OPTION PRICING MODELS AND THEIR ACCURACY

Team Code - E39

FINSEARCH ENDTERM REPORT

Amol Pagare 22B3971
Angad Singh 22B1211
Krish Bansal 23B0661

Mentor: Srishti Makkar
August 15, 2024

Contents

WHAT ARE OPTIONS?

1.1 Basics Of Options

It is a non-contingent claim where the party initiating the contract is known as the writer or seller of the option contract and the other party is called the holder or the buyer.

The buyer holds the long position in the contract and the seller holds the short position in the contract.

1.2 Types of option contracts –

Call Option

❖ Here the writer is the seller and the holder is the buyer and since the holder does not have the **obligation** to buy the contract he has to pay a premium to the writer to buy the contract and that premium is called the price of the option.

Put Option

❖ Here the writer is the buyer and the holder is the seller and since the holder does not have the **obligation** to sell the contract he has to pay a premium to the writer to buy the contract and that premium is also called the price of the option.

Note: **Strike price(K)**: At which the contract is made on.

1.3 Categories of option contracts –

1) **American option**

The contract can be exercised if required at any time till the contract matures.

2) **European option**

The contract can only be exercised if required only at the time of maturity.

1.4 Payoff and Gain :

For a call option if the spot price of the underlying asset is more than the strike price the holder would love to exercise it.

Similarly For a put option if the spot price is less than the strike price the holder would love to exercise it.

❖ A CALL option is said to be
1. in-the-money if $K < S_t$;
2. at-the-money if $K = S_t$;
3. out-of-the-money if $K > S_t$;

❖ A PUT option is said to be
1. in-the-money if $K > S_t$;
2. at-the-money if $K = S_t$;
3. out-of-the-money if $K < S_t$;

The payoff is expressed as:

❖ Payoff for buyer (long position):

Call Option: $\max(0, S_T - K)$
Put Option: $\max(0, K - S_T)$

❖ Payoff for seller (short position):

Call Option: $\max(0, S_T - K)$
Put Option: $\max(0, K - S_T)$

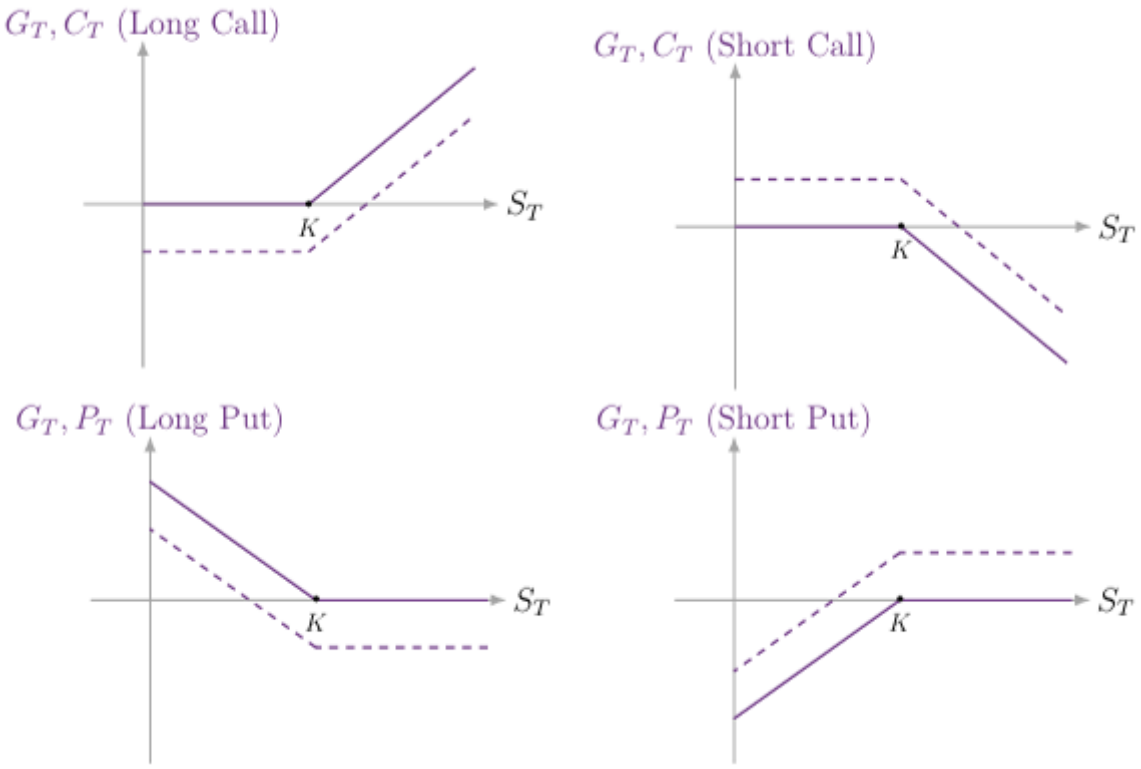
And now if we take the premium into account the gain is given by –

