How can stochastic differential equations be applied to enhance investment returns through stock price prediction?

Enhancing Investment Returns through Stock Price Prediction Using the Generalized Geometric Brownian Motion Model

Page Count: 18

Contents

1 Introduction

1.1 Background

Quantitative finance is the field of applied mathematics concerned with financial markets. It involves the application of mathematical models to maximize profits. Stock price prediction plays a crucial role in quantitative investment strategies. Accurate forecasts can lead to enhanced investment returns and more effective risk management strategies. Over the years, various approaches to stock prediction have emerged. Stochastic Differential Equation (SDE), though not the latest, is an important model in quantitative finance. SDE provides a mathematical framework for describing the random behavior of financial variables over time, accounting for both regular trends and random fluctuations.

This investigation primarily focuses on two core topics from the Mathematics AA HL syllabus: Topic 4: Statistics and Probability, and Topic 5: Calculus. Statistics and probability provide the framework for analyzing random fluctuations in stock prices, while calculus is essential for modeling continuous-time processes.

As a student with a keen interest in finance, I've always been drawn to the intersection of mathematics and economics. This led me to choose both Mathematics AA HL and Economics HL for my IB curriculum, hoping to build a strong foundation for my future career. Although I had initially planned to participate in business competitions, time constraints made it challenging to commit the necessary time. Instead, I decided to channel my passion into this IA.

Also, I choose NVIDIA's stock for a reason. Began in September 2020 when, as a middle school student, I was researching graphics cards for my computer. Learning about NVIDIA's upcoming 30 series GPUs, I correctly predicted a rise in their stock price but lacked the money to invest. Later, in early 2023, the emergence of generative AI, specifically ChatGPT 3.5, revolutionized the whole world. Not only this, it also sparked a surge in demand for GPUs for AI training. Again, I anticipated NVIDIA's stock would increase, and it did. However, again, lack of money prevented me from acting on this.

This IA offers me an opportunity to explore what could have been. I've created a scenario: if I had \$ 10,000 to invest in NVIDIA stock from June 8, 2023, to June 7, 2024, how could I maximize returns without foreknowledge of future prices?

1.2 Aim

The aim of this investigation is to maximize investment returns in NVIDIA's stock market given that an initial investment of \$ 10,000, the investment period is from June 8, 2023, to June 7, 2024, and future stock prices are unknown at the time of investment.

To achieve this aim, four tasks have to be done:

- Develop a prediction model for forecasting NVIDIA's stock prices over the entire investment period (GGBM model).
- Create a dynamic prediction model that adapts to new information as it becomes available, simulating real-world trading conditions where future prices are unknown (GGBM model).
- Design a feasible trading strategy based on dynamic predictions, including both risk management and position sizing techniques.
- Compare the performance of the GGBM-based strategy against other methods, specifically the Moving Average (MA) model, to evaluate its effectiveness.

1.3 Research Question

How can stochastic differential equations be applied to enhance investment returns through stock price prediction?

2 Model Derivation

2.1 Arithmetic Brownian Motion

When used to predict stock price, SDE has two mean assumptions:

1. The stock prices follow a continuous and differentiable path, without abrupt jumps or discontinuities.
2. Markets are efficient, with current prices reflecting all available information and future price movements being driven by new, random information.

The arithmetic Brownian motion model is the simplest form of stochastic differential equations. It can be written as $dX(t) = \mu\, dt + \sigma\, dW(t)$. Substituting $X(t)$ with the stock price at time t, S_t, it can be used for prediction (Musiela & Rutkowski, 2006):

$$dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW_t$$

Where S_t is the stock price at time t, $\mu(S_t, t)$ is the drift term, representing the expected rate of change of the price, $\sigma(S_t, t)$ is the volatility term, indicating the magnitude of price fluctuations, and W_t is the standard Wiener process (Brownian motion).

2.2 Generalized Geometric Brownian Motion Model

When used for stock price prediction, SDE often presents challenges: the SDE does not take into account the non-negativity of stock prices or the long-term growth trend; the SDE fails to consider that stock prices tend to revert to their long-term mean.

The generalized geometric Brownian motion model(GGBM) is better suited for simulating long-term stock price trends, particularly for stocks expected to experience sustained growth. This model allows for unlimited price growth, which aligns with the long-term performance of many stocks (Musiela & Rutkowski, 2006). It is shown as follows:

$$dS_t = \mu(t)S_t dt + \sigma(t)S_t dW_t$$

Where $\mu(t)$ and $\sigma(t)$ are now functions of time, independent of S_t.

Moving S_t from the right side of the equation to the left side gives:

$$\frac{dS_t}{S_t} = \mu(t)dt + \sigma(t)dW_t$$

Since $\frac{d\ln x}{x} = \frac{1}{x}$, this equation can be written as:

$$\frac{dS_t}{S_t} = d(\ln S_t) = dp_t = \mu(t)dt + \sigma(t)dW_t$$

$$dp_t = \mu(t)dt + \sigma(t)dW_t$$

Where $p_t = \ln S_t$ is the log price. The reason for using the log prices is simple: in financial markets, relative changes in price, instead of absolute changes in price, are often emphasized. Log prices can provide a more accurate representation of these relative changes. For instance, if a stock price increases from \$ 100 to \$ 110, the absolute change is \$ 10; however, the relative change is 10%, represented as $\ln(110) - \ln(100)$. In contrast, if another stock rises from \$ 1,000 to \$ 1,010, the absolute change is also \$ 10, but the relative change is only 1%, expressed as $\ln(1010) - \ln(1000)$. Log prices hence allow for a consistent framework to analyze relative changes across different price levels.

