**Frontiers in Business, Economics and Management**
ISSN: 2766-824X | Vol. 17, No. 3, 2024

# Stock Price Forecasting Based on the Gauss-Newton Method

**Quan Zhou**

Department of Mathmatics and Statistics, Donghua University, Shanghai, China
19129663622@qq.com

**Abstract:** The purpose of this study is to explore the stock price forecasting method based on the Gauss-Newton method. In this work, through a literature review, the existing stock forecasting methods were analyzed, and it was noted that there is a lack of attempts to construct nonlinear regression models using the Gauss-Newton method in existing research. The practical operation section provides a detailed description of data reading, fitting of the logarithmic regression model, the construction process of the exponential regression model and the power function regression model, as well as parameter estimation and forecasting using the Gauss-Newton iterative method. By comparing the mean square error and graphical analysis of different models, the optimal model is determined. The significance of this study lies in providing investors with a new tool for understanding market fluctuations, which helps in formulating more scientific risk management strategies.

**Keywords:** Stock price forecasting, Gauss-Newton method, Nonlinear regression, Risk management, Market volatility.

## 1. Introduction

In the current economic situation in China, the stock market is facing some challenges and opportunities. In recent years, due to factors such as the pandemic, the Chinese stock market has experienced several ups and downs, with significant overall fluctuations. However, as the economic situation warms up, the number of investors gradually increases, and the market's activity level has also improved. In addition, the government is also increasing its regulatory efforts on the capital market to enhance the market's transparency and stability. In the short term, the stock market may fluctuate due to the impact of macroeconomic fluctuations and changes in the external environment. But in the long run, the continuous growth trend of China's economy will provide momentum for the growth of the stock market.

Stocks hold an indispensable importance in today's society. For a country, stock trading activities can stimulate the activity and development of the capital market, which helps attract high-quality companies to go public and raise funds, further promoting the country's economic development. However, the fluctuations in the stock market can also lead to financial risks and economic instability, so the country needs to strengthen regulatory efforts to guard against systemic risks. For enterprises, the stock market is an important channel for financing. By issuing stocks, companies can raise funds for business expansion, technological innovation, and other activities. Moreover, the stock price performance after a company goes public will directly affect the company's overall valuation and business development. Therefore, companies also need to enhance risk management. For individual investors, stock trading can provide a certain level of investment returns. However, stock trading also carries market risks and requires investors to have a certain level of financial knowledge and awareness of risks, as well as reasonable investment strategies.

## 2. Literature Review and Significance of the Study

### 2.1. Literature Review

There are various methods and tools for stock prediction. Ma Jingjing from Sichuan College of Finance and Economics (2024) used K-line chart analysis to analyze the overall trend of stock price data, and then used logistic regression and random forest, two classification models, to predict the rise and fall of stock prices and evaluate their accuracy. Guo Hui Ting and others from Beijing University of Chemical Technology (2024) designed an ensemble prediction algorithm based on deep learning to predict future stock prices. Xu Yifan from the University of Shanghai for Science and Technology (2024) used pattern recognition time series data analysis methods for stock prediction.

### 2.2. Significance of the Study

Due to the high volatility of stocks, many investors rely on luck when trading stocks. Stock prediction can help investors better understand the potential fluctuations of the market, thereby formulating corresponding risk management strategies, such as adjusting the allocation of investment portfolios to reduce potential losses. The aforementioned method, such as K-line chart analysis, uses a linear regression model. The linear regression model has few variables and is easy to fit. However, due to the large fluctuations in stock data, more complex models can be used for prediction, such as non-linear regression models. The Gaussian-Newton method used in this paper is a method for solving non-linear least squares, and it is widely used because of its concise format and small computational load. The followings are the advantages of the Gaussian-Newton method:

Fast convergence: The Gauss-Newton method is a second-order convergent method. Compared to first-order methods (such as the Newton-Raphson method), it has a faster convergence rate and can usually quickly converge to the minimum point of the function or the root of the equation.

Effective in local areas: The Gauss-Newton method is more

suitable for solving problems with a "good initial value" (where there is a minimum point or root near the initial value), as it is based on the first or second derivatives at the current iteration point for iteration. In the local area, the Gauss-Newton iterative method can quickly find a local optimal solution.