❖ For holder:
$G_{C_t} = C_T - \text{premium} \cdot e^{rT}$

❖ For writer:
$G_{C_{t'}} = -G_{C_t}$

1.4 PAYOFF AND GAIN :

Graphs illustrating the gains for both holders and writers, taking the premium into account



1.5 Mathematical Modeling of Options Parameters

Bounds for premium :
I am writing out the direct results for it. Although there other proper derivations for it but most it can be directly observed and reasoned out why the result is so!

Case 1: **European Call on Non dividend Paying Stock**

$$\max(0, S_0, K_0 - Ke^{-rT}) \leq C_0^{\epsilon} \leq S_0$$

Case 2: **European Put on Non dividend paying stock**

$$\max(0, Ke^{-rT} - S_0) \leq P_0^{\epsilon} \leq Ke^{-rT}$$

Case 3: **European Option on dividend paying stock**

$$\text{For Call Option: } \max(0, S_0 - D_0 - Ke^{-rT}) \leq C_0^{\epsilon} \leq S_0 - D_0$$
$$\text{For Put Option : } \max(0, Ke^{-rT} + D_0 - S_0) \leq P_0^{\epsilon} \leq Ke^{-rT}$$

Case 4: **Non dividend paying American option**

$$C^{\epsilon} = C^{a}$$

Case 5: **American put on non dividend paying stock**

$$\max(0, K - S_0) \leq P_0^{a} \leq K$$

Case 6: **American option on dividend paying stock**

$$\text{For Call Option: } \max(0, S_0 - D_0 - Ke^{-rT}, S_0 - K) \leq C_0^{a} \leq S_0$$
$$\text{For Put Option : } \max(0, S_0 - D_0 - Ke^{-rT}, S_0 - K) \leq C_0^{a} \leq S_0$$

Relation between call and put option premium for both American and European options :
For non dividend paying stock European

$$C_0^{\epsilon} - P_0^{\epsilon} = S_0 - Ke^{-rT}$$

For non dividend paying stock American

$$S_0 - K \leq C_0^{a} - P_0^{a} \leq S_0 - Ke^{-rT}$$

OPTION PRICING MODELS

Option Pricing Models are mathematical tools that utilize specific variables to compute the theoretical value of an option. This theoretical value represents an estimation of an option's worth based on known inputs. Essentially, option pricing models help us determine the fair value of an option. Armed with this estimate, finance professionals can fine-tune their trading strategies and portfolios. Consequently, option pricing models serve as potent instruments for finance professionals engaged in options trading.

2.1 Binomial Option Pricing Model

The binomial options pricing model (BOPM) is a versatile numerical approach in finance that is utilized for valuing options. This method employs a "discrete-time" model based on a lattice to track the changing price of the underlying financial instrument over time.

The model offers a clear and comprehensible technique for determining the value of options. It achieves this by simulating potential price movements of the underlying asset. This approach allows for the consideration of factors such as volatility, strike price, and time remaining until the option expires.

True to its name, this method assumes that the underlying price is confined to only two states in any given interval of time. The binomial option pricing model uses an iterative procedure, enabling the specification of nodes, or points in time, during the time span between the valuation date and the option's expiration date.

Assumptions

The binomial model is based on the following assumptions:

1. The stock price can move to one of two possible prices in each time step (up or down).
2. The probabilities of up and down movements are constant in each time step.
3. The option can be exercised at any point before expiration (American-style options can be priced with modifications).
4. There are no transaction costs or taxes.
5. The risk-free interest rate is constant and known.

