Interview Questions

June 2018

Implied Volatility Calculator

This problem will require you to write an application that will take an input file ”input.csv” and write out a new file calculated from the inputs.

- we are looking for a solution in python,
- a procedural solution is not acceptable,
- the solution should include appropriate unit testing,
- you will be evaluated on your coding style, and on the correctness of the results.

Input

The input file is a sequence of rows that represents a simplified version of market data. Each row represents a trade. The columns are

- *ID*: unique identifier of the trade,
- *Underlying Type*: represents the option underlying type. It can be either `Stock` of `Future`,
- *Underlying*: represents the option underlying mid,
- *Risk-Free Rate*: represents the prevailing risk-free rate for the option maturity,
- *Days To Expiry*: represents the number of calendar days to option expiry (you can assume the day counting convention is ACT/365),
- *Strike*: represents the option strike,
- *Option Type*: represents whether the option is a `Call` or `Put`,
- *Model Type*: denotes what pricing model the market convetion assumed. It can be either `BlackScholes` or `Bachelier`,

- *Market Price*: represents the last traded price of the option

You should handle cases where the input is not valid, in which case, you should skip the offending row.
Sample input:

```
82,Stock,0.0575,-0.0014,344.5459,1.4992,Put,BlackScholes,1.4439
```

The first row always consists of the header.

Output

Calculate the market implied volatility for each trade. If you can't find a solution for a given entry the output vol must be `nan`[1]. The solution should be accurate up to $10^{-}8$ absolute tolerance in price space.
The output is a CSV that contains the following columns:

- *ID*,
- *Spot*,
- *Strike*,
- *Risk-Free Rate*,
- *Years To Expiry*,
- *Option Type*,
- *Model Type*,
- *Implied Volatility*,
- *Market Price*

Be aware that spot in case of a future represents the value of the future underlying.
Sample output (you should report as many digits as you can for computed quantities):

```
40,0.3445,0.1284,-0.0045,0.6504,Call,BlackScholes,0.5732,0.2163
```

Requirements

- You should create a private git repository that shows your progress over time,
- Using third party library is fine, but it should be limited as much as possible
- Any computational/numerical part should be written without any external library

[1] Use `float(’nan’)` or `numpy.nan`