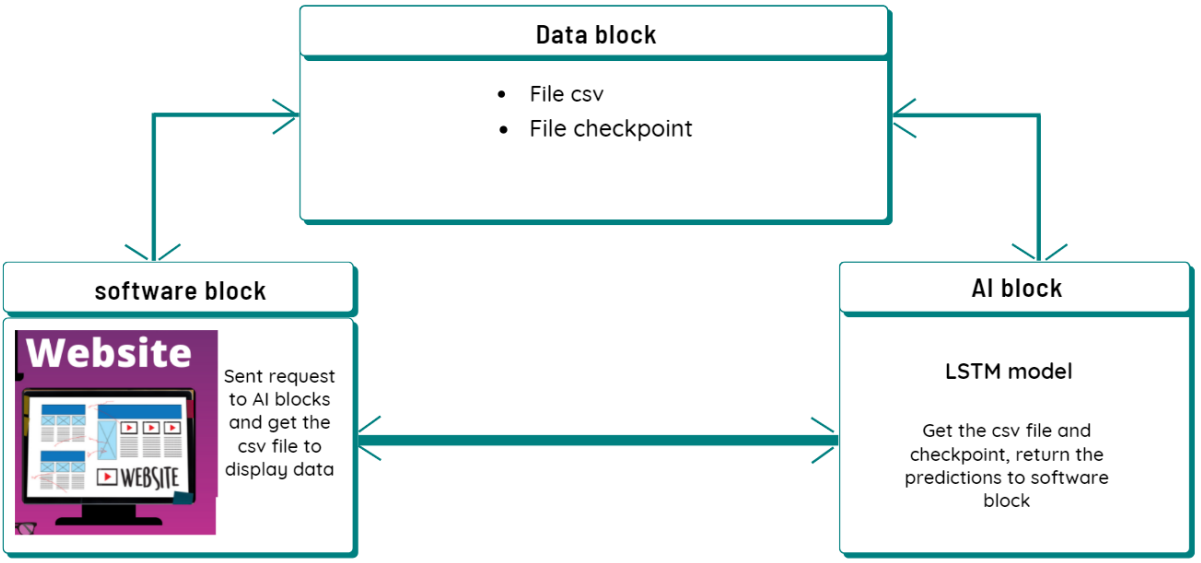


Figure 4.1: Overview of the website architecture.

4.1.2 AI block design

Data processing:

Data processing is an important step before training the model. This helps clean up the data, transforming it into a form that is easily receptive to the model.

- **Read file csv:**

```
def read_file(file_name, main_feature, date_feature, stock_name):  # main_feature now is choosen <CloseFixed>
    df = pd.read_csv(file_name)
    date_ = df.filter([date_feature])
    data = df.filter([main_feature])
    dataset = data.values
    date_ = date_.values.flatten()
    date_data = [str(e)[0:4] + "-" + str(e)[4:6] + "-" + str(e)[6:8] for e in date_]
    stock_namee = df.filter([stock_name])
    return dataset.flatten(), date_data, stock_namee
```

- **Normalize data in range(0-1):**

The goal of normalization is to change the values of numeric columns in the dataset to

a common scale, without distorting differences in the ranges of values

```
def fit_transform(self, x):
    self.mu = np.mean(x, axis=(0), keepdims=True)
    self.sd = np.std(x, axis=(0), keepdims=True)
    normalized_x = (x - self.mu) / self.sd
    return normalized_x
```

- **Inverse transform:**

  This helps transform data back to the original value after normalizing service for work plot and visualizing data.

```
def inverse_transform(self, x):
    return (x * self.sd) + self.mu
```

- **Time Series data:**

The input of the LSTM model requires time series data, so we need to convert it to a list that contains many sublists, each of which contains the data of the previous n days and is labeled as the n + 1 day.

```
def prepare_data_x(x, window_size):
    # perform windowing
    n_row = x.shape[0] - window_size + 1
    output = np.lib.stride_tricks.as_strided(x, shape=(n_row, window_size), strides=(x.strides[0], x.strides[0]))
    return output[:-1], output[-1]


def prepare_data_y(x, window_size):
    # # perform simple moving average
    # output = np.convolve(x, np.ones(window_size), 'valid') / window_size

    # use the next day as label
    output = x[window_size:]
    return output
```

```
def prepare_data(scaler, data_date, num_data_points, normalized_data_close_price, config, plot=False):
    data_x, data_x_unseen = prepare_data_x(normalized_data_close_price, window_size=config["data"]["window_size"])
    data_y = prepare_data_y(normalized_data_close_price, window_size=config["data"]["window_size"])

    # split dataset

    split_index = int(data_y.shape[0] * config["data"]["train_split_size"])
    data_x_train = data_x[:split_index]
    data_x_val = data_x[split_index:]
    data_y_train = data_y[:split_index]
    data_y_val = data_y[split_index:]
```

Figure 4.2: Data visualization.

Build LSTM model:

```
class LSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_layer_size=32, num_layers=2, output_size=1, dropout=0.2):
        super().__init__()
        self.hidden_layer_size = hidden_layer_size

        self.linear_1 = nn.Linear(input_size, hidden_layer_size)
        self.relu = nn.ReLU()
        self.lstm = nn.LSTM(hidden_layer_size, hidden_size=self.hidden_layer_size, num_layers=num_layers,
                            batch_first=True)
        self.dropout = nn.Dropout(dropout)
        self.linear_2 = nn.Linear(num_layers * hidden_layer_size, output_size)

        self.init_weights()
```

```python
def init_weights(self):
    for name, param in self.lstm.named_parameters():
        if 'bias' in name:
            nn.init.constant_(param, 0.0)
        elif 'weight_ih' in name:
            nn.init.kaiming_normal_(param)
        elif 'weight_hh' in name:
            nn.init.orthogonal_(param)
```

```python
def forward(self, x):
    batchsize = x.shape[0]

    # layer 1
    x = self.linear_1(x)
    x = self.relu(x)

    # LSTM layer
    lstm_out, (h_n, c_n) = self.lstm(x)

    # reshape output from hidden cell into [batch, features] for `linear_2`
    x = h_n.permute(1, 0, 2).reshape(batchsize, -1)

    # layer 2
    x = self.dropout(x)
    predictions = self.linear_2(x)
    return predictions[:, -1]
```

Training LSTM model:

I use MSE loss , batch size 64 and Adam optimizer.

```python
def run_epoch(config, scheduler, optimizer, criterion, model, dataloader, is_training=False):
    epoch_loss = 0
    if is_training:
        model.train()
    else:
        model.eval()
    for idx, (x, y) in tqdm(enumerate(dataloader), total=len(dataloader)):
        if is_training:
            optimizer.zero_grad()
        batchsize = x.shape[0]
        x = x.to(config["training"]["device"])
        y = y.to(config["training"]["device"])
        out = model(x)
        loss = criterion(out.contiguous(), y.contiguous())
```

```
        if is_training:
            loss.backward()
            optimizer.step()
        epoch_loss += (loss.detach().item() / batchsize)
    lr = scheduler.get_last_lr()[0]
    return epoch_loss, lr
```