Steps to Use the Binomial Model to Price Options

1. **Create the Binomial Tree**:
   - Divide the time to expiration T into n discrete intervals (time steps) of equal length $\Delta t = \frac{T}{n}$.
   - At each time step, the stock price can move up by a factor u or down by a factor d.
2. **Calculate Up and Down Factors**:
   - Calculate the up factor u and down factor d using the stock's volatility σ. Typically, $u = e^{\sigma\sqrt{\Delta t}}$ and $d = e^{-\sigma\sqrt{\Delta t}}$.
3. **Determine Risk-Neutral Probabilities**:
   - Calculate the risk-neutral probability p of an upward move using the risk-free rate r.
   - The probability of a downward move is $1 - p$.
4. **Initialize the Tree**:
   - Construct the binomial tree by calculating the stock prices at each node using the up and down factors.
5. **Calculate Option Value at Expiration**:
   - At the final nodes (at expiration), calculate the option's payoff. For a call option:
     $$C_{n,j} = \max(S_{n,j} - X, 0)$$
     and for a put option:
     $$P_{n,j} = \max(X - S_{n,j}, 0)$$
6. **Perform Backward Induction**:
   - Move backward through the tree to calculate the option value at each node by discounting the expected value of the option prices at the next time step.

For American options, compare the value of immediate exercise to the value of holding the option.

2.2 The Black-Scholes Model

The Black–Scholes model is a widely used mathematical model for pricing options in finance, it provides a theoretical estimate of the price of European-style options based on risk-neutral pricing. The model relies on continuously revising the portfolio of the underlying asset to eliminate risk, known as "continuously revised delta hedging." While widely adopted, the model often requires adjustments to account for real-world market conditions. One key feature is its reliance on the parameter of future volatility, which guides the calibration of other derivative pricing models based on the "volatility surface." The model's insights include no-arbitrage bounds and risk-neutral pricing, with the Black–Scholes equation allowing for derivative pricing using numerical methods when an explicit formula is not feasible.

Assumptions

The Black-Scholes model is based on several key assumptions:

1. The stock price S_t follows a geometric Brownian motion with constant drift μ and volatility σ:
$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$$
where W_t is a standard Wiener process (Brownian motion).
2. The risk-free interest rate r is constant and known.
3. There are no transaction costs or taxes, and securities are perfectly divisible.
4. The option can only be exercised at expiration (European-style option).
5. The stock pays no dividends during the life of the option.
6. There are no arbitrage opportunities.

❖ Itô's Lemma

Stochastic Processes

Let X_t denote a stochastic process, typically modeled as:

$$dX_t = \mu(X_t, t)\, dt + \sigma(X_t, t)\, dW_t$$

where:

- $\mu(X_t, t)$ is the drift term, representing the expected instantaneous rate of change of X_t.
- $\sigma(X_t, t)$ is the volatility term, representing the standard deviation of the random fluctuations around X_t.
- dW_t is the differential of a standard Wiener process (Brownian motion), which represents the random noise or uncertainty.

Itô's Lemma Statement

Itô's Lemma provides a formula for finding the differential dY of a function $Y(X_t, t)$ of a stochastic process X_t. The lemma states:

$$dY = \frac{\partial Y}{\partial t}\, dt + \frac{\partial Y}{\partial X}\, dX + \frac{1}{2}\frac{\partial^2 Y}{\partial X^2}\, (dX)^2$$

Explanation

- **Drift Term ($\frac{\partial Y}{\partial t}\, dt$)**: This term accounts for the change in Y due to changes in time t.
- **Deterministic Term ($\frac{\partial Y}{\partial X}\, dX$)**: This term reflects how Y changes with respect to changes in X_t.
- **Stochastic Term ($\frac{1}{2}\frac{\partial^2 Y}{\partial X^2}\, (dX)^2$)**: This term captures the additional variability introduced by the random component dX squared, which is $(dW_t)^2 = dt$.

