



A Neural Network-based Stock Timing Model

Zhuoxin Lei

Sendelta International Academy Shenzhen, Shenzhen 518108, Guangdong, China.

How to cite this paper: Zhuoxin Lei. (2025) A Neural Network-based Stock Timing Model. *Journal of Applied Mathematics and Computation*, 9(1), 48-52. DOI: 10.26855/jamc.2025.03.006

Received: February 25, 2025

Accepted: March 23, 2025

Published: April 21, 2025

***Corresponding author:** Zhuoxin Lei, Sendelta International Academy Shenzhen, Shenzhen 518108, Guangdong, China.

Abstract

The stock market is inherently complex and volatile, making stock prediction crucial for investors. Traditional forecasting methods often struggle to adapt to the ever-changing nature of market dynamics. This paper proposes a neural network model for stock timing, aiming to predict stock price trends and provide actionable insights for investment strategies. We utilize historical daily trading data of Sinopec (600028.SH) from 20010808 to 20240618, including key features such as opening price, closing price, highest price, lowest price, and trading volume. The data undergoes thorough preprocessing, including missing value imputation, outlier detection, and normalization, ensuring that the neural network model can learn effectively from the time-series data. The model is built using a Multi-Layer Perceptron (MLP) architecture, with the Sigmoid activation function for binary classification of stock price movement trends. The model's performance is evaluated based on accuracy, with results indicating a 37% error rate. This study demonstrates the potential of neural networks to enhance stock prediction accuracy while emphasizing the importance of continuous model refinement and the incorporation of up-to-date market data for improved forecasting.

Keywords

Stock Prediction; Neural Networks; Timing

1. Introduction

The stock market is highly complex and volatile, making stock prediction crucial for investors. Traditional methods often fail due to market changes. Neural networks, similar to the human nervous system, can improve prediction ability through learning. This paper uses a neural network model for stock timing.

2. Machine Learning Algorithms

2.1 Logistic Regression

2.1.1 Introduction to Logistic Regression

Logistic Regression is a generalized linear regression model primarily used for binary classification problems, though it can also be extended to multi-class classification problems. The core principle is to map the output of linear regression to the range of (0,1) using the Sigmoid function, thereby converting continuous values into probabilities for classification tasks [1].

2.1.2 Introduction to the Logistic Regression Activation Function

Logistic Regression uses the Sigmoid function as its activation function to map the output of a linear combination to the range of (0,1), representing the probability of an event occurring. The expression for the Sigmoid function is:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

In this case, z is the output of the linear combination, i.e., $z = wTx + b$, where w is the weight, x is the input feature, and b is the bias term.

2.1.3 Introduction to the Loss Function

A Loss Function is a function used in machine learning and deep learning to measure the difference between the model's predicted results and the true outcomes. Its primary purpose is to evaluate the model's performance by calculating a numerical value and to improve the model by optimizing this value. The smaller the loss function, the closer the model's predicted results are to the true outcomes, indicating better model performance. The loss function is a non-negative real-valued function, typically represented as $L(Y, f(x))$, where Y represents the true value and $f(x)$ represents the model's predicted value. The goal of the loss function is to optimize the model parameters by minimizing this value, thereby improving the model's predictive accuracy [2].

2.1.4 Maximum Likelihood Estimation (MLE)

Maximum Likelihood Estimation (MLE) is a statistical method used to estimate the parameters of a probabilistic model. The goal is to find a set of parameters that makes the predicted probability distribution of the model as close as possible to the actual data distribution. By optimizing the objective function, logistic regression can uncover hidden features and perform classification based on probabilities.

2.2 SVM

2.2.1 Introduction to the SVM Principle

The basic principle of Support Vector Machine (SVM) is to find a hyperplane that separates data points of different classes, while maximizing the margin between the hyperplane and the closest data points. SVM is a binary classification model, and its fundamental model is a linear classifier that maximizes the margin in the feature space. The core idea of SVM is to find the optimal hyperplane by maximizing the margin, thereby enabling classification and regression analysis of the data [3].

2.2.2 Hyperplane and Optimal Hyperplane

The hyperplane is one of the core concepts in SVM classification and can be understood as a "decision boundary" that separates data into different classes. In a 2D space, it is a line; in a 3D space, it is a plane; and in higher-dimensional spaces, it is a hyperplane.

The optimal hyperplane is the hyperplane that maximizes the margin between the data points of different classes.

