Option Pricing Project Report

Raunak Kumar
Mathematics Department
IIT Bombay
Mumbai, India
kraunak1402@gmail.com

Project Overview

The **Option Pricing Project** is designed to implement and test multiple models used for option pricing in financial markets. The project demonstrates the calculation of option prices using three distinct methods:

- **Black-Scholes Model**
- **Binomial Tree Model**
- **Monte Carlo Simulation**

Additionally, the project fetches stock data from Yahoo Finance using the `pandas-datareader` library and allows the visualization of stock prices over time.

Objectives

The primary objectives of the project are:

- Implement three different option pricing models:
  - **Black-Scholes Model**: A closed-form solution for pricing European call and put options.
  - **Binomial Tree Model**: A numerical method to price options by discretizing time and price movements.
  - **Monte Carlo Simulation**: A stochastic approach to estimating option prices through random sampling.
- Fetch historical stock price data from Yahoo Finance for use in pricing options.
- Visualize stock price movements using matplotlib.
- Provide a framework to test and validate the option pricing models.

Project Structure

The project is organized as follows:

```
option_pricing_project/
 main.py
 option_pricing/
    __init__.py
    black_scholes.py
    binomial_tree.py
    monte_carlo.py
    option_pricing_model.py
    ticker.py
 test_script.py
 README.md
```

- `main.py`: This script acts as the entry point for demonstrating the functionality of the option pricing models and stock data fetching. It includes examples of how to fetch stock data, apply pricing models, and visualize results.
- $\texttt{option}_p ricing/ : Thisisthemainpackagecontainingthecorefun$
  - `black_scholes.py`: Implements the Black-Scholes option pricing formula.
  - `binomial_tree.py`: Implements the Binomial Tree method for option pricing.
  - `monte_carlo.py`: Implements the Monte Carlo simulation for pricing options.
  - `option_pricing_model.py`: This is the base class for option pricing models, containing common properties and methods shared by the other models.
  - `ticker.py`: Contains methods for fetching stock data from Yahoo Finance and plotting it.

`test_script.py`: A testing script that demonstrates the full functionality of the `option_pricing` package. It includes examples of:

- Fetching stock data from Yahoo Finance
- Using the Black-Scholes model to price options
- Using the Binomial Tree model to price options
- Using Monte Carlo simulations to estimate option prices

`README.md`: The readme file contains instructions for setting up and running the project.

Functionality

Stock Data Fetching

The project fetches stock data using the `pandas-datareader` library, specifically from Yahoo Finance. The `Ticker` class provides methods to:

- Fetch historical stock data (`get_historical_data`)
- Retrieve specific columns such as adjusted closing prices (`get_columns`, `get_last_price`)
- Plot stock price data (`plot_data`)

For example, to fetch the stock data for Tesla (TSLA):

```
data = Ticker.get_historical_data('TSLA')
```

Option Pricing Models

1. Black-Scholes Model: The **Black-Scholes Model** is implemented in the `black_scholes.py` module. It uses the famous Black-Scholes formula to calculate the price of European call and put options based on the following parameters:

- Spot Price (S)
- Strike Price (K)
- Time to Expiration (T)
- Risk-Free Rate (r)
- Volatility (σ)

Example:

```
BSM = BlackScholesModel(100, 100, 365, 0.1, 0.2)
BSM.calculate_option_price('Call Option')
BSM.calculate_option_price('Put Option')
```

2. Binomial Tree Model: The **Binomial Tree Model** is implemented in `binomial_tree.py`. This model approximates the price of an option by modeling price movements in discrete time intervals. It uses a binomial tree to represent possible outcomes over the life of the option.

Example:

```
BOPM = BinomialTreeModel(100, 100, 365, 0.1, 0.2, 15000)
BOPM.calculate_option_price('Call Option')
BOPM.calculate_option_price('Put Option')
```

3. Monte Carlo Simulation: The **Monte Carlo Simulation** is implemented in `monte_carlo.py`. This model uses random sampling to simulate a wide range of possible future stock price movements, and then computes the average payoff of the option under these simulated scenarios.

Example:

```
MC = MonteCarloPricing(100, 100, 365, 0.1, 0.2, 10000)
MC.simulate_prices()
MC.calculate_option_price('Call Option')
MC.calculate_option_price('Put Option')
```

Visualization

The project also allows you to visualize the historical stock data. This functionality is provided by the `plot_data` method in the `Ticker` class, which generates plots of the stock's adjusted closing prices over time.

TESTING

The `test_script.py` provides comprehensive tests for the models implemented. It:

- Fetches stock data for a specific ticker (e.g., TSLA)
- Tests option pricing using the Black-Scholes model, Binomial Tree model, and Monte Carlo simulation
- Outputs the calculated option prices and visualizes the stock data

RESULTS

The results of running `test_script.py` would output the following:

- Stock data for the selected ticker (e.g., TSLA) will be fetched from Yahoo Finance and printed.
- Option prices will be calculated for both call and put options using the Black-Scholes, Binomial Tree, and Monte Carlo models.
- The Monte Carlo simulation results will be plotted to visualize the range of possible outcomes.

CONCLUSION

This project demonstrates how multiple models can be used to estimate option prices, each with its own strengths and use cases. The integration of stock data fetching and visualization provides a comprehensive framework for understanding and applying option pricing in real-world scenarios.