Save model into checkpoints files:

```
def save_model(epochs, model, optimizer, criterion, checkpoint_name):
    """
    Function to save the trained model to disk.
    """
    torch.save({
        'epoch': epochs,
        'model_state_dict': model.state_dict(),
        'optimizer_state_dict': optimizer.state_dict(),
        'loss': criterion,
    }, f"E:/GR1/checkpoint/excel_fpt/" + str(checkpoint_name))
    print("Saved sucessfully!!")
```

```
E:\GR1\venv\Scripts\python.exe E:\GR1\main.py
Train data shape (3208, 20, 1) (3208,)
Validation data shape (802, 20, 1) (802,)
100%|██████████| 51/51 [00:02<00:00, 22.00it/s]
100%|██████████| 13/13 [00:00<00:00, 79.33it/s]

Epoch[1/100] | loss train:0.007710, test:0.338503 | lr:0.010000
Saved sucessfully!!
100%|██████████| 51/51 [00:01<00:00, 45.53it/s]
100%|██████████| 13/13 [00:00<00:00, 81.76it/s]

Epoch[2/100] | loss train:0.001608, test:0.306573 | lr:0.010000
Saved sucessfully!!
 67%|██████    | 34/51 [00:00<00:00, 40.00it/s]
```

Figure 4.3: Training process.

4.1.3 Software block design

- **Candlesticks bar chart:**

Each candlestick bar is designed from the original bar chart in chartjs library with

idea:

- Cut through the bar chart and just show from low to high value.
- In the center of head and tail of bar chart, draw 2 lines from high to open and low to close or vice versa.
- Which bar chart has value open < close will be colored green, and vice versa is red.

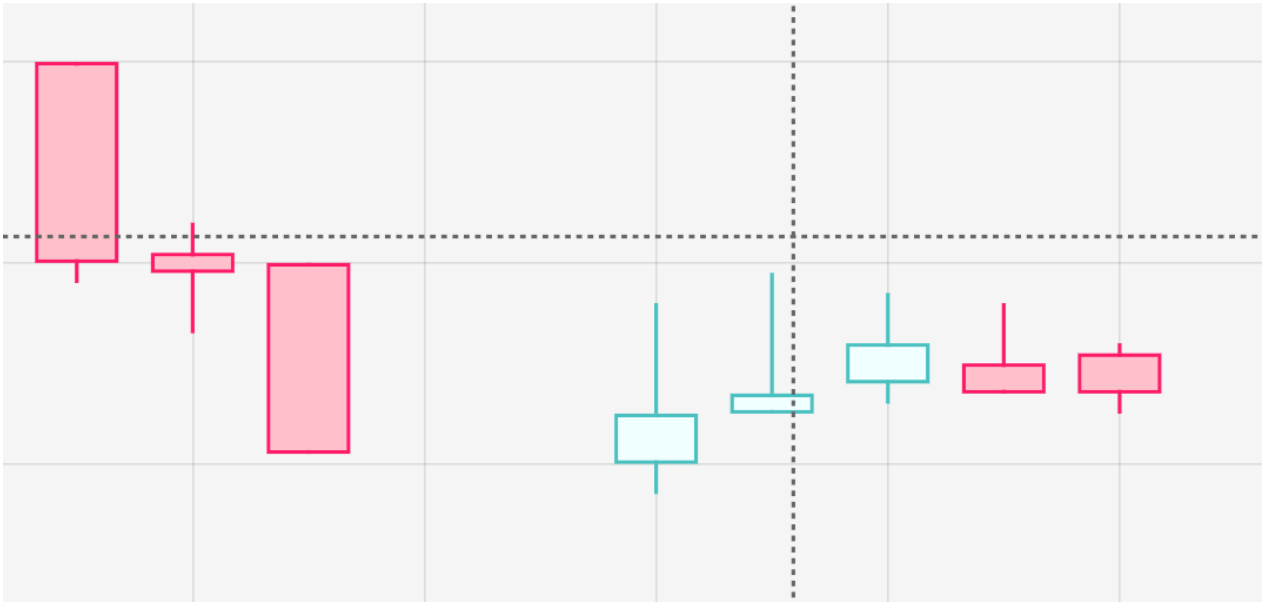


Figure 4.4: Candlestick chart .

Comparing the results directly with the lightning SSI price list, we see that the data displayed by the candlestick is correct and coincides with the SSI.

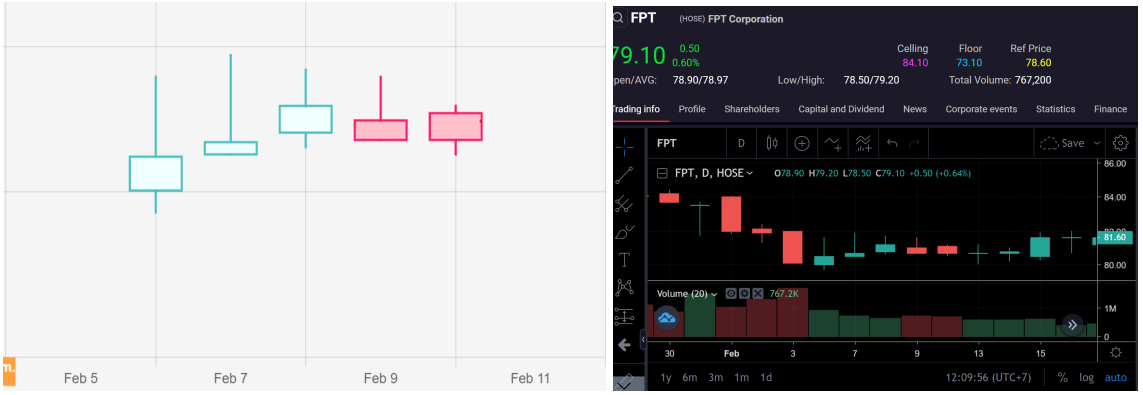


Figure 4.5: FPT stock data for 5 days from February 6 to February 10 on my own website with SSI lightning.

Toolbar

Toolbar includes 4 tools for supporting users draw directly into the chart: trendline, fib

retracement, brush, and text.

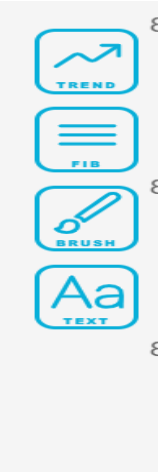


Figure 4.5: Tool bar .