This equation can be seen as describing every change in the stock price (log prices) of NVIDIA, as dp_t is the increment of log stock price at an instant. The right part of this equation states the two components of dp_t: predictable trends and random fluctuations.

The first part of the formula, $\mu(t)dt$, represents the expected growth rate of the stock price, closely tied to market trends, company performance, and other factors. For instance, when Nvidia releases its new 40 series graphics cards, the market may anticipate that this product will significantly boost the company's revenue, leading to an upward push in the stock price. The current log stock price p_t is influenced by these expectations. If the price is below market expectations, there will be incentives for the price to adjust upward. Conversely, if the market reacts poorly after the release, the price may decrease.

The second part of the formula, $\sigma(t)dW_t$, considers market volatility. Here, $\sigma(t)$ is a time-varying volatility parameter that reflects market uncertainty, which may change due to economic data, competitor actions (such as new products from AMD), industry news (like semiconductor shortages), etc. The term dW_t represents the random component in the market, which can be seen as noise. Mathematically, dW_t is the random increment of Brownian motion over an infinitesimal time interval dt, following a normal distribution with mean 0 and variance dt, or $dW_t \sim \mathcal{N}(0, dt)$. This part of the equation indicates that Nvidia's stock price is influenced not only by the target price but also by market fluctuations. For example, if a news report coincides with the release of the new graphics cards and announces that the manufacturing process is significantly flawed, it could lead to significant short-term price volatility. In other words, the stock price of it may change more drastically and dramatically.

In financial markets, the concept of mean reversion posits that asset prices or returns tend to revert to their long-term average levels. This characteristic can be captured by introducing a mean reversion term, expressed as (Predictable, 1995):

$$\mu(t) = -\kappa(p_t - \theta(t))$$

Where $\kappa > 0$ is the speed of mean reversion, determining how quickly the price returns to its long-term mean, and $\theta(t)$ is the long-term mean at time t, reflecting the historical average stock price. This mean reversion term implies that when the stock price exceeds the long-term mean, $\mu(t)$ will be negative, causing the price to move downward. Conversely, when the price falls below the long-term mean, $\mu(t)$ will be positive, pushing the price upward.

For example, suppose $\theta(t) = 4.6$ (corresponding to a price of approximately 100), and currently $\ln S_t = 4.8$ (corresponding to a price of approximately 121):

$$(\theta(t) - \ln S_t) = 4.6 - 4.8 = -0.2$$

When multiplied by the positive κ, this negative value will generate a downward force, causing the price to trend back down toward the long-term mean.

Conversely, if currently $\ln S_t = 4.4$ (corresponding to a price of approximately 81):

$$(\theta(t) - \ln S_t) = 4.6 - 4.4 = 0.2$$

When multiplied by the positive κ, this positive value will generate an upward force, causing the price to trend back up toward the long-term mean.

The size of κ affects the speed of mean reversion. For example, if $\kappa = 0.1$, the reversion will be relatively slow; if $\kappa = 1$, the reversion will be faster.

Substituting $\mu(t)$ with $-\kappa(p_t - \theta(t))$

$$dp_t = -\kappa(p_t - \theta(t))dt + \sigma(t)dW_t$$

The equation requires Itô integrals (Oksendal, 2013) to derive analytical solutions, a complex process that may impede subsequent calculations. Consequently, the Euler-Maruyama method is used to discretize the stochastic differential equation, thereby facilitating obtaining numerical solutions.

The Euler-Maruyama method states that for a given SDE

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t$$

The iterative is

$$X_{n+1} = X_n + a(t_n, X_n)h + b(t_n, X_n)\Delta W_n$$

Where ΔW_n is the increment of Brownian motion, usually assumed to be $\Delta W_n \sim \mathcal{N}(0, h)$. h is the time step size, $h = \frac{T}{N}$. Where T is the final time to approximate the solution, and N is the number of time steps. Larger h will reduce the computational cost but may decrease the approximation accuracy, and vice versa. In this case, $h = \Delta t = 1$day, as both are indicating time increment.

Applying this method to the stock-predicting equation

$$p_{t+\Delta t} - p_t = -\kappa(p_t - \theta(t))\Delta t + \sigma(t)\Delta W_n$$

To generate random increments in numerical simulations, we can use a random variable ϵ_t drawn from the standard normal distribution $N(0, 1)$, and then scale it as $\Delta W_t = \sqrt{\Delta t} \cdot \epsilon_t$. The reason for using $\sqrt{\Delta t}$ is that, since the variance of ϵ_t is 1, multiplying it by $\sqrt{\Delta t}$ results in a random variable with a variance of Δt:

$$\text{Var}(\sqrt{\Delta t} \cdot \epsilon_t) = (\sqrt{\Delta t})^2 \cdot \text{Var}(\epsilon_t) = \Delta t \cdot 1 = \Delta t$$

Therefore, the final equation becomes:

$$p_{t+\Delta t} - p_t = -\kappa(p_t - \theta(t))\Delta t + \sigma(t)\sqrt{\Delta t} \cdot \epsilon_t$$

3 Model Application

3.1 Data Collection and Pre-processing

The NVIDIA stock price data for the entire year, encompassing 252 trading days, was sourced from Yahoo Finance. The downloaded dataset includes the open, high, low, close, and adjusted close prices, utilizing daily adjusted closing prices. This data has been organized into a CSV file to facilitate future analysis.