Derivation of the Black-Scholes PDE

The price of a derivative $V(S,t)$ depends on the underlying asset price S and time t. By applying Itô's lemma to $V(S,t)$, we obtain:

$$dV = \frac{\partial V}{\partial t}\,dt + \frac{\partial V}{\partial S}\,dS + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\,dS^2$$

Substituting dS_t and dS_t^2 from the stock price dynamics:

$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$$

$$dS_t^2 = \sigma^2 S_t^2\,dt$$

we get:

$$dV = \frac{\partial V}{\partial t}\,dt + \frac{\partial V}{\partial S}(\mu S_t\,dt + \sigma S_t\,dW_t) + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S_t^2\,dt$$

Simplifying, we have:

$$dV = \left(\frac{\partial V}{\partial t} + \mu S_t\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 V}{\partial S^2}\right)dt + \sigma S_t\frac{\partial V}{\partial S}\,dW_t$$

To eliminate the stochastic term, construct a portfolio consisting of one option and $-\Delta$ shares of the underlying stock, where $\Delta = \frac{\partial V}{\partial S}$:

$$\Pi = V - \Delta S$$

The dynamics of Π are given by:

$$d\Pi = dV - \Delta\,dS$$

Substituting the expressions for dV and dS, and noting that the dW_t terms cancel out, we obtain:

$$d\Pi = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 V}{\partial S^2}\right)dt$$

Since the portfolio Π is risk-free, it must earn the risk-free rate:

$$d\Pi = r\Pi\,dt = r(V - \Delta S)\,dt$$

Equating the two expressions for $d\Pi$ and simplifying, we get the Black-Scholes partial differential equation (PDE):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 V}{\partial S^2} + rS_t\frac{\partial V}{\partial S} - rV = 0$$

Solution to the Black-Scholes PDE

The solution to the Black-Scholes PDE for a European call option is given by the Black-Scholes formula:

$$C = S_0 N(d_1) - X e^{-rt} N(d_2)$$

For a European put option, the formula is:

$$P = X e^{-rt} N(-d_2) - S_0 N(-d_1)$$

Where:

$$d_1 = \frac{\ln(S_0/X) + (r + \sigma^2/2)t}{\sigma\sqrt{t}}$$
$$d_2 = d_1 - \sigma\sqrt{t}$$

Here:

- C is the call option price.
- P is the put option price.
- S_0 is the current stock price.
- X is the strike price of the option.
- r is the risk-free interest rate.
- t is the time to expiration.
- σ is the volatility of the stock.
- $N(\cdot)$ is the cumulative distribution function of the standard normal distribution.

CASE STUDY: IMPLEMENTATION OF BOTH MODELS

Option Valuation Parameters

- Current stock price (S_0): Rs.100
- Strike price (K): Rs.95
- Time to maturity (T): 3 months
- Risk-free interest rate (r): 8% per annum
- Volatility (σ): 20% per annum

3.1 Using Black-Scholes Model

Calculate d_1 and d_2:

$$d_1 = \frac{\ln\left(\frac{100}{95}\right) + (0.08 + 0.5 \cdot 0.2^2) \cdot \frac{3}{12}}{0.2 \cdot \sqrt{\frac{3}{12}}} = 0.76$$

$$d_2 = d_1 - 0.2 \cdot \sqrt{\frac{3}{12}} = 0.66$$

Find $N(d_1)$ and $N(d_2)$:

$$N(d_1) = 0.7764, \quad N(d_2) = 0.7454$$

Value of the call option (C):

$$C = 100 \cdot N(d_1) - \frac{95}{e^{0.08 \cdot \frac{3}{12}}} \cdot N(d_2) = 8.23$$

3.2 Using Binomial Method

Calculate U and D:

$$U = e^{0.2 \cdot \sqrt{\frac{3}{12}}} = 1.1, \quad D = \frac{1}{U} = 0.9$$

Calculate risk-neutral probabilities:

$$p_u = \frac{e^{0.08 \cdot \frac{3}{12}} - D}{U - D} = 0.6, \quad p_d = 1 - p_u = 0.4$$

Compute the expected value of the option 3 months from now:

$$\text{Expected value} = Cu \cdot p_u + Cd \cdot p_d = 15 \cdot 0.6 + 0 \cdot 0.4 = 9$$