- Trendline:

A trendline is a chart feature used to determine the overall direction and trajectory of the price of an asset.

Trendlines are one of the most fundamental aspects of financial analysis. Using a simple line or pair of lines on a chart — hence ‘trend line’ — traders can see whether an asset is in an uptrend or downtrend and how strong that trend is.

In this website, I use canvas library to draw a line from a start click coordinates to the end click coordinates. After each time creating a trendline, the coordinates of all points in the trendline will be stored into an array. This array always be brought back to draw all existed trend lines again whenever chart is change.

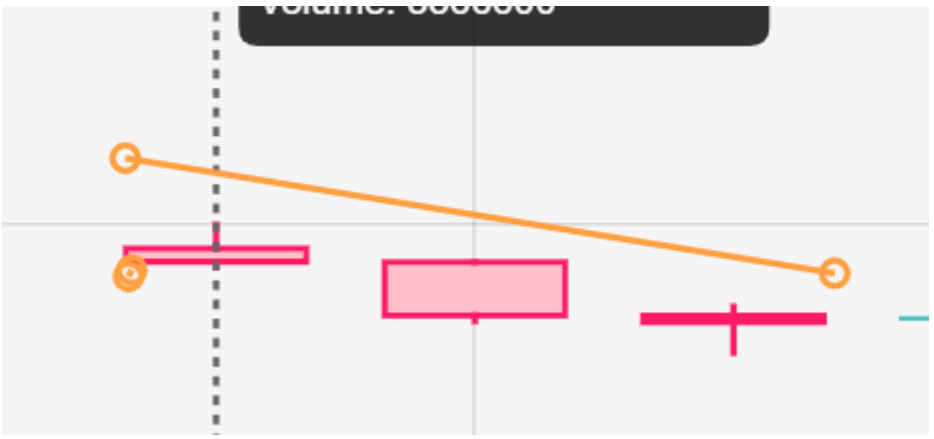


Figure 4.6: Trend line .

- Fibonacci Retracement:

Fibonacci retracement levels—stemming from the Fibonacci sequence—are horizontal lines that indicate where support and resistance are likely to occur.

Each level is associated with a percentage. The percentage is how much of a prior move the price has retraced. The Fibonacci retracement levels are 23.6%, 38.2%, 61.8%, and 78.6%. While not officially a Fibonacci ratio, 50% is also used.

The indicator is useful because it can be drawn between any two significant price points, such as a high and a low. The indicator will then create the levels between those two points.

Suppose the price rises from $10 to $15, and these two price levels are the points used to draw the retracement indicator. Then, the 23.6% level will be at $13.82 ($15 - ($5 × 0.236) = $13.82). The 50% level will be at $12.50 ($15 - ($5 × 0.5) = $12.50).

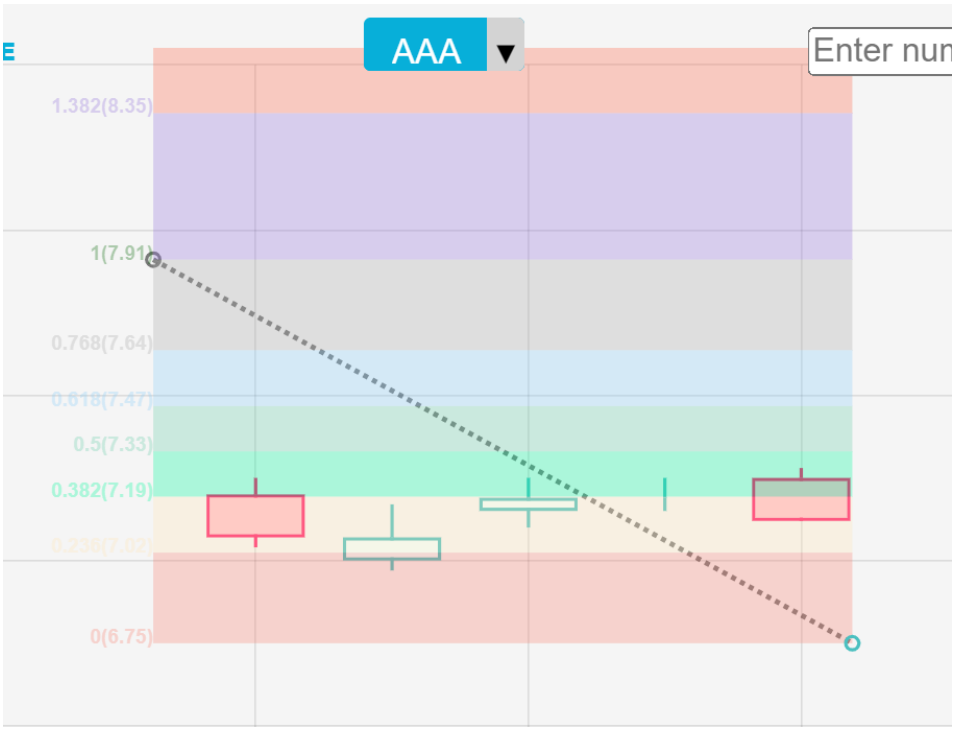


Figure 4.7: Fibonacci Retracement .

Similar to trendline, a line will be drawn from start click mouse coordinates to end click mouse coordinates. In each length in 23.6%, 38.2%, 61.8%, and 78.6% of y, a rectangle will be drawn with different colors. Eventually, all coordinates of a fib retracement also are stored into a different list.

- Brush:

Like to another brush tool in any paint app, on this website, we design a new one that helps people to note in any important place in their screen work.

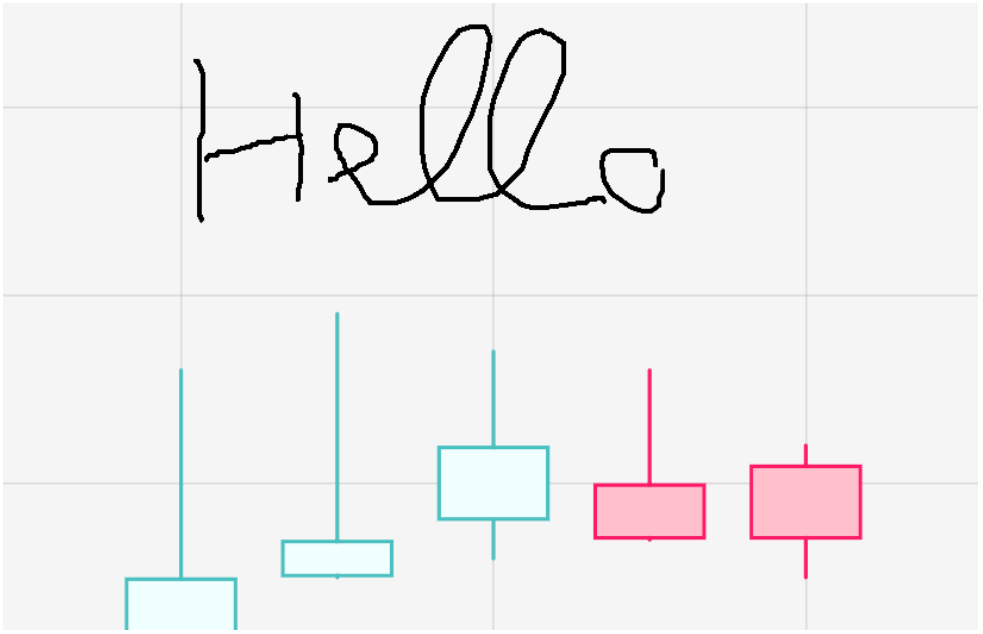


Figure 4.8: Brush tool .