The stock price depicted in the figure1 demonstrates a clear overall upward trend over the course of the year, which aligns with the random walk characteristic of the model. This trend can be represented through the drift term, reflecting the market's long-term growth potential. Furthermore, the stock price exhibits volatility, a factor that is also incorporated within the model. Therefore, the stock price data for NVIDIA over the past year is suitable for predictive analysis.

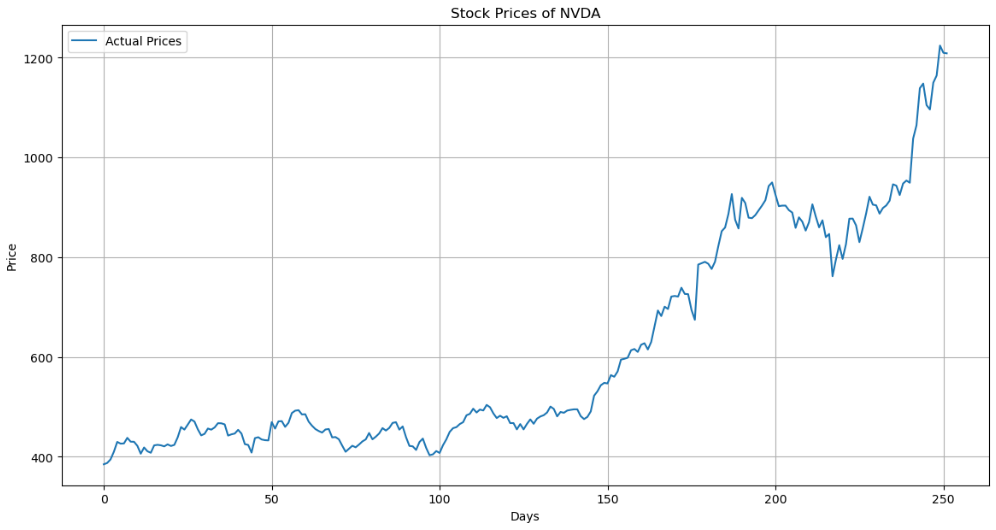

Figure 1: The Actual Price of NVIDIA's stock price from 2023.6.8 to 2024.6.7

3.2 Parameter Estimation

In traditional SDE-based stock price predicting models, parameters are usually fixed. Specifically, the parameters μ and σ are not subject to change. They are calculated as a fixed mean and variance in a given time period. This model considers a dynamic estimation of the two parameters.

3.2.1 Estimation of κ

The logarithmic regression method is employed to estimate the parameter κ based on the initial 20 price data points. This approach is widely utilized in financial time series analysis (Shumway, Stoffer, & Stoffer, 2000) and involves fitting an AR(1) model:

$$p_t = c + \phi p_{t-1} + \xi_t$$

Where $y_t = \ln(S_t)$ is the logarithm of the price at time t, c is a constant term, ϕ is the autoregressive coefficient, and ξ_t is white noise.

This formula shows the idea that the current value is primarily determined by the previous day's value, along with a fixed baseline and some random fluctuations. That is, it is regressing against itself (auto-regression). The extent to which the previous day's value influences today's value is controlled by the coefficient ϕ. Additionally, some unpredictable factors (random noise) may cause today's value to differ from what is expected.

By estimating ϕ, the mean reversion speed κ can be derived as follows:

$$\kappa = -\ln(\phi) \cdot N$$

where N is the number of trading days in a year. This method effectively captures the mean reversion properties of price series.

When prices deviate from their average, they tend to revert to the mean at a rate that decreases over time (mean reversion), following an exponential pattern. A logarithmic transformation is used to linearize this exponential relationship, making it easier to understand the speed of mean reversion, which is represented by κ.

In the AR(1) model, the parameter ϕ typically falls between -1 and 1, ensuring the process is stable. By applying the transformation $-\ln(\phi)$, κ is always positive, which aligns with the concept of the speed at which prices return to the mean.

Multiplying by N converts the daily mean reversion rate into an annualized rate, giving the model's parameters a clear economic interpretation. This approach also maintains consistency between discrete and continuous models.

3.2.2 Estimation of θ and σ

$\theta(t)$ is estimated based on the arithmetic mean of the logarithmic prices of historical trades, effectively capturing market dynamics and the time-varying characteristics of the long-term mean. This is formally expressed as:

$$\theta = \frac{1}{n} \sum_{i=1}^{n} \ln(S_i)$$

where n is the number of historical trading days.

Similarly, $\sigma(t)$ is derived from the standard deviation of the logarithmic returns over historical trading days, providing a measure of short-term market volatility. The calculation is given by:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (r_i - \bar{r})^2}$$

where $r_i = \ln(S_{i+1}/S_i)$ is the logarithmic return from day i to day $i+1$, and $\bar{r}$ is the mean logarithmic return.

Two methodologies are considered for parameter estimation:

- **Full Historical Data Method:** This approach utilizes all available data from the initial date to the current date for parameter estimation.
- **Rolling Window Method:** This approach uses a fixed window of 60 trading days for parameter estimation, thereby enhancing the model's sensitivity to recent market dynamics.

