Project 9: Options Pricing

Instructions

Project 9 is due on November 22th by 10:00 pm. This is a hard deadline, so no exceptions. You must push your local repository back to GitHub before the deadline. Your repository must contain:

- The Python code you used to complete the project;
- A notebook file named `main.ipynb` that generates all required plots;
- A `report.pdf` file with your answers to the project questions. The report must also contain an Appendix with the code used to solve the project;
- All plots in the report must be self-contained. Self-contained means that a reader who only sees your figure (image and caption, but not the surrounding text) can understand what you are plotting. This translates to all plots having axis titles, correct units on the axis, and a caption that summarizes what is plotted.

This project makes use of stock and **options** data. Refer to the Data page and the questions below for instructions on how to download the data and which files to download (requires Duke login). You must complete all exercises for both of your stocks using the data at the 5-minutes sampling frequency, unless stated otherwise.

You can obtain the repository for this project by clicking **on this link**.

Questions

The purpose of this project is to evaluate the BLS option pricing model by first computing the predicted option price using the realized variance for volatility, and then comparing the BLS model value for the option to the market price. Since the realized variance is the best estimate of volatility over the trading day, perhaps RV will do well in the options setting. Let's see what the data say.

In the course's data page, you will find several files containing options data. Each data file contains about 30–60 observations on the closing prices of **put** options with various strikes and expirations, along with some other data needed to implement the Black-Scholes formula. Each row of the file corresponds to a particular put option on the ETF SPY, for the S&P 500. The table below summarizes the contents of each column:

In class, we discussed the Black-Scholes formula for the call option. It turns out there is generally more trading activity and economic interest in puts than calls. The likely reason is that the put option is like an insurance, in the sense that it guarantees the buyer of the put the minimum price of K, should the buyer happen to hold the stock and

Column	Description
1	YYYYMMDD date of the trading day
2	HHMM always 1600 (4 PM) since these are closing prices
3	S_0 closing price of SPY
4	yearly interest rate in percent (0.02 means 0.02%)
5	yearly dividend yield on SPY in percent
6	closing price of the Put option in dollars (1.14 means \$1.14)
7	strike price K
8	days until expiration (36 days)
9	YYYYMMDD of expiration date (unused here)

the stock price collapse at maturity, $K/S_T > 1$. Of course the trader on the other side of the put option, i.e., the trader who writes the put option (the insurer), loses $K - S_T$ per contract in this case. The writer of the put is very much like an insurance company.

A.

Select a single file containing options data. Specify which file you chose below. Write a function to import the data in the appropriate format. To know what columns of the data will be needed you should read what the exercises below are asking. Remember to change the units to the appropriate format to be used to calculate option prices from the Black-Scholes model. You should use Numpy for storing the data in arrays.

Hint: When importing with `np.loadtxt`, you can set the input `usecols` to import only the requested columns.

B.

Each line of the options data contains the data at the market close time. Compute the realized variance for each of those days. Convert the values to the appropriate units.

Hint: The method `np.unique` returns the unique values in an array. You can use it to find the dates you care about. Also, consider using `np.loadtxt` again for loading the stock data. This function also takes an input named `unpack`, which can be used to directly unpack the data into several variables. Consider reshaping your array of prices to facilitate the rest of the computations, you can do so with `np.reshape` (remember to specify the correct value of `order`). Finally, the function `np.diff` can be used to take differences across lines or columns of an array.

C.

The data file has the closing market price for each put, along with the information needed to compute the Black-Scholes model-implied value. Plot the market prices for these options against each option's strike-to-underlying ration. Interpret the results.

Hint: Remember to use `matplotlib` and, in particular, you should use `plt.scatter` to plot the values. Add a vertical line when the ratio is 1, you can use `plt.axvline` to do so.

D.

The Black-Scholes model has a closed form solution:

$$\begin{aligned}\text{Put}(S, K, T, r, q, \sigma) &= K\mathrm{e}^{-rT}\Phi(-d_2) - S\mathrm{e}^{-qT}\Phi(-d_1)\\ d_1 &= \frac{\log\frac{S}{K} + (r - q + \sigma^2/2)T}{\sigma\sqrt{T}}\\ d_2 &= d_1 - \sigma\sqrt{T}\end{aligned}$$

Where Φ is the cdf of a standard normal distribution. This solution takes into account the dividend yield, but assumes it is constant and known. The idea is that the price of an option where the underlying pays a continuous dividend yield of q is equal to the price of an option where the underlying pays no dividend but its price is $S\mathrm{e}^{-qT}$. In practice, the dividend yield of the market index is stable and around 2.1% per year.

Write a function that computes the price of an option given the inputs. Your function should work with single or multiple inputs (in the form of a numpy array).

The normal cdf is available in the `scipy` package, inside the subpackage `stats`. You can import it directly by running `from scipy.stats import norm`. The cdf function can then be used by calling `norm.cdf`. Read the function's documentation.

Hints: Assume each input to your function is a numpy array. For example S is a numpy array containing several prices, and so on for K, T, r, q and σ. What is the return type of `norm.cdf`?

E.

Use your function from the previous exercise to compute the model-implied prices using the realized variance as the measure of variance (in the appropriate units). Plot these prices against the strike-to-underlying ratio. Also plot the actual market prices on the same figure. Make sure to use a different color for the plots and add a legend. Interpret the results.

Hint: Use `plt.scatter` but specify a different color for the prices computed using the realized variance.

F.

As a general rule, analysts and financial economists have found that BLS tends to underprice (under value) far out-of-the-money put options, and usually, but by no means always, correctly prices put options close to the money: 0.98 < strike-to-underlying ratio < 1.02. Is your plot consistent with this finding?

G.

Choose 9 other options data files and repeat the exercise E for each of them. Create a single figure (with many plots) summarizing your findings.

Hints: To create a single figure with multiple plots, you will need to use the function `plt.subplots`. Consider the code below:

```
# Fake data for example:
x = np.arange(5)
y = np.arange(5)
# Create a figure that can hold many plots in a 2x2 grid
```

```
fig, axes = plt.subplots(nrows=2, ncols=2)
# fig: the window that will hold the plots
# axes: array that will hold each plot in the 2x2 grid
print(axes.shape)                    # (2, 2)
# Plot on position 0, 0
axes[0, 0].scatter(x, y)
# Plot on position 1, 0
axes[1, 0].scatter(x, y)
# And so on ...
```

The code will create a figure window that holds a 2 by 2 grid. Each position of this grid has an empty plot that we can be filled. Elements in a Numpy array can be iterated over in a for-loop:

```
fig, axes = plt.subplots(nrows=2, ncols=2)
for line in axes:
    for ax in line:
        ax.plot(x, y)
```

H.

Suppose you work for a firm in the financial services industry, and your