This brush tool will save the coordinates of each mouse move and recolor those coordinates. All coordinates of each draw will also be saved in a new list.

- Text Note:

A text box will appear in the coordinate that the user clicks on.

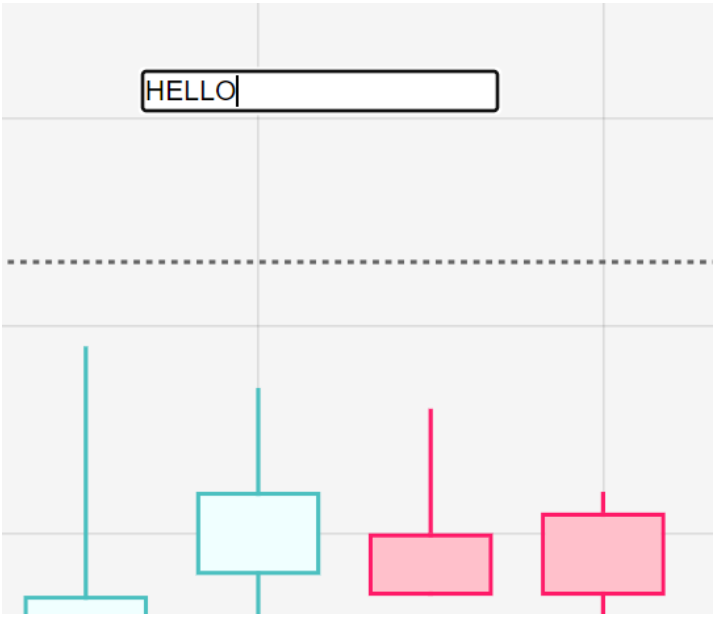


Figure 4.9: Fill text tool .

Indicator bar:

Apply algorithms shown in section 2.3, 4 popular indicators are designed to provide users another look in their data and they can have a best decision later.

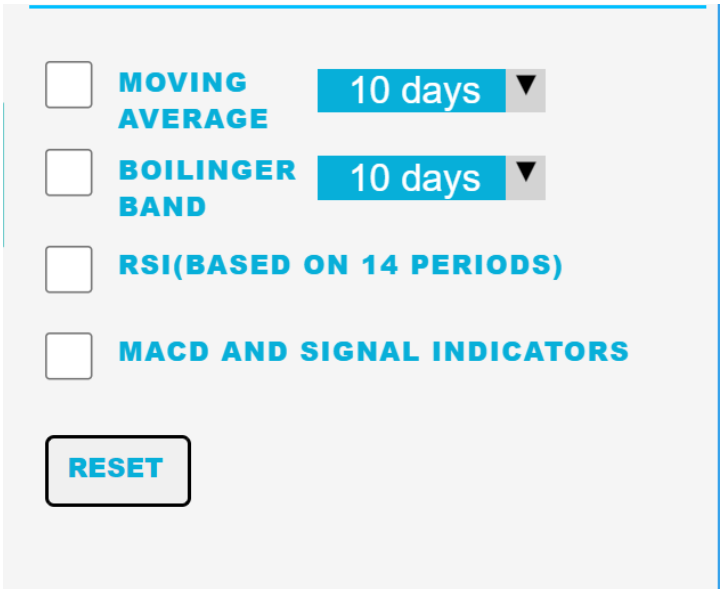


Figure 4.10: Indicator window .

- Moving average:

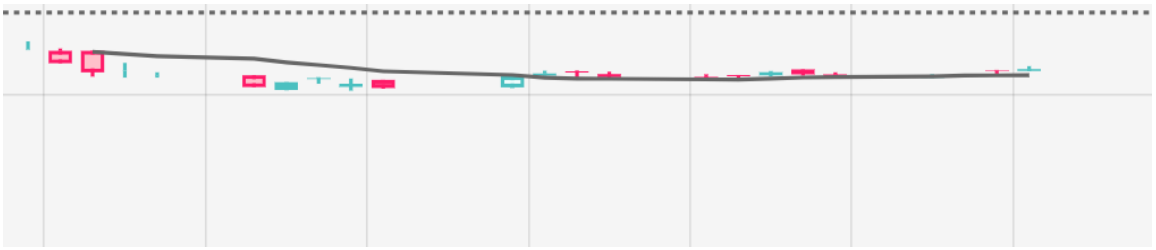


Figure 4.11: Moving average indicator .

- Bollinger Band:

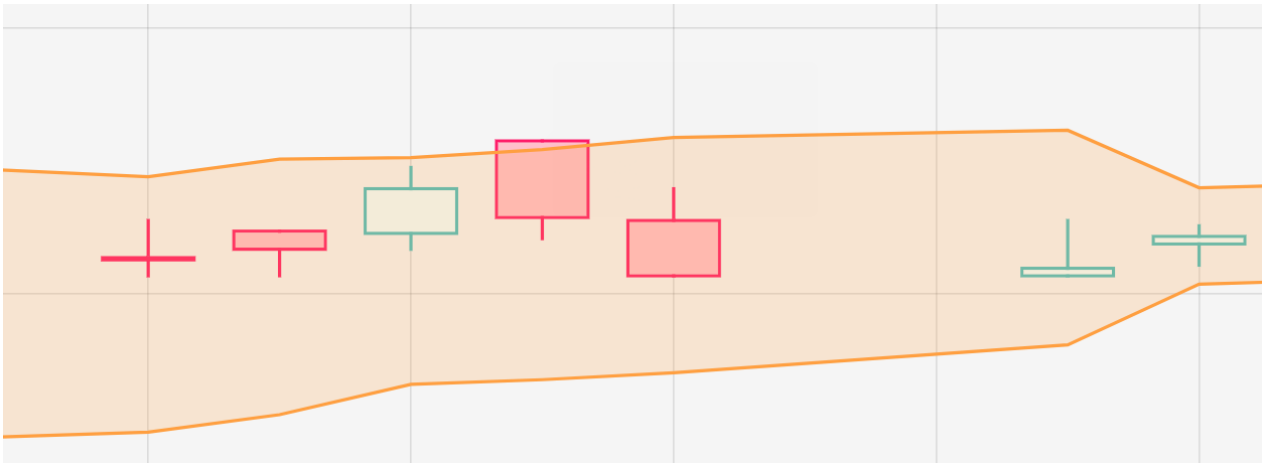


Figure 4.12: Bollinger Band indicator .