By including the entire historical dataset, the full historical data method provides a comprehensive view of long-term trends and reduces the impact of short-term noise.

The rolling window method, in contrast to the full historical data method, demonstrates an ability to capture short-term market fluctuations. This is particularly evident in the case of technology stocks, such as NVIDIA, where market conditions can change rapidly.

Later analysis will determine which method is better for enhancing investment returns.

4 Prediction

4.1 Assessment Indices

The prediction performance is evaluated using two key indices: Mean Squared Error (MSE) and Mean Absolute Error (MAE). For both indices, the smaller the value, the better the model's performance.

The Mean Squared Error (MSE) is defined as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(S_i - \hat{S}_i)^2$$

Where $\hat{S}_i$ is the predicted price at day i and S_i is the actual price at day i. This index is particularly sensitive to larger errors because it squares the deviations.

On the other hand, the Mean Absolute Error (MAE) is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|S_i - \hat{S}_i|$$

MAE measures the average magnitude of the errors in the predictions, taking the absolute difference between the actual and predicted values. Unlike MSE, MAE treats all errors equally, making MAE less sensitive to outliers compared to MSE.

4.2 Prediction Result

The result of the prediction is shown in Figure 2. The dynamic prediction of θ and σ is shown in Figure 3.

Generally, both forecasting methods capture the general trend of stock prices, particularly in the early and mid-periods, where the predicted prices closely align with the actual prices. The sliding window approach exhibits superior performance, especially in the short term (e.g., 60 days). When using this method, MSE = 7308.39 and MAE = 56.44

In contrast, the full historical data method demonstrates stable performance across the prediction period but tends to underestimate the magnitude of price increases, particularly in the later stages. When using this method, MSE = 14033.04 and MAE = 79.26.

It is noteworthy that actual prices exhibit a significant rise after day 200. Although neither prediction method fully captures this sharp upward trend, the sliding window approach responds more promptly and effectively to the price increase. Consequently, the sliding window method shows a clear advantage in predictive performance.

4.3 Confidence Interval

To generate the forecast distribution of stock prices, the study employs the Monte Carlo method, simulating 1,000 price paths for each prediction day (Glasserman, 2013). The mean of these 1,000 paths is used as the point estimate. The 2.5th percentile is designated as the lower bound of the confidence interval, while the 97.5th percentile serves as the upper bound, thereby constructing a 95% confidence interval.

This approach effectively captures the uncertainty inherent in price forecasts, offering more comprehensive information for investment decision-making. A wider confidence interval