High applicability: The Gauss-Newton method can not only be used to solve non-linear equations or minimize non-linear functions but also applied to various fields such as parameter estimation, thus it has a wide range of application value.

# 3. Theoretical Foundation

## 3.1. Taylor Expansion

The Taylor expansion is a method of expressing a differentiable function f(x)at a point aa as an infinite series near that point. This expansion uses the increment of xx relative to the point aa to approximate the value of the function f(x). The general form of the Taylor series is as follows:

$$f(x) = f(a) + f'(a)(x-a) + f''(a)\,2!\,(x-a)2 + f'''(a)\,3!\,(x-a)3 + \cdots + f(n)(a)\,n!\,(x-a)\,n + Rn(x)$$

In this formula: Rn (x) is the remainder term, representing the error after the nth-order expansion. It measures the difference between the nnth-order expansion and the actual value of the function.

## 3.2. Jacobian Matrix

The Jacobian matrix is a matrix representation of the derivative of a multivariable function. If there is a vector-valued function f: Rn→Rm, which maps an n-dimensional real vector to an m-dimensional real vector, then the Jacobian matrix of this function is an m×nmatrix, the elements of which are defined as the partial derivatives of each component of the function with respect to each variable.

Specifically, if the i-th component of the function f is fi, and x is the n-dimensional variable vector (x1, x2, …, xn), then the Jacobian matrix Jcan be represented as:

$$\begin{bmatrix} \frac{\partial f1}{\partial x1} & \cdots & \frac{\partial f1}{\partial xn} \\ \vdots & \ddots & \vdots \\ \frac{\partial fm}{\partial x1} & \cdots & \frac{\partial fm}{\partial xn} \end{bmatrix}$$

## 3.3. Gauss-Newton Method

(1) Principle

The method uses the Taylor series expansion to approximate the non-linear regression model and then iteratively corrects the regression coefficients multiple times to make them approach the optimal regression coefficients of the non-linear regression model, ultimately minimizing the residual sum of squares of the original model. The intuitive idea is to first select a parameter value β for the parameter vector. If the function f $(X_t, \beta)$ has continuous second-order partial derivatives in the vicinity of $\beta_0$, then within the neighborhood of $\beta_0$, f $(X_t, \beta)$ can be approximated as linear, and thus can be solved approximately using the linear least squares method.

Assuming a model function: $Y_i = f(X_i, \beta) + \mu_i$, where f can also be represented as f $(x_1, x_2, ...x_p; \beta_1, \beta_2, ..., \beta_n)$, with a total of m observation points, each point depending on p variables. Let $Q=\sum_{i=1}^{m}[f(X_i,\beta) - Y_i]^2$, with the initial value of β being $\beta_0$, expand f at $\beta_0$ using Taylor expansion.

$$F(X_i, \beta) = f(X_i, \beta_0) + \sum_{i=1}^{m}[\frac{df}{d\beta_k}]_{\beta_0}(\beta_k - \beta_0)$$

Substitute into the model function and find $\beta_1$ that minimizes Q.

Use $\beta_1$ as the new iteration value and continue the iteration with this method.

$$\beta_{k+1} - \beta_k = (J(\beta_k)^T J(\beta_k))^{-1} J(\beta_k)^T Q(\beta_k)$$

Stop the iteration when $\|\beta_{k+1} - \beta_k\| < \varepsilon$, where ε is a small positive threshold value.

(2) Fields of Application

Data Fitting: In science and engineering, it is used for fitting experimental data to theoretical models.

Machine Learning: In certain types of machine learning algorithms, such as parameter estimation in Support Vector Machines (SVM).

Signal Processing: In signal estimation and system identification, it is used for parameter estimation.

Computer Vision: In 3D reconstruction and camera calibration, it is used for solving geometric parameters.

# 4. Practical Operation

The stock prediction in this study is implemented by Python code, and the followings will provide a detailed introduction to the code and results.

## 4.1. Data Acquisition

The data used consists of daily OHLC data for Ping An Bank's stock, ranging from January 4, 2005, to June 20, 2024. It includes the date, opening price, closing price, highest price, and lowest price.

