CN4238R Final Project - Black Scholes Model for Options Pricing

Name: Mario Lorenzo

Matric No: A0193781U

Description

This is the file containing descriptions and instructions regarding the code developed for CN4238R final project.

File Descriptions

`BlackScholesModel.py`

This is the file containing the Black-Scholes model, implemented as a Python class. The rationale of creating the Black-Scholes model as a Python class is to make the code neater by applying the Object-Oriented Programming concept.

`black-scholes_xxxx_plot.py`

This is the Python scripts that are used to generate plots on input parameters change versus the options price calculated by the Black-Scholes model. There are five files named in this format. The `xxxx` can be changed by `duration`, `interestrate`, `spotprice`, `strikeprice`, and `volatility`, which represents what parameter is adjusted to observe the change in the options price.

`options_price_retriever.py` and `options_price_retriever.ipynb`

This is the Python script that are used to scrape the options price from Yahoo! Finance website. This file is created because Yahoo! Finance does not provide a way to download the file for the options price in its website. Therefore, I created this script that uses HTTP GET request to the Yahoo! Finance API server to retrive the options price data in form of JSON string. This is then formatted and converted to two `.csv` files. One file is the call options data, and the other is the put options data. The instruction on using this Python script is available in the later section. The `.ipynb` file is the Jupyter Notebook version of the script. It is strongly encouraged to use the `.ipynb` version instead since it provides a step-by-step process of the script, if you are interested to study the step-by-step processing.

`black-scholes_demo.py` and `black-scholes_demo.ipynb`

This is the Python script that are used that simulates the options pricing using the Black-Scholes model, then compare it with the actual options price, retrieved from `options_price_retriever.py`. It utilizes `pandas_datareader` library to scrape the 1-year stocks price from Yahoo! Finance (Fortunately, there is a way to scrape the stocks price from Yahoo! Finance directly). This stock price is used for computing the volatility factor. More instructions is provided in the later section to use the demo code. The `.ipynb` file is the Jupyter Notebook version of the script. It is strongly encouraged to use the `.ipynb` version instead since it provides a step-by-step process of the script, if you are interested to study the step-by-step processing. Also, using the `.ipynb` version allows us to see the plot and dataframe change step by step, which is nicer for illustration purposes.

Instructions

Setting up

Before running the scripts, it is best to ensure that you have installed all the libraries used. The libraries used for all scripts are listed as such: 1. `datetime` 2. `matplotlib` 3. `numpy` 4. `pandas` 5. `pandas_datareader` 6. `requests` 7. `seaborn`

Please make sure that prior to running the scripts, you have the libraries listed above installed.

Alternatively, `requirements.txt` file is provided. You can automatically install the libraries above by running `pip install -r requirements.txt` in your terminal.

Retrieving Options Price

As mentioned in the previous section, the options price can be retrieved using `options_price_retriever.py`. In order to use this script, you need to register for an API key at here. Register for the free plan, and the API key can be obtained from the Dashboards menu. Note that the free plan allows for only 100 daily API requests limit, thus this script can only be run 100 times per day for one API key, but it should be more than enough. If you have the API key with you, copy and paste the key to `api_key.txt` provided. Initially, it is left blank to be filled by your API key. If you do not want to register for the key, but just want to try the script, you can request my own API key to be tried by emailing me :).

Inside the `options_price_retriever.py`, there are several parameters to be tuned as needed. The first one is the `tickerCode` variable. Change this variable to the stock's ticker code that you want to retrieve the data. The default ticker code set is `TSLA`. Another variables to be changed is the `day`, `month`, and `year` which represent the expiration date. The default date set is 5th of November 2021.

Using Demo Code

Using the demo code is relatively straightforward. The tunable parameters is placed in one section, where the description is available there and where can the information needed be found. Basically, the parameters that need to be tuned are the date range for the stocks price information to calculate the volatility, the ticker code, the Black-Scholes model-related parameter (spot price, maturity date, and risk-free interest rate), and the options dataset `.csv` file.