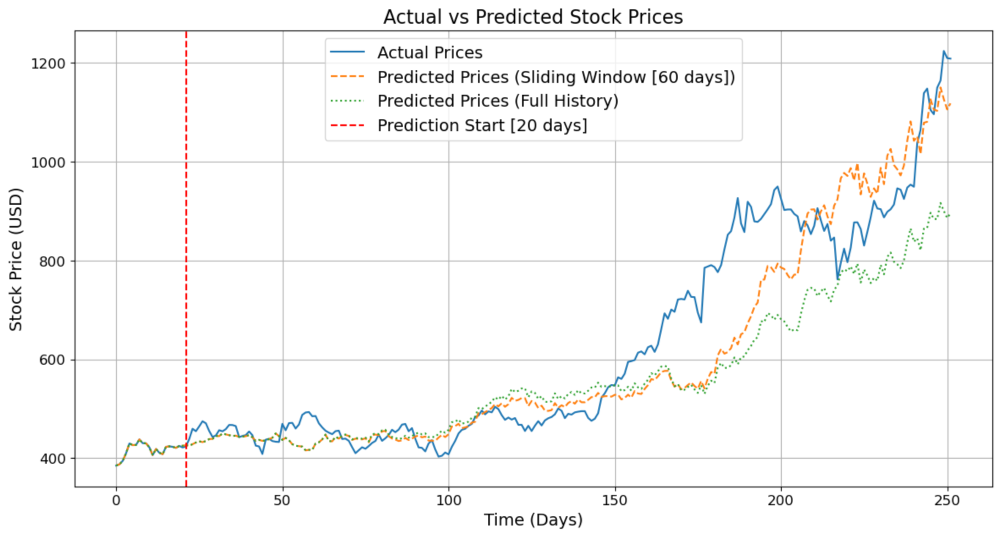

Figure 2: Predicted Stock Price

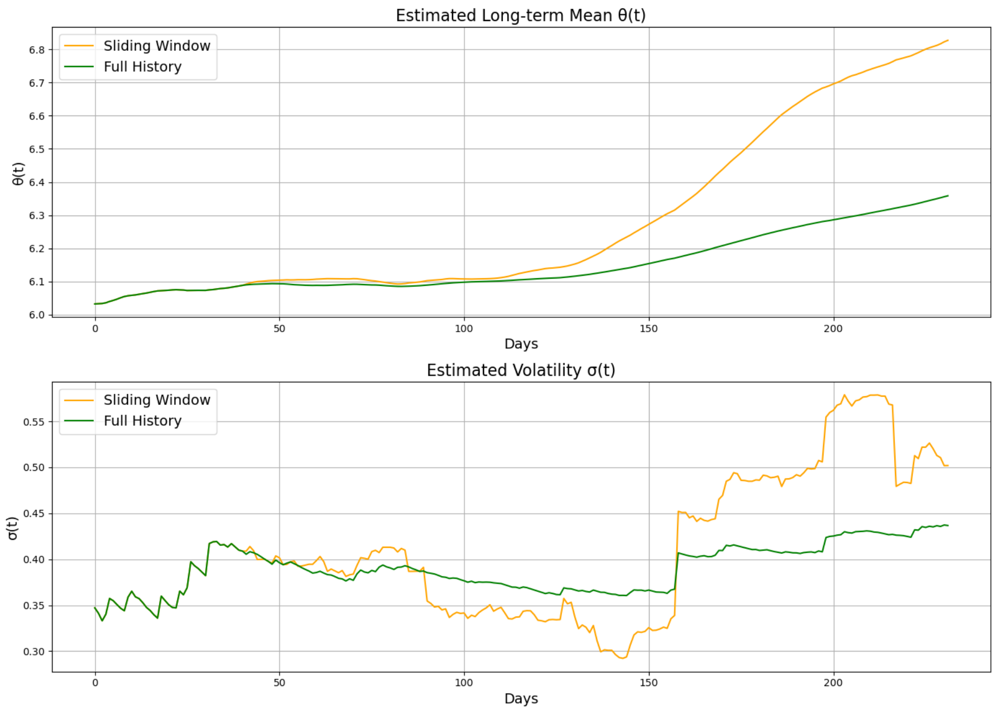

Figure 3: Dynamic Prediction of Thera and Sigma

indicates a higher degree of uncertainty in the prediction.

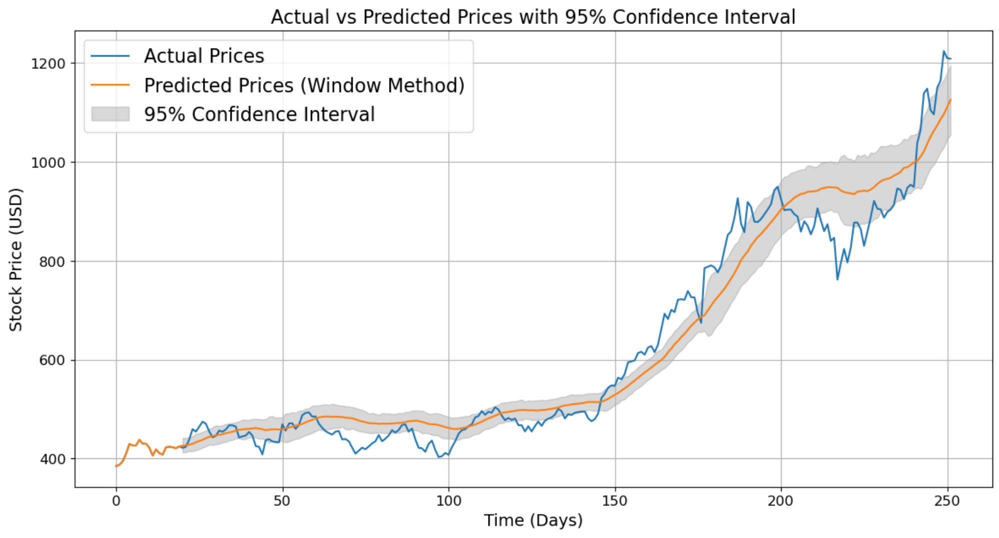

Figure 4: Confidence Interval of the Prediction

5 Enhancing Investment Returns

5.1 Trading Strategy

Monte Carlo simulations are used to generate multiple price paths, from which the expected price E_i and standard deviation σ_i at each time point i are calculated. Trading signals are then derived based on the relationship between the current price and future expectations according to the following criteria:

$$\begin{cases} S_i > E_{i+1} + k \cdot \sigma_{i+1} & \text{sell} \\ S_i < E_{i+1} - k \cdot \sigma_{i+1} & \text{buy} \\ E_{i+1} - k \cdot \sigma_{i+1} \leq S_i \leq E_{i+1} + k \cdot \sigma_{i+1} & \text{hold} \end{cases}$$

Here, $k = 1.5$ serves as the threshold. For a normal distribution, this corresponds to a probability coverage range of:

$$P(-1.5\sigma < X < 1.5\sigma) \approx 0.8664$$

In this trading strategy, the threshold implies that buy or sell signals are triggered only when the current price significantly deviates from the expected future price (beyond the

86.64% probability range). This approach is intended to reduce the occurrence of false trading signals, though it may result in missed trading opportunities.

