monte-carlo-black-scholes-merton

September 1, 2023

[15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from scipy.stats import norm
```

[2]:
```
datasets = ['BAC']

for dataset in datasets:
    Ticker = yf.Ticker(dataset)
    data = Ticker.history(start="2023-08-01", end="2023-09-01")
    filename = f"{dataset}_data.csv"
    data.to_csv(filename)
    print(f"Download data for {dataset} and saved as {filename}")
```

```
Download data for BAC and saved as BAC_data.csv
```

[3]:
```
Ticker = 'BAC'
start_date = '2023-08-01'
end_date = '2023-09-01'
data = yf.download(Ticker, start=start_date, end=end_date)
```

```
[**********100%%**********]  1 of 1 completed
```

[4]:
```
stock_date = yf.download(Ticker, start=start_date, end=end_date)
```

```
[**********100%%**********]  1 of 1 completed
```

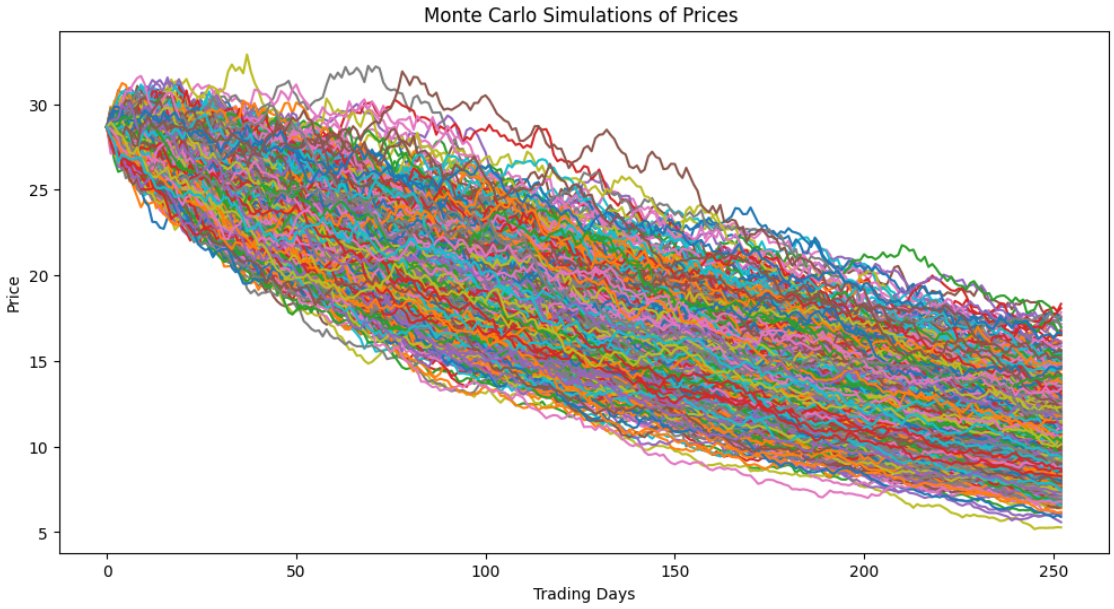
[5]:
```
stock_date['Daily_Return'] = stock_date['Adj Close'].pct_change().dropna()
```

[6]:
```
initial_price = stock_date['Adj Close'][-1]
num_simulations = 1000
num_days = 252
```

[7]:
```
np.random.seed(0)
Daily_Return_mean = stock_date['Daily_Return'].mean()
Daily_return_std = stock_date['Daily_Return'].std()
```

```
simulations = []
for _ in range(num_simulations):
    daily_return_simulated = np.random.normal(Daily_Return_mean,␣
 ↪Daily_return_std, num_days)
    price_path = [initial_price]
    for day_return in daily_return_simulated:
        price_path.append(price_path[-1] * (1 + day_return))
    simulations.append(price_path)
```

[8]:
```
plt.figure(figsize=(12, 6))
for simulation in simulations:
    plt.plot(simulation)
plt.title('Monte Carlo Simulations of Prices ')
plt.xlabel('Trading Days')
plt.ylabel('Price')
plt.show()
```



1 Black-Scholes-Merton

[14]:
```
stock_date['Daily_Returns'] = stock_date['Adj Close'].pct_change()
stock_date['Log Returns'] = np.log( 1 + stock_date['Daily_Returns'])
```

[32]:
```
def black_scholes_merton(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + 0.5*sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
```

```
    return call_price
```

[33]:
```
stock_date.tail()
```

[33]:
```
                 Open       High        Low      Close  Adj Close    Volume  \
Date
2023-08-25  28.639999  28.790001  28.299999  28.500000  28.264462  34228600
2023-08-28  28.690001  29.000000  28.570000  28.760000  28.522314  33075200
2023-08-29  28.889999  29.260000  28.719999  29.170000  28.928925  30428200
2023-08-30  29.219999  29.270000  28.930000  29.040001  28.800001  33366400
2023-08-31  28.930000  28.969999  28.530001  28.670000  28.670000  37203500

            Daily_Return  Daily_Returns  Log Returns  Volatility
Date
2023-08-25     -0.004193      -0.004193    -0.004202    0.013004
2023-08-28      0.009123       0.009123     0.009081    0.013004
2023-08-29      0.014256       0.014256     0.014155    0.013004
2023-08-30     -0.004457      -0.004457    -0.004467    0.013004
2023-08-31     -0.004514      -0.004514    -0.004524    0.013004
```

[34]:
```
stock_date['Volatility'] = stock_date['Daily_Return'].std()
```

[35]:
```
stock_date.tail()
```

[35]:
```
                 Open       High        Low      Close  Adj Close    Volume  \
Date
2023-08-25  28.639999  28.790001  28.299999  28.500000  28.264462  34228600
2023-08-28  28.690001  29.000000  28.570000  28.760000  28.522314  33075200
2023-08-29  28.889999  29.260000  28.719999  29.170000  28.928925  30428200
2023-08-30  29.219999  29.270000  28.930000  29.040001  28.800001  33366400
2023-08-31  28.930000  28.969999  28.530001  28.670000  28.670000  37203500

            Daily_Return  Daily_Returns  Log Returns  Volatility
Date
2023-08-25     -0.004193      -0.004193    -0.004202    0.013004
2023-08-28      0.009123       0.009123     0.009081    0.013004
2023-08-29      0.014256       0.014256     0.014155    0.013004
2023-08-30     -0.004457      -0.004457    -0.004467    0.013004
2023-08-31     -0.004514      -0.004514    -0.004524    0.013004
```

[36]:
```
volatility = 0.01
interests_rate = 0.05
strike_price = 28.67
expiry = '2023-08-31'
```

[37]:
```
stock_date['Option Price'] = black_scholes_merton(
```

```
    stock_date['Adj Close'], strike_price, (pd.to_datetime(expiry) - stock_date.
 ↪index).days / 360,
    interests_rate, volatility)
```

[38]:
```
plt.figure(figsize=(12, 6))
plt.plot(stock_date.index, stock_date['Adj Close'], label='Stock Price',␣
 ↪linewidth=2)
plt.plot(stock_date.index, stock_date['Option Price'], label='Option Prices',␣
 ↪linestyle='--', linewidth=2)
plt.title('Stock Prices vs Option Stock')
plt.xlabel('Date')
plt.ylabel('Prices')
plt.legend()
plt.grid(True)
plt.show()
```