- RSI:

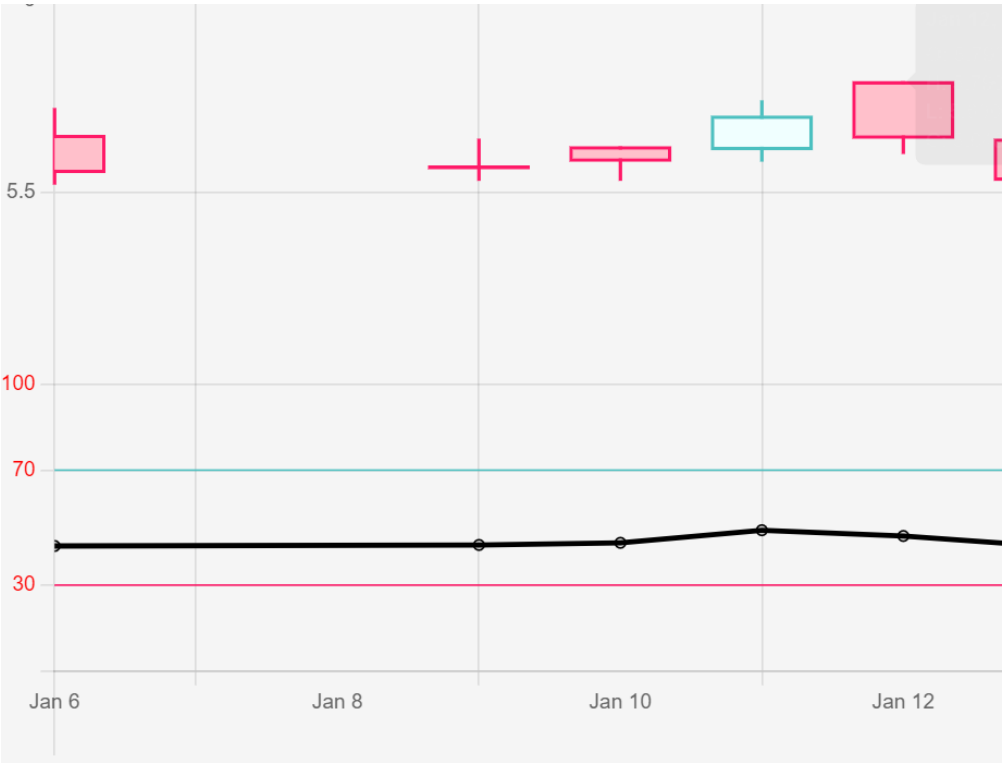


Figure 4.13: RSI indicator .

- MACD and Signal:

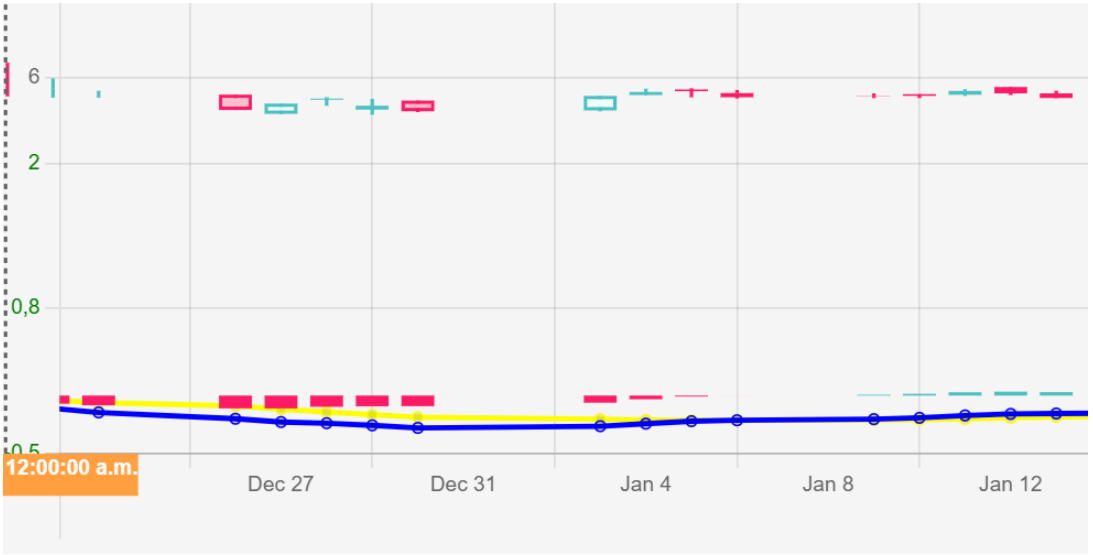


Figure 4.14: MACD and Signal indicator .

- Volume:

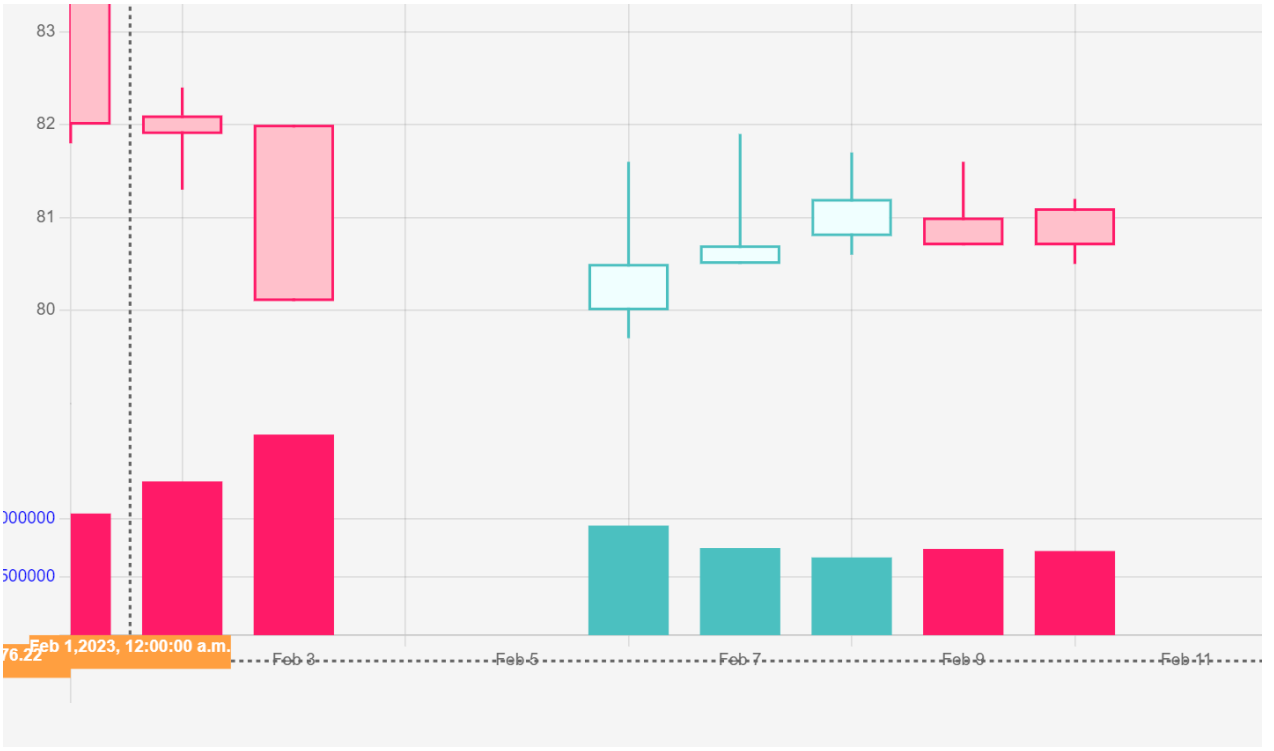


Figure 4.15: Volume indicator .

Data Window:

Data Window shows the detail information of a specific stock: Name, Date, Open, High, Low, Close, Volume.

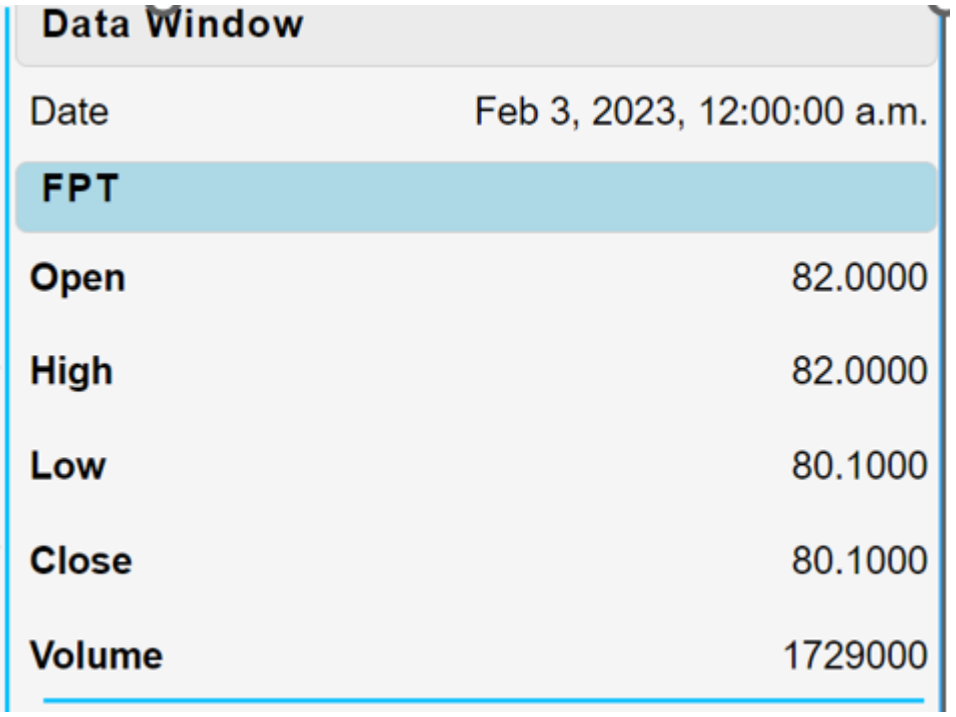


Figure 4.16: Data window.

AI Assistant Tool:

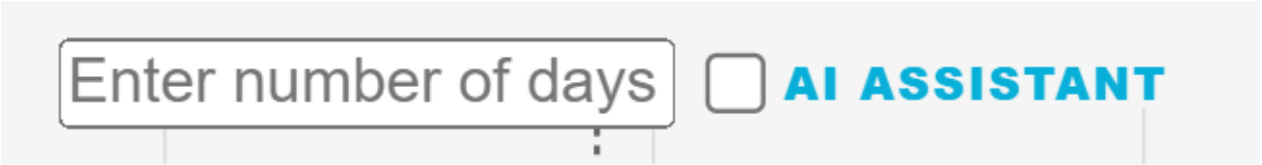


Figure 4.17: AI Assistant Tool.

Website get the input like the number of next days that users require to predict. Using method ajax to send requests to the AI model by storing data into json files. The Flask file store AI model code that gets the request, executes the request, using existing checkpoint to predict data received and sent back to the website, shows prediction into the chart.

Sending request to the Flask:

```
$.ajax({
    url:"/test",
    type:"POST",
    contentType: "application/json",
    data: JSON.stringify(s),
    success: function (data) {
```

Flask get data from request by decode JSON file and sent back the prediction to website:

```
@app.route('/test', methods=['GET', "POST"])
def test():
    global predict_open, predict_high, predict_low, predict_close, predict_volume, num_days
    output = request.get_json()  # This is the output that was stored in the JSON within the browser
    result = json.loads(output)  # this converts the json output to a python dictionary
    num_days = int(result["number_of_days"])
    url_path = result["url"]
    filename = os.path.basename(url_path)
    name_without_extension = os.path.splitext(filename)[0]
    checkpoint_path = r"E:/GR1/checkpoint/" + name_without_extension
    print(checkpoint_path, num_days,url_path)
    predict_open, predict_high, predict_low, predict_close, predict_volume = predict_all(num_days,
                                                                                          url_path, checkpoint_path)
    pred_data = {"pred_open": predict_open, "pred_high": predict_high,
                 "pred_low": predict_low, "pred_close": predict_close,
                 "pred_volume": predict_volume}

    j = json.dumps(pred_data)
    print(j)
```

Show prediction:

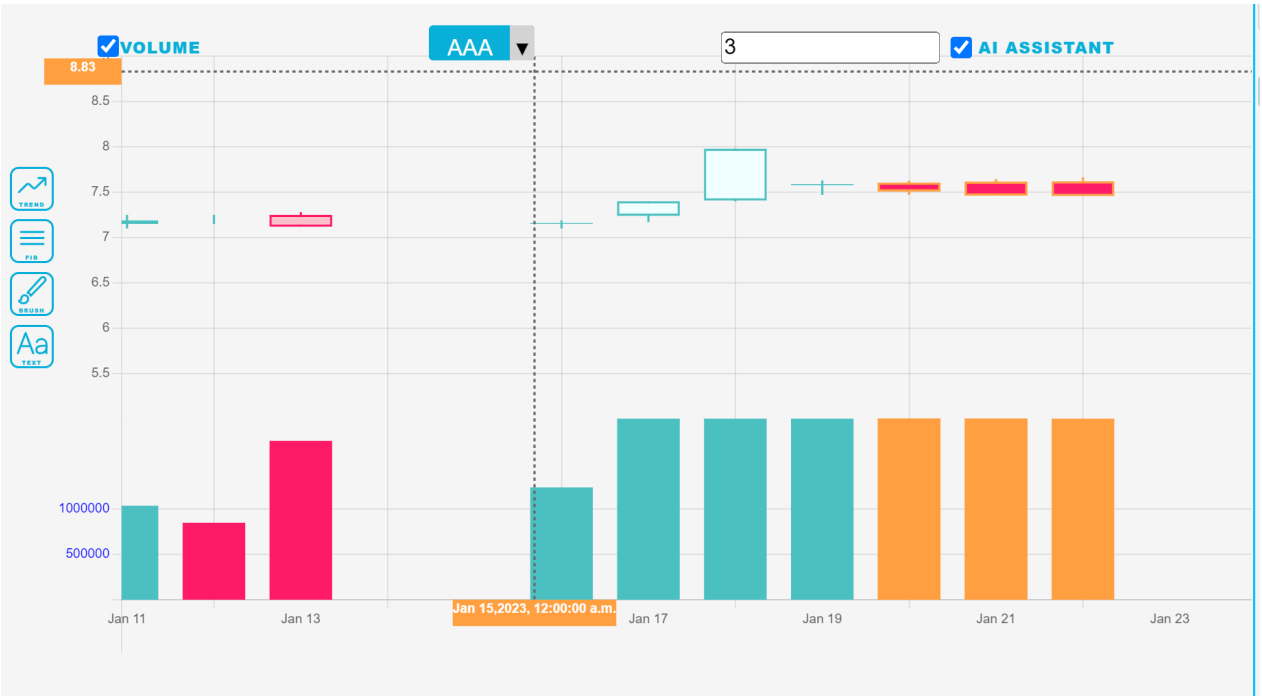


Figure 4.18: Visualization of AI predictions on my website.

CHAPTER 5: Deployment and results

5.1 Achievement

In the process, I have developed an AI model system that helps predict future outcomes based on past long-term information. The system takes as input n days and predicts the result of the n+1 day. Experimental application to the actual fuel when

predicting the information of stocks in the future to achieve relatively feasible results.

5.2 System Deployment

I have trained LSTM model 10 times corresponding to 10 different stocks. Each stock will be extracted from a 10-year history of 5 parameters: open, high, low, close, volume. The training results of each stock are saved in 5 checkpoint files corresponding to the above 5 parameters.

Table 5.1 below describes the training results of 2 representative stocks

Stock	**Type of data**	**Loss in train**	**Loss in test**
AAA	Open	0.004811	0.000333
	High	0.004489	0.000202
	Low	0.004692	0.000288
	Close	0.006158	0.000256
	Volume	0.026001	0.242105
FPT	Open	0.000490	0.265329
	High	0.000466	0.277339
	Low	0.000496	0. 175430
	Close	0.000609	0.291856
	Volume	0.107589	0.330164

Table 5.1 describes the training results of 10 stocks

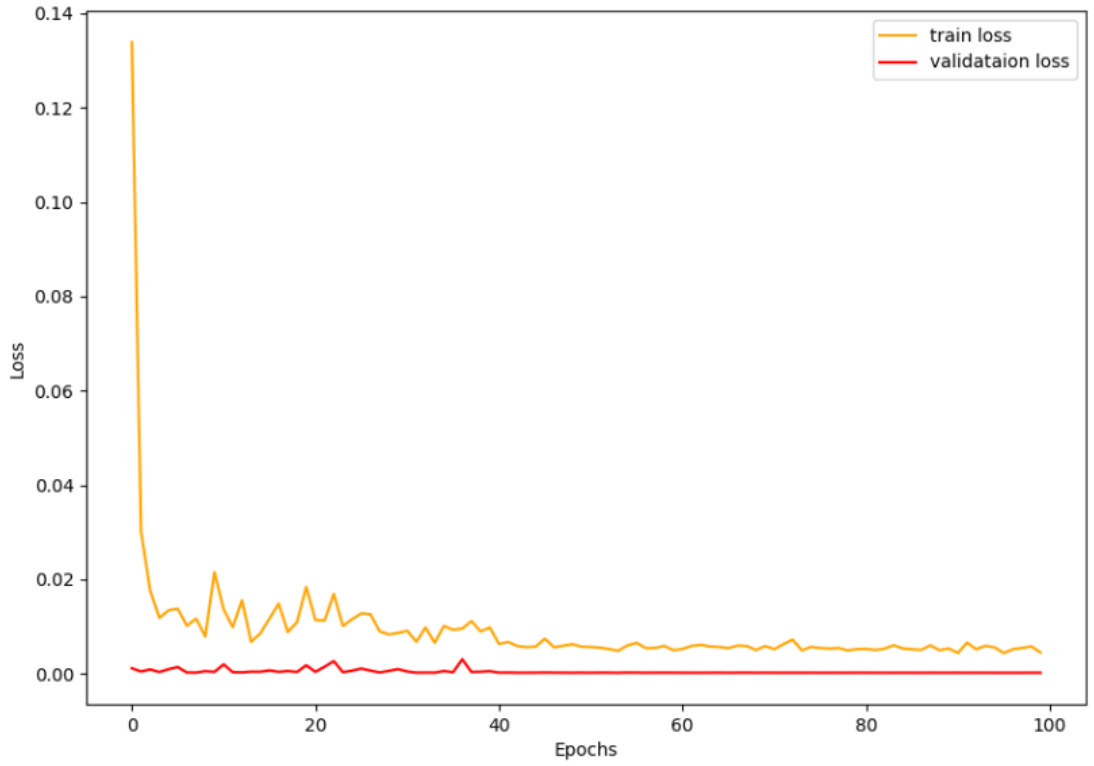


Figure 5.2 describes the loss training set and loss validation set in AAA high price

5.3 Illustrate the deployment of the system and the results achieved

Looking at Table 5.1, it is observed that in some types of data, the loss of value is smaller than the loss of the train, which proves that the train process has good results. However, in some other cases, the loss val is much larger than the loss train. This partly shows that the results seem to be overfit, that is, the prediction is overfit with the training set data, so it does not reach the correctness when trying with the actual data. Observing Figures 5.3 and 5.4 below, we immediately see that the data on FPT's validation set deviates quite a lot from the real data, in which AAA has good accuracy.

Figure 5.3 describe the prediction compare to original data in Open Price in AAA

Figure 5.4 describe the prediction compare to original data in Open Price in FPT

The question is why does this happen. To account for this we need to look back at the original data set. For example, with the OpenFixed field data in the stock code FPT - The dataset is overfit, we compare 20% of the total data at the end, corresponding to the validation set with the initial data set, we can comment that in In the last 20%, the range of data varies quite a bit. At the end of the training set, the data only revolved around 30 000 VND, however in the valid set OpenFixed increased rapidly to 40 000, 50 000 and ended at 80 000. It is the huge difference in the range of data on the set. val and on the train set prevented the model from making the best prediction.

Solution: The best solution to handle overfitting in this problem, I suggested that after the closing data of a day ends, we should scrape today's data, put it in the training set, and

retrain it again. . This helps the model to always adapt to the gradual change of new data every day.

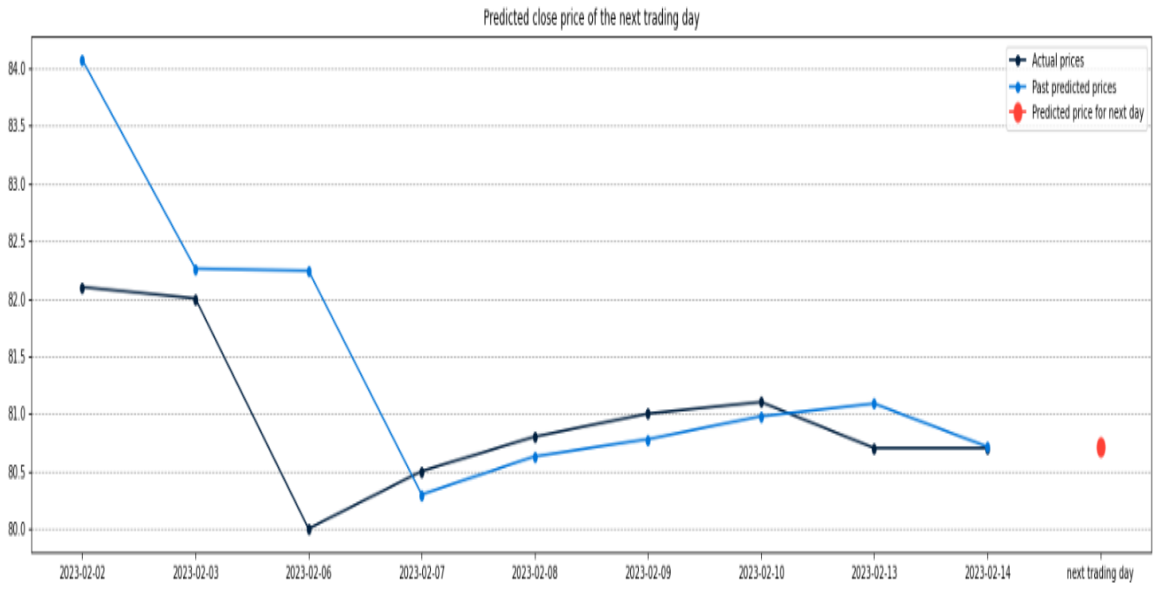


Figure 5.5 Predicted close price of the next trading day

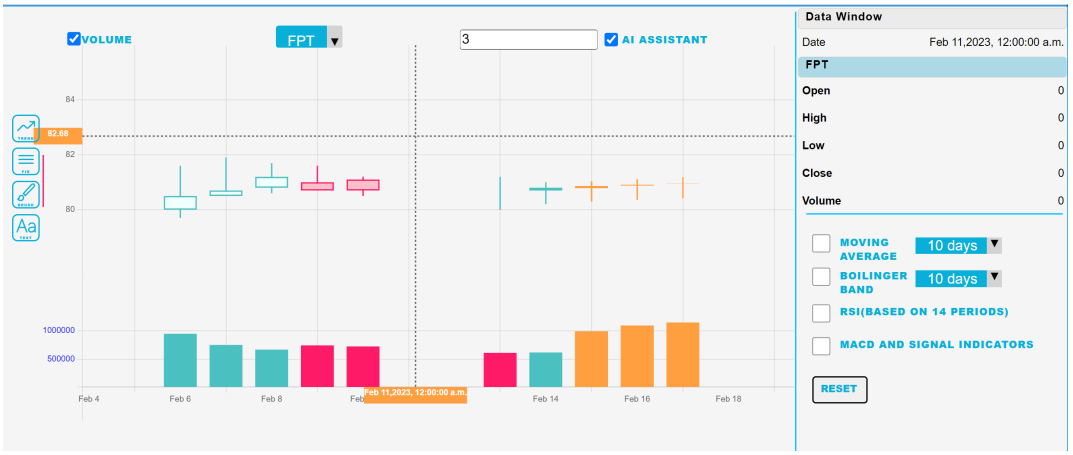


Figure 5.6 describes the result is showed by AI assistant in website interface.

CHAPTER 6. CONCLUSION AND FUTURE WORK

In the previous chapters, I have detailed, examined the problem, analyzed the requirements, built, tested and deployed the system. In this chapter 6, I would like to draw conclusions and directions for future development of the topic.

6.1 Conclusion

As for the chart interface, although I can't provide full support tools as well as many different types of charts like TradingView, in this project I have tried to build the most popular indicators and illustration tools. , the most commonly used. There is a problem when building the chart in this project that the chart cannot load too large data sets. Data sets over 300 days will cause the interface to appear laggy, overloaded and frozen.

As for AI, the model has the ability to predict the next 1, 2 days short-term relatively accurately, but if the long-term range will not be correct due to the data. Moreover, some stocks with high volatility will bring bad results. This will be improved in the next project when I will build a real-time data system and automatically update stock prices and retrain every day.

Through the process of working on the project, I have learned a lot of knowledge and experience that will be useful for my future work: skills in analysis, design, system building, report writing skills,. These are extremely valuable experiences for next projects and gave me more confidence.

6.2 Future work

After the project is over, I will continue to research and improve my product, overcome security limitations, optimize the cost to be able to achieve the highest efficiency and closest to nowadays best system is Trading View.

REFERENCE

[1] LSTM, https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2

[2] Stock data storage, https://www.cophieu68.vn/stockdaily.php

[3] Build Model LSTM, https://www.kaggle.com/code/taronzakaryan/predicting-stock-price-using-lstm-model-pytorch

[4] ChartJS, https://github.com/chartjs/awesome