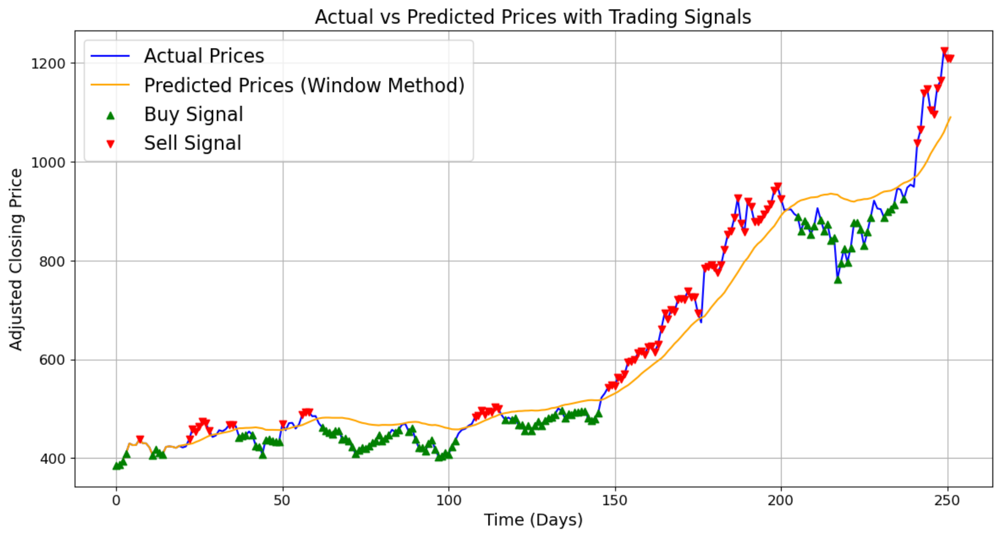

Figure 5: Buying and Selling Signals

5.2 Result Analysis

Finally, a backtest comparison was conducted between two mean-reversion strategies:

- **Buy:** Allocate all available cash to purchase the maximum number of shares (after accounting for transaction costs).
- **Sell:** Liquidate all holdings by selling the shares and converting them to cash (after accounting for transaction costs).

The strategy based on stochastic differential equations (SDE) has already been described earlier.

The alternative strategy, based on actual stock prices, uses the 20-day moving average as a mean reference. In this strategy, stocks are sold when the actual price exceeds the moving average and bought when the actual price falls below the moving average.

Both strategies start with an initial capital of \$ 10,000 and account for a fixed transaction cost of \$ 10 per trade. The performance of these strategies was simulated through backtesting, with daily account values being tracked. The final account values of both strategies

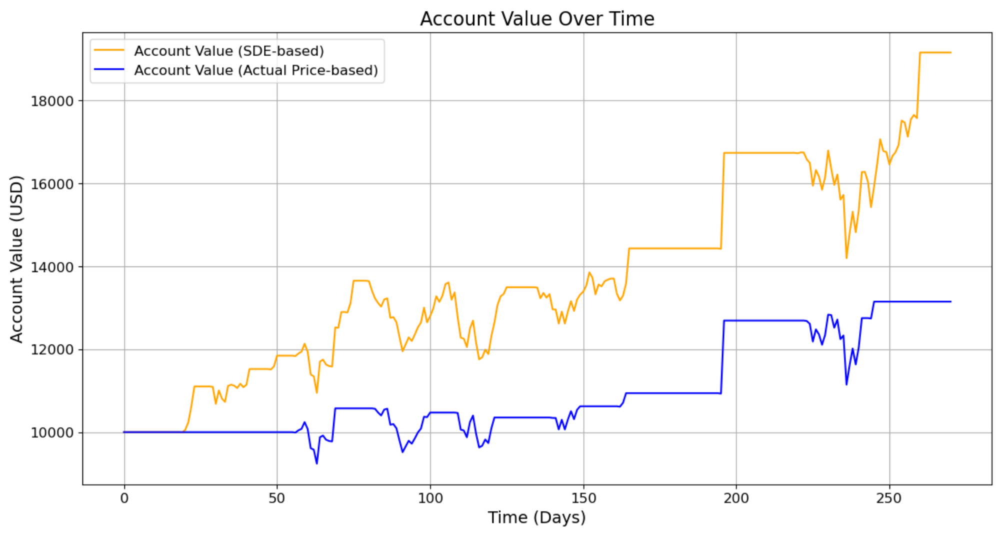

Figure 6: Backtesting of the Two Strategies

were then compared.

The results indicate that the mean-reversion strategy based on the SDE yields significantly higher returns than the strategy based on actual prices.

6 Worked Sample

The prediction of stock price is calculated using Python code (see appendix). This section provides a sample of how the code implements this function using a flow chart.

A sample of the day 150 will be used here. To predict the stock price on the 150th day, the model first estimates key parameters: $\kappa = 99.02$, $\theta = 6.46$, and $\sigma = 0.33$. Using these, the predicted price is 495.36 with a confidence interval of $[500.22, 544.77]$. The actual price was 548.19, slightly above the predicted range. A mean-reversion prediction gave a price of 498.02, showing the model's tendency to slightly underestimate the actual price, but still providing a reasonable forecast within the confidence interval.