2.2.3 Kernel Function

A kernel function is a special function used in Support Vector Machines (SVM) for nonlinear mapping. The purpose of the kernel function is to map the input data from the original feature space to a higher-dimensional feature space, such that nonlinear problems in the original space become linearly separable or approximately linearly separable in the higher-dimensional space. Among the various kernel functions, the Gaussian kernel (also known as the Radial Basis Function, RBF) is commonly used to compute the similarity between vectors in SVM problems and to construct new features.

2.3 Neural Networks

2.3.1 Introduction to the Principle of Neural Networks

The basic principle of neural networks is to simulate the function of the human brain's neural system, where the connections and computations between multiple nodes (also known as neurons) enable the combination and output of nonlinear models. A neural network consists of multiple layers, including the input layer, hidden layers, and output layer. The input layer receives external data, the hidden layers process the data, and the output layer generates the final results [4].

2.3.2 Basic Components of Neural Networks

The basic component of a neural network is the neuron. Each neuron receives multiple input signals and generates an output signal through weighted summation and the activation function. The mathematical expression is: $y = f(w \cdot x + b)$, where x is the input signal, w is the weight, b is the bias, and f is the activation function. Common activation functions include Sigmoid, Tanh, and ReLU, each of which is suited for different applications.

2.3.3 Characteristics of Neural Networks

The core characteristics of neural networks include nonlinear functions, parameter weights, and the backpropagation algorithm. Nonlinear functions allow neural networks to model complex systems in the real world; parameter weights are continuously adjusted during training to better fit the data, and the backpropagation algorithm propagates errors backward

from the output layer to the input layer, gradually adjusting weights and biases to optimize network performance.

2.3.4 Activation Functions

(1) Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$

In cases where the output is not exactly 0 or 1, the Sigmoid function exhibits excellent nonlinearity. It is commonly used in binary classification problems, such as in logistic regression. The Sigmoid function smoothly projects the input to the range of (0,1), with the output centered around 0.5. However, it can suffer from the vanishing gradient problem, especially when the input values are too large or too small.

(2) Tanh Function

An improved version of the Sigmoid function, the Tanh function compresses the output values to the range of (-1, 1), with the output centered around 0, leading to faster convergence.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(3) ReLU Function

$$f(x) = \max(0, x)$$

In most cases, ReLU is the first choice, as it helps solve the vanishing gradient problem and is computationally faster than the Sigmoid and Tanh functions. When the input is negative, the function outputs 0, which can cause certain weights in the network to stop updating, making it difficult to train the model.

(4) Softmax Function

The Softmax function is typically used as the final layer of a neural network for multi-class classification problems. Essentially, it is an activation function that normalizes a vector of values into a probability distribution, where the sum of all probabilities equals 1.

3. Stock Timing Based on Neural Networks

3.1 Data Preparation

The historical daily trading data of Sinopec (600028.SH) from 20010808 to 20240618, including opening price, closing price, highest price, lowest price, trading volume, and other relevant information, is used for analysis.

3.2 Data Preprocessing

Introduction: Handling Missing Values, Outlier Detection, and Data Normalization Methods

We preprocessed the data by checking for missing values. For a small number of missing values, we used mean imputation to fill in the gaps. Outliers were identified and addressed, as they might result from data errors or extreme market events. We defined reasonable thresholds to detect outliers, considering data values that exceed the mean ± 3 standard deviations as outliers. These outliers were then corrected by replacing them with the mean or deleting them if necessary. Given the possibility of extreme market events in the stock market, special attention was given to removing abnormal fluctuations caused by such events, ensuring that these data points did not excessively influence the model's training.

Since different features have varying dimensions and value ranges, we standardized the data to help the neural network learn more effectively. We employed Z-score normalization, transforming the data into a distribution with a mean of 0 and a standard deviation of 1. This step helps eliminate dimensional differences between features and improves the training efficiency and convergence speed of the neural network. Considering that our data is time-series data, we particularly focused on the time dependency. During the normalization process, we applied adaptive methods to ensure that the model could fully leverage the time-series characteristics and capture the underlying patterns in stock prices [5].

3.3 Data Splitting

The dataset is divided into two main subsets: the training set and the test set. The training set, which makes up 70% of the total data, is used to train the neural network model. During training, the model learns to identify patterns in the data and

adjusts its internal parameters (weights and biases) accordingly. This set is also used for hyperparameter tuning, such as determining the number of layers, the number of neurons per layer, and the learning rate.

The remaining 30% of the data forms the test set, which is used to evaluate the model's performance after it has been trained. The test set represents unseen data, providing a realistic estimate of how well the model will perform on new, real-world data. This split ensures that the model is not overfitting to the training data and gives a better indication of its ability to generalize.