## 4.2. Data Processing

(1) Read Data

df = pd.read_csv("\Users\zhouquan\Desktop\data.csv", index_col=0)

**Table 1.** Data for Ping An Bank's Stock

| datetime | open | close | low | high |
|---|---|---|---|---|
| 2005/1/4 | 1.32 | 1.31 | 1.29 | 1.32 |
| 2005/1/5 | 1.31 | 1.29 | 1.27 | 1.31 |
| 2005/1/6 | 1.30 | 1.31 | 1.29 | 1.32 |
| 2005/1/7 | 1.32 | 1.30 | 1.29 | 1.32 |
| 2005/1/10 | 1.30 | 1.32 | 1.28 | 1.32 |
| ... | ... | ... | ... | ... |
| 2024/6/14 | 10.15 | 10.18 | 9.99 | 10.23 |
| 2024/6/17 | 10.13 | 10.10 | 10.08 | 10.16 |
| 2024/6/18 | 10.11 | 10.08 | 10.04 | 10.16 |
| 2024/6/19 | 10.10 | 10.15 | 10.08 | 10.21 |
| 2024/6/20 | 10.16 | 10.05 | 10.05 | 10.17 |

(2) Select the 'open', 'low', 'high' columns as features X from the data frame df, and choose the 'close' column as the target variable y. Take the logarithm of the features X to obtain X_log. Then, create an all-zero array as the initial parameters params_init, including the intercept.

```
X = df[['open', 'low', 'high']].values
y = df['close'].values
X_log = np.log(X)
params_init = np.zeros(X_log.shape[1] + 1)
```

## 4.3. Model Building and Prediction

(1) Logarithmic Model

Define the `log_model` function to calculate the predicted values of the logarithmic model. Define the `residuals` function to calculate the residuals. Define the `gauss_newton` function to estimate the parameters using the Gauss-Newton iterative method. Call the `gauss_newton` function to estimate the parameters `params_estimated_gn_custom`. Use the estimated parameters to predict `X_log` to get `y_pred_gn_custom`. Calculate the mean squared error `mse_gn_custom` between the predicted results `y_pred_gn_custom` and the actual values `y`. Output the estimated parameters `params_estimated_gn_custom` and the mean squared error `mse_gn_custom`.

```
def log_model(params, X):
    return params[0] + np.dot(X, params[1:])

def residuals(params, X, y):
    return y - log_model(params, X)
def gauss_newton(X, y, params_init, max_iter=100, tol=1e-6):
    params = params_init
    for i in range(max_iter):
            y_pred = log_model(params, X)
        residuals = y - y_pred
        J = np.hstack((np.ones((X.shape[0], 1)), X))
        delta = np.linalg.inv(J.T @ J) @ J.T @ residuals
        params_new = params + delta
        if np.linalg.norm(delta) < tol:
            break
        params = params_new
    return params
params_estimated_gn_custom = gauss_newton(X_log, y, params_init)
y_pred_gn_custom = log_model(params_estimated_gn_custom, X_log)
mse_gn_custom = mean_squared_error(y, y_pred_gn_custom)
params_estimated_gn_custom, mse_gn_custom
```

(2) Exponential Model

Take the logarithm of the target variable y to obtain y_log. Define the exponential model `exp_model` and the linearized model `linearized_model`. Define the residuals function `residuals`, which is used to calculate the residuals between the observed values and the predicted values. Define the Gauss-Newton iterative method function `gauss_newton`, which is used to estimate the parameters through iterative optimization. Use the Gauss-Newton iterative method to estimate the parameters `params_estimated_gn_custom`. Convert the estimated parameters back to the original parameters. Use the estimated parameters to make predictions and obtain `y_pred_gn_custom`. Calculate the mean squared error `mse_gn_custom` between the predicted values and the actual values. Output the estimated parameters `params_estimated_gn_custom` and the mean squared error `mse_gn_custom`.

```
y_log = np.log(y)
def exp_model(params, X):
    return params[0] * np.exp(np.dot(X, params[1:]))
def linearized_model(params, X):
    return params[0] + np.dot(X, params[1:])
def residuals(params, X, y):
    return y - linearized_model(params, X)
def gauss_newton(X, y, params_init, max_iter=100, tol=1e-6):
    params = params_init
    for i in range(max_iter):
            y_pred = linearized_model(params, X)
            residuals = y - y_pred
            J = np.hstack((np.ones((X.shape[0], 1)), X))
        delta = np.linalg.inv(J.T @ J) @ J.T @ residuals
        params_new = params + delta
        if np.linalg.norm(delta) < tol:
            break
        params = params_new

    return params
params_estimated_gn_custom = gauss_newton(X, y_log, params_init)
params_estimated_gn_custom[0] = np.exp(params_estimated_gn_custom[0])
y_pred_gn_custom = exp_model(params_estimated_gn_custom, X)
mse_gn_custom = mean_squared_error(y, y_pred_gn_custom)
params_estimated_gn_custom, mse_gn_custom
```

(3) Power Function Model

Take the logarithm of the features and the target variable, define the linearized power function model `power_model`.

```
X_log = np.log(X)
y_log = np.log(y)
def power_model(params, X):
return params[0] + np.dot(X, params[1:])
```

The rest of the code, excluding the model names, is consistent with the code for the exponential model.

## 4.4. Graphical Observation

(1) Logarithmic Model

Draw a line chart of the actual and predicted closing prices. First, create a chart and axes, then plot the actual closing prices in red dashed line and add the label 'True'. Next, plot the predicted values in blue solid line and add the label 'Pred'. Set the chart title to 'True vs Pred', the X-axis label to 'Datetime', and the Y-axis label to 'Close Price'. Finally, add a legend and adjust the layout of the chart.

```
fig, ax = plt.subplots(figsize=(12, 8))
df.close.plot(ax=ax, c='r', linestyle=':', label='True')
ax.plot(y_pred_gn_custom, 'b-', label='Pred')
ax.set_title('True vs Pred')
ax.set_xlabel('Datetime')
ax.set_ylabel('Close Price')
ax.legend(loc='best')
fig.tight_layout()
```

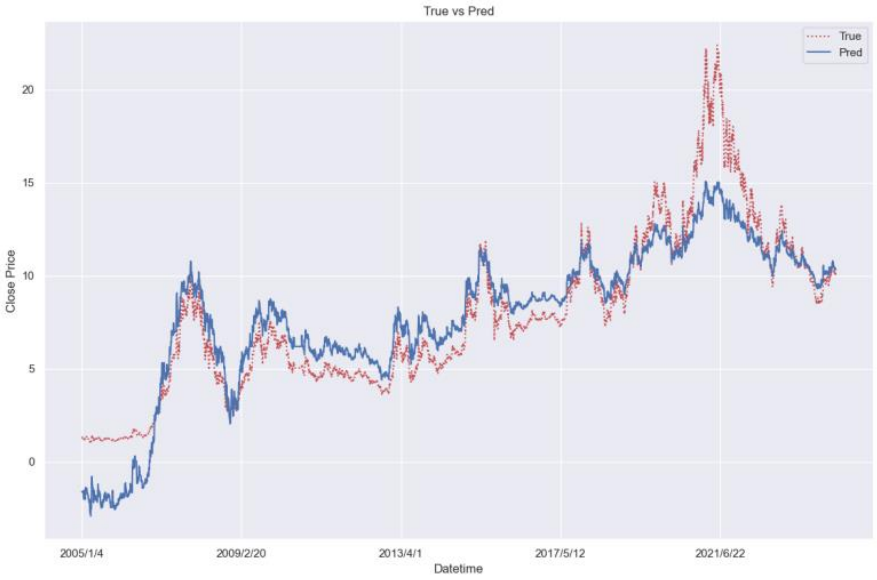


**Figure 1.** Plot for Logarithmic Model

(2) Exponential Model for the logarithmic model.
The code for plotting is consistent with the plotting code

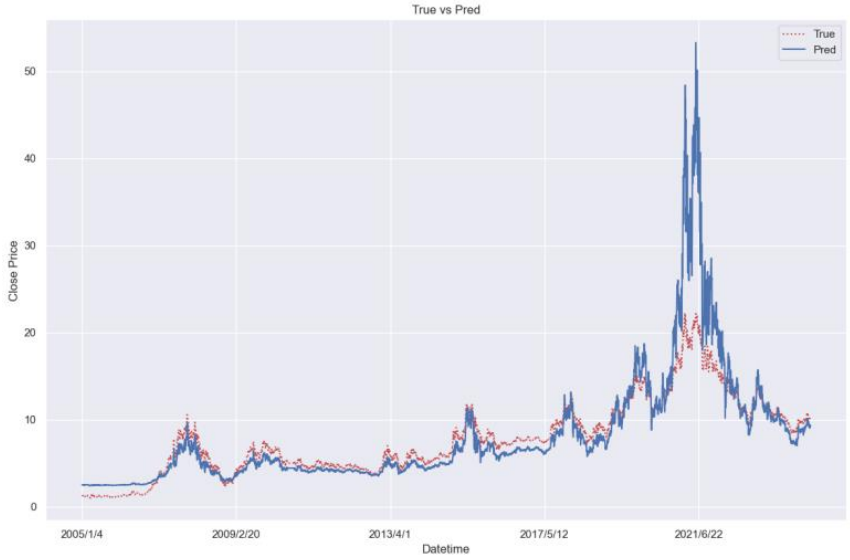


**Figure 2.** Plot for Exponential Model

(3) Power Function Model for the logarithmic model.
The code for plotting is consistent with the plotting code

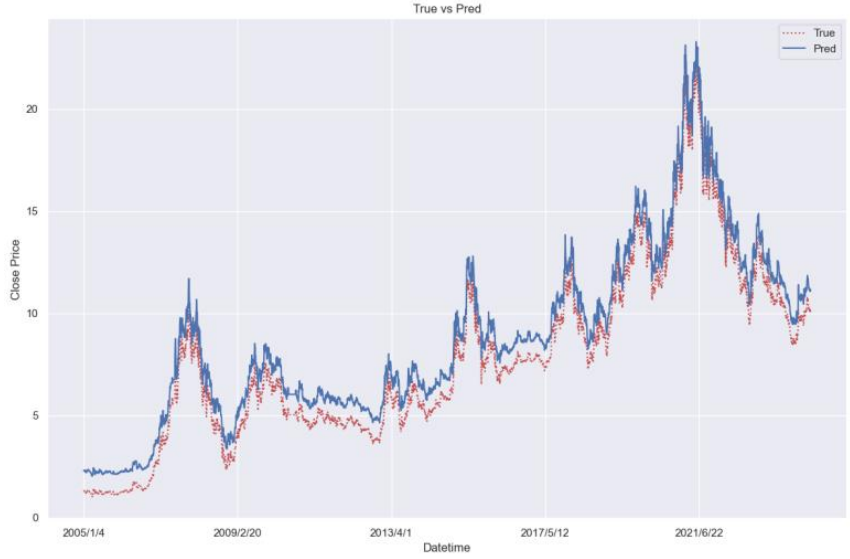


**Figure 3.** Plot for Power Function Model

## 5. Research Results and Summary

This work proposes a non-linear model for predicting future stock closing prices based on historical stock data, including opening price, closing price, highest price, lowest price, and trading volume. Running the code yields a mean squared error (MSE) of 3.033 for the logarithmic model, 11.105 for the exponential model, and 1.020 for the power function model. It is evident that the power function model has the smallest MSE, indicating the most accurate prediction. Investors can construct the power function model and estimate parameters using the Gauss-Newton method to forecast future stock prices, thereby formulating rational investment strategies. In future research, the model can also be applied to actual stock market data to test its performance under various market conditions.

## References


[1] Brock, William, Josef Lakonishok, and Blake LeBaron. (1992). "Simple Technical Trading Rules and the Stochastic Properties of Stock Returns". Journal of Finance, 47(5), 1731-1764.

[2] Fama, Eugene F., and Kenneth R. French. (1988). "Dividend Yields and Expected Stock Returns". Journal of Financial Economics, 22(3), 3-25.

[3] MA Jingjing. (2024). "Regression and Classification in Stock Price Trend Forecasting". Computer Knowledge and Technology, 20(12), 1009-3044.

[4] GUO Huiting, CHANG Yanzhen, LI Liang, XU Yongli. (2024). "Stock Price Prediction Based on Integrated Deep Learning". Modern Information Technology, 8(9), 2096-4706.

[5] WU Ling, LIU Zhong, LU Faxing. (2003). "Global Convergence Gauss-Newton Method Applied to Nonlinear Least Square Estimation in Target Location". Fire Control Radar Technology, 32, 1008-8652.