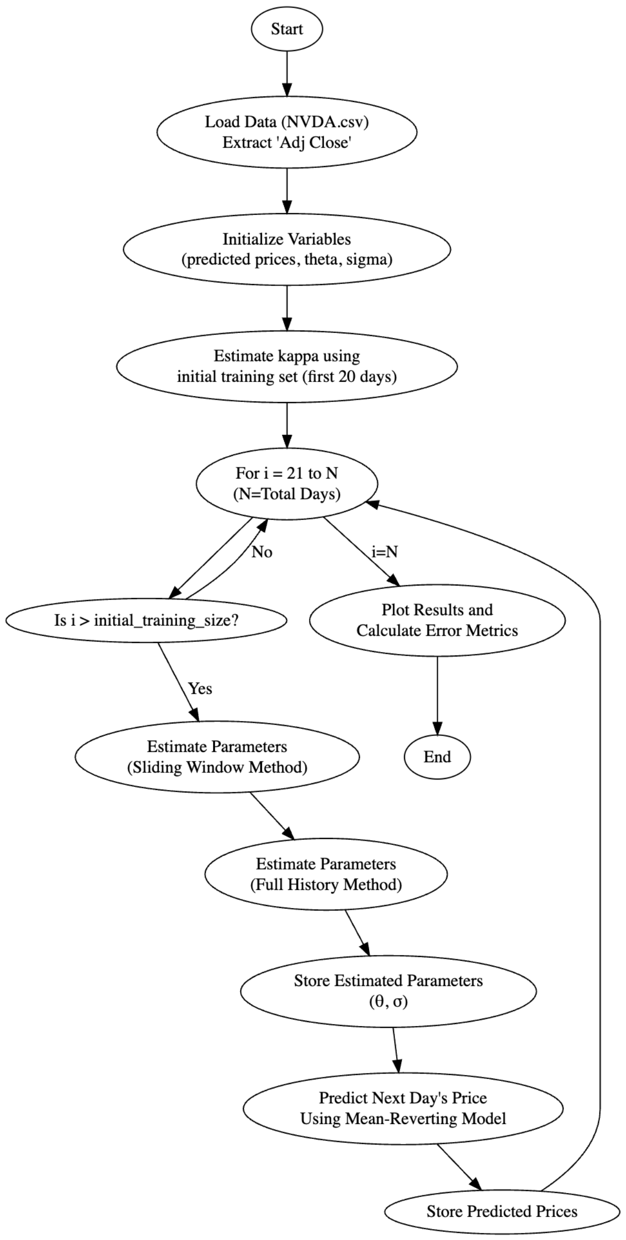

Figure 7: Worked Sample

7 Evaluation

7.1 Strength and Weakness

The strengths and weaknesses will be given based on the aims of this project. **Strength:**

1. **Development of a Prediction Model (GGBM Model):** An effective prediction model using the Generalized Geometric Brownian Motion (GGBM) model is developed. The model successfully integrates stochastic differential equations (SDEs) to capture both the long-term growth trends and the random fluctuations in stock prices.

2. **Creation of a Dynamic Prediction Model:** A dynamic prediction model that is able to adapt to new information is developed. The Euler-Maruyama method was used to discretize the SDEs, which reduces the computational costs of running this model. Also, using dynamic parameter estimation techniques, specifically the rolling window method, the model demonstrates strong adaptability to changing market conditions.

3. **Design of a Feasible Trading Strategy:** A feasible trading strategy based on dynamic predictions was designed. The strategy uses a statistically sound threshold ($k = 1.5$) for generating buy and sell signals. It ensures that trading decisions are made based on significant deviations from expected prices, which helps manage the risks.

4. **Comparison with Other Methods:** A comparison is made between the GGBM-based strategy and the Moving Average (MA) model. The project demonstrates that the GGBM-based strategy, particularly when using the rolling window method, is more robust than the MA model in terms of both predictive accuracy and returns.

However, despite the model's effectiveness, it also comes with some limitations. **Weakness:**

1. **Development of a Prediction Model (GGBM Model):** The GGBM model assumes a constant mean reversion speed κ throughout the investment period. Hence, the model may not fully capture the variations in market conditions, limiting the model's effectiveness in periods of extreme volatility.

2. **Creation of a Dynamic Prediction Model:** External factors such as macroeconomic indicators or news about NVIDIA's competitors are not included, which can significantly influence stock prices. This omission may be the cause of the decreased predictive power after day 150.

3. **Design of a Feasible Trading Strategy:** The trading strategy may be overly conservative due to the high threshold set for generating buy and sell signals. This may limit the investment return.

4. **Comparison with Other Methods:** Although proved more robust when compared with MA model, the GGBM model should also be compared to other models. Also, other models can be more robust, as the GGBM model is outdated to some extent. Other new methods, such as those that are based on neural networks, Long Short-Term Memory(LSTM)-based, and Transformer, for example, can be introduced. These models are more robust in predicting time series, of which stock prices are one example.

7.2 Extension

Still, based on the four aims, possible extensions are listed to improve the weaknesses

1. **Development of a Prediction Model (GGBM Model):** The mean reversion speed κ can be made time-varying to better capture different market conditions, hence improving the accuracy of predictions.

2. **Creation of a Dynamic Prediction Model:** Incorporating macroeconomic variables and company-specific news as additional factors can improve the robustness of the model. For example, the model could include an additional term, $\beta(t)Z(t)$, where $Z(t)$ represents a macroeconomic indicator (e.g., interest rate, GDP growth rate) or a company-specific factor (e.g., earnings reports, product launches), and $\beta(t)$ is a time-varying coefficient that measures the sensitivity of stock prices to this factor.

3. **Design of a Feasible Trading Strategy:** Three different approaches, with each corresponding to a distinct risk tolerance level- conservative, intermediate, and aggressive- can be introduced. The conservative strategy would maintain a higher threshold for buy and sell signals, thereby minimizing risk but potentially missing out on some opportunities, likewise for the other two cases.

4. **Comparison with Other Methods:** In addition to comparing the GGBM-based strategy, another valuable extension would be to include comparisons with other non-neural network-based predictive models (as they will definitely outperform the GGBM model). Examples could include traditional econometric models like the Autoregressive Integrated Moving Average (ARIMA) model.