Expected value of the option at present:

$$\text{Present value} = 9 \cdot e^{-0.08 \cdot \frac{3}{12}} = 8.82$$

3.3 Differences between Both Models

Model	**Binomial Model**	**Black-Scholes Model**
Type of Model	Discrete-time	Continuous-time
Option Types	European, American	European
Pricing Method	Recursive backward induction	Closed-form analytical formula
Assumptions	Flexible, can handle varying conditions	Constant volatility, no dividends, log-normal asset price distribution
Complexity	More complex for large number of steps	Simpler to use once formula is known
Computational Efficiency	Less efficient for very fine time intervals	Highly efficient

MONTE CARLO SIMULATIONS

Monte Carlo simulations for option pricing are a powerful tool in financial mathematics, leveraging computational algorithms to estimate the value of options by simulating the random paths of the underlying asset's price. This method, renowned for its flexibility and applicability to complex derivatives, provides a robust framework for incorporating various factors like volatility, strike price, and time to expiration into the pricing process.

Monte Carlo simulation is one of the most important algorithms in quantitative finance. If we look at the Black-Scholes Merton formula, it is given by:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = rV$$

To discretize this equation for simulating price paths, we consider:

$$S_{t+\Delta t} = S_t \exp\left[\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} Z_t\right]$$

where Z_t is a random draw from a standard normal distribution, $\mathcal{N}(0,1)$.

For the specific case used in this report for performing the Monte Carlo simulations, the asset price can be simulated as:

$$S = 100 \times \exp\left[\left(0.08 - 0.5 \times 0.2^2\right)\Delta t + 0.2 \times \sqrt{\Delta t} \times Z_t\right]$$

where Z_t can be obtained using the function:

$$\text{random.standard\_normal}((M+1, I), \text{axis} = 0)$$

in Python, where M is the number of time steps, and I is the number of paths. Thus, this is referred to as simulating I paths with M time steps, where $\Delta t = \frac{\text{time}}{M}$.

Consider $M = 100$ and $I = 100000$. We initialize a matrix S with $S[0] = S_0$. Then, the value of the option is given by:

$$\text{Option Value} = \exp(-rT) \times \frac{\sum \text{maximum}(S[-1] - K, 0)}{I}$$

Here is the Python code for the process:

```
import math
import numpy as np

S0 = 100.0
K = 95.0
T = 3/12
r = 0.08
sigma = 0.2
M = 100
dt = T / M
I = 100000

S = S0 * np.exp((r - 0.5 * sigma**2) * dt + sigma * math.sqrt
    (dt) * np.random.standard_normal((M+1, I)), axis=0)

S[0] = S0
Final_value = math.exp(-r * T) * np.sum(np.maximum(S[-1] - K,
    0)) / I
```

The process of Monte Carlo simulations involves:

- **Defining the problem:** Identify and define the uncertain variables and the form of their probability distributions.
- **Random Sampling:** Generate values for these variables using random sampling.
- **Simulation:** Use these random inputs to run the model multiple times, simulating different scenarios.
- **Analysis of Results:** Aggregate the results to explain the probability distribution of possible outcomes.

IMPLEMENTATION OF MODELS ON REAL STOCK

https://github.com/Amolpagare10/FinSearch-24
Apple Stock(APPL) from JAN'23 to JAN'24 stock was used.

5.1 Monte Carlo Simulation

```
import yfinance as yf
import numpy as np
import pandas as pd

def monte_carlo_simulation(S0, K, T, r, sigma, M, I):
    dt = T / M
    Z = np.random.standard_normal((M, I))
    S = np.zeros((M + 1, I))
    S[0] = S0
    for t in range(1, M + 1):
        S[t] = S[t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt + sigma * np.sqrt(dt) *

    payoff = np.maximum(S[-1] - K, 0)
    option_price = np.exp(-r * T) * np.mean(payoff)
    return option_price

data = yf.download('AAPL', start='2023-01-01', end='2024-01-01')

S0 = data['Close'][-1]  # Last closing price
K = 180.0               # Strike price
T = 30 / 365            # Time to maturity (30 days)
r = 0.05                # Risk-free rate (5%)
sigma = 0.25            # Volatility (25%)
M = 100                 # Number of time steps
I = 100000              # Number of simulations


entry_dates = []
exit_dates = []
profits = []
```

```
for i in range(len(data) - 1):
    S0 = data['Close'][i]
    entry_date = data.index[i]

    monte_carlo_price = monte_carlo_simulation(S0, K, T, r, sigma, M, I)

    if monte_carlo_price < 2.0:
        entry_dates.append(entry_date)

        exit_date = data.index[min(i + 10, len(data) - 1)]
        S1 = data['Close'][min(i + 10, len(data) - 1)]

        exit_price = monte_carlo_simulation(S1, K, T, r, sigma, M, I)
        profits.append(exit_price - monte_carlo_price)
        exit_dates.append(exit_date)

trade_details_monte_carlo = pd.DataFrame({
    'Entry Date': entry_dates,
    'Exit Date': exit_dates,
    'Profit': profits
})

trade_details_monte_carlo.to_csv('monte_carlo_trade_details.csv', index=False)

print("Monte Carlo trade details saved to monte_carlo_trade_details.csv")
```