This 70/30 split is a common practice in machine learning, offering a good balance between training and evaluation. In some cases, other ratios or techniques like cross-validation could be considered to further validate the model's performance.

3.4 Features and Labels

The feature data calculation methods, such as the calculation of price-volume indicators, are based on financial knowledge and exploratory data analysis.

We constructed several derived features to reflect the trends in stock prices. These include the moving average (10-day moving average), MACD, Bollinger Bands, and moving weighted averages. Additionally, we calculated the Relative Strength Index (RSI) to measure overbought and oversold conditions in the stock price, as well as the Average True Range (ATR) to reflect the average real price volatility. These derived features may help the neural network capture underlying patterns in the stock market.

In addition to these features, we calculated the future 10-day price change as the label, which indicates the trend of stock price movement over the next 10 days. This provides a clear prediction target for the model. By incorporating these technical indicators, the neural network can better understand stock market price fluctuations and effectively improve the accuracy of stock timing strategies.

3.5 Model Construction

For the construction of the neural network model, we chose a Multi-Layer Perceptron (MLP) as the base architecture, consisting of an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer depends on the number of input features. The number of hidden layers and the number of neurons in each layer need to be determined through experimentation and tuning. Generally, the number of neurons in the hidden layers can start with a small value and be adjusted based on the model's performance on the validation set.

The output layer typically consists of a single neuron to predict the stock price movement trend, with an output of 1 indicating an upward movement and -1 indicating a downward or neutral movement. In the hidden layers, a suitable activation function is selected. Since this is a binary classification (predicting stock price movement), we use the Sigmoid function to map the output to the $[0, 1]$ range, representing the probability of an upward movement. The loss function used is binary cross-entropy, which measures the difference between the model's predicted results and the true labels. The neural network is trained by minimizing this loss function.

For optimization, we use the Adam optimizer, which is a commonly used adaptive learning rate optimization algorithm. Adam automatically adjusts the learning rate based on the gradients of different parameters, and it typically yields good results during training [6].

3.6 Model Evaluation

We evaluated the model using the validation set, with accuracy as the primary metric. Accuracy indicates the proportion of correct predictions relative to the total number of predictions. Our model produced an error rate of 37%, meaning it incorrectly predicted stock movements 37% of the time.

While accuracy provides a general sense of model performance, it does not fully capture the nuances of stock prediction. False positives and false negatives can have significant financial consequences, so additional metrics like precision, recall, and F1-score would offer a more comprehensive understanding of the model's strengths and weaknesses.

Given the complexity and volatility of the stock market, a 37% error rate is not entirely unexpected, but it points to areas for improvement. Future efforts will focus on refining the model, such as improving feature selection, tuning hyperparameters, and incorporating additional data to reduce the error rate and improve prediction accuracy.

4. Conclusion

Building a stock timing model based on neural networks requires a comprehensive approach, considering aspects like data processing, model architecture, training, and evaluation. Continuous optimization and refinement are essential to improving prediction accuracy. Moving forward, the model can be enhanced by regularly updating it with the latest stock market and

macroeconomic data. This will ensure the model adapts to changing market conditions and incorporates the most relevant information.

Preprocessing the new data, including handling missing values, detecting outliers, and normalizing, will help maintain the model's effectiveness. With these updates, the model can more accurately predict stock price trends, providing valuable insights for investment strategies such as buying, selling, or holding stocks. Over time, these continuous improvements should reduce the error rate and increase the model's utility for investors looking to make informed decisions based on stock market movements.

References

- [1] Zhou Z. Machine Learning. 1st ed. Tsinghua University Press; 2016.
- [2] Lopez de Prado M. Advances in Financial Machine Learning. 1st ed. Wiley; 2018.
- [3] Chan E. Algorithmic Trading: Winning Strategies and Their Rationale. 1st ed. Wiley; 2013.
- [4] Zhang W. Research on Multi-Factor Stock Selection and Timing Strategies Based on Industry [dissertation]. Zhongnan University of Economics and Law; 2022.
- [5] Hua Y. Research on Stock Timing Based on BP Neural Network Model [dissertation]. Shenyang University of Technology; 2019.
- [6] Hu Y. Stock Market Timing Model Based on Convolutional Neural Networks: A Case Study on the Shanghai Composite Index. *Financ Econ.* 2018;(04):71-4.
- [7] Lu M, Xu X. TRNN: An efficient time-series recurrent neural network for stock price prediction. *Information Sciences.* 2024;657:119951.
- [8] Wong S Y K, Chan J S K, Azizi L, et al. Time-varying neural network for stock return prediction. *Intell. Syst. Account. Finance Manag.* 2022;29(1):3-18.