A Code

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
from scipy import stats

# Global variable for trading days
TRADING_DAYS_PER_YEAR = 252

# Load the data
data = pd.read_csv('NVDA.csv')
prices = data['Adj␣Close'].values

# Initialize lists to store predicted prices and dynamic parameters
predicted_prices_window = [prices[0]]
predicted_prices_full = [prices[0]]
theta_values_window = []
sigma_values_window = []
theta_values_full = []
sigma_values_full = []

# Start with a small initial training set
initial_training_size = 20
window_size = 15  # Use a 60-day window for parameter estimation

# Estimate kappa
log_prices = np.log(prices[:initial_training_size])
X = log_prices[:-1].reshape(-1, 1)
y = log_prices[1:]
model = stats.linregress(X.flatten(), y.flatten())
phi = model.slope
kappa = -np.log(phi) * TRADING_DAYS_PER_YEAR

for i in range(1, len(prices)):
    if i < initial_training_size:
        # Skip the first few steps until we have enough data to
           estimate parameters
        predicted_prices_window.append(prices[i])
        predicted_prices_full.append(prices[i])
        continue

    # Window method
    current_prices_window = prices[max(0, i-window_size):i]
    log_prices_window = np.log(current_prices_window)

    # Full history method
    current_prices_full = prices[:i]
    log_prices_full = np.log(current_prices_full)

    # Estimate parameters
    dt = 1 / TRADING_DAYS_PER_YEAR  # Daily time step

    # Window method
    theta_window = np.mean(log_prices_window)
    sigma_window = np.std(np.diff(log_prices_window)) * np.sqrt(
       TRADING_DAYS_PER_YEAR)

    # Full history method
    theta_full = np.mean(log_prices_full)
    sigma_full = np.std(np.diff(log_prices_full)) * np.sqrt(
       TRADING_DAYS_PER_YEAR)

    # Store the estimated parameters for analysis
    theta_values_window.append(theta_window)
    sigma_values_window.append(sigma_window)
    theta_values_full.append(theta_full)
    sigma_values_full.append(sigma_full)

    # Predict next day's price using mean-reverting model
    epsilon = np.random.normal(0, 1)

    # Window method
    dp_window = (- kappa * (np.log(predicted_prices_window[-1]) -
       theta_window)) * dt + sigma_window * np.sqrt(dt) * epsilon
    next_price_window = predicted_prices_window[-1] * np.exp(
       dp_window)
    predicted_prices_window.append(next_price_window)

    # Full history method
    dp_full = (- kappa * (np.log(predicted_prices_full[-1]) -
       theta_full)) * dt + sigma_full * np.sqrt(dt) * epsilon
    next_price_full = predicted_prices_full[-1] * np.exp(dp_full)
    predicted_prices_full.append(next_price_full)

# Convert to numpy arrays for comparison
predicted_prices_window = np.array(predicted_prices_window)
predicted_prices_full = np.array(predicted_prices_full)

# Plot the actual vs predicted prices
plt.figure(figsize=(14, 7))
plt.plot(prices, label='Actual␣Prices')
```

```python
plt.plot(predicted_prices_window, label='Predicted Prices (Sliding Window [60 days])', linestyle='--')
plt.plot(predicted_prices_full, label='Predicted Prices (Full History)', linestyle=':')
plt.axvline(x=initial_training_size+1, color='red', linestyle='--', label='Prediction Start [20 days]')
plt.title('Actual vs Predicted Stock Prices', fontsize=16)
plt.xlabel('Time (Days)', fontsize=14)
plt.ylabel('Stock Price (USD)', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(True)
plt.show()

# Calculate and print error metrics
mse_window = mean_squared_error(prices, predicted_prices_window)
mae_window = mean_absolute_error(prices, predicted_prices_window)
mse_full = mean_squared_error(prices, predicted_prices_full)
mae_full = mean_absolute_error(prices, predicted_prices_full)
print(f"Window method - MSE: {mse_window:.2f}, MAE: {mae_window:.2f}")
print(f"Full history method - MSE: {mse_full:.2f}, MAE: {mae_full:.2f}")

# Plot the estimated parameters
plt.figure(figsize=(14, 10))
plt.subplot(2, 1, 1)
plt.plot(theta_values_window, color='orange', label='Sliding Window')
plt.plot(theta_values_full, color='green', label='Full History')
plt.title('Estimated Long-term Mean \theta(t)', fontsize=16)
plt.xlabel('Days', fontsize=14)
plt.ylabel('\theta(t)', fontsize=14)
plt.legend(fontsize=14)
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(sigma_values_window, color='orange', label='Sliding Window')
plt.plot(sigma_values_full, color='green', label='Full History')
plt.title('Estimated Volatility \sigma (t)', fontsize=16)
plt.xlabel('Days', fontsize=14)
plt.ylabel('\sigma (t)', fontsize=14)
plt.legend(fontsize=14)
plt.grid(True)
```

References

Glasserman, P. (2013). *Monte carlo methods in financial engineering* (Vol. 53). Springer Science & Business Media.

Musiela, M., & Rutkowski, M. (2006). *Martingale methods in financial modelling.* Springer Berlin Heidelberg.

Oksendal, B. (2013). *Stochastic differential equations: an introduction with applications.* Springer Science & Business Media.

Predictable, W. A. R. A. (1995). Implementing option pricing models. *The Journal of Finance*, *50*(1).

Shumway, R. H., Stoffer, D. S., & Stoffer, D. S. (2000). *Time series analysis and its applications* (Vol. 3). Springer.