5.2 Binomial Model

```
import numpy as np
import pandas as pd
import yfinance as yf

def binomial_model(S0, K, T, r, sigma, N, option_type='call'):
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))  # Up factor
    d = 1 / u                        # Down factor
    p = (np.exp(r * dt) - d) / (u - d)  # Risk-neutral probability

    ST = np.zeros(N + 1)
    for i in range(N + 1):
        ST[i] = S0 * (u ** (N - i)) * (d ** i)

    if option_type == 'call':
        option = np.maximum(ST - K, 0)
    elif option_type == 'put':
        option = np.maximum(K - ST, 0)

    for j in range(N - 1, -1, -1):
        for i in range(j + 1):
            option[i] = np.exp(-r * dt) * (p * option[i] + (1 - p) * option[i + 1])

    return option[0]


data = yf.download('AAPL', start='2023-01-01', end='2024-01-01')


K = 180.0               # Strike price
T = 30 / 365            # Time to maturity (30 days)
r = 0.05                # Risk-free rate (5%)
sigma = 0.25            # Volatility (25%)
N = 100                 # Number of time steps
```

```
entry_dates = []
exit_dates = []
profits = []

for i in range(len(data) - 1):
    S0 = data['Close'][i]
    entry_date = data.index[i]

    binomial_price = binomial_model(S0, K, T, r, sigma, N, option_type='call')

    if binomial_price < 2.0:
        entry_dates.append(entry_date)

        exit_date = data.index[min(i + 10, len(data) - 1)]
        S1 = data['Close'][min(i + 10, len(data) - 1)]

        exit_price = binomial_model(S1, K, T, r, sigma, N, option_type='call')
        profits.append(exit_price - binomial_price)
        exit_dates.append(exit_date)


trade_details_binomial = pd.DataFrame({
    'Entry Date': entry_dates,
    'Exit Date': exit_dates,
    'Profit': profits
})

trade_details_binomial.to_csv('binomial_trade_details.csv', index=False)

print("Binomial trade details saved to binomial_trade_details.csv")
```

5.3 Black Scholes Model

```
import numpy as np
import pandas as pd
import yfinance as yf
from scipy.stats import norm

def black_scholes(S, K, T, r, sigma, option_type='call'):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    if option_type == 'call':
        price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    elif option_type == 'put':
        price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)

    return price

data = yf.download('AAPL', start='2023-01-01', end='2024-01-01')

K = 180.0               # Strike price
T = 30 / 365            # Time to maturity (30 days)
r = 0.05                # Risk-free rate (5%)
sigma = 0.25            # Volatility (25%)
```

```
entry_dates = []
exit_dates = []
profits = []

for i in range(len(data) - 1):
    S0 = data['Close'][i]
    entry_date = data.index[i]

    bs_price = black_scholes(S0, K, T, r, sigma, option_type='call')

    if bs_price < 2.0:
        entry_dates.append(entry_date)

        exit_date = data.index[min(i + 10, len(data) - 1)]
        S1 = data['Close'][min(i + 10, len(data) - 1)]

        exit_price = black_scholes(S1, K, T, r, sigma, option_type='call')
        profits.append(exit_price - bs_price)
        exit_dates.append(exit_date)

trade_details_black_scholes = pd.DataFrame({
    'Entry Date': entry_dates,
    'Exit Date': exit_dates,
    'Profit': profits
})

trade_details_black_scholes.to_csv('black_scholes_trade_details.csv', index=False)

print("Black-Scholes trade details saved to black_scholes_trade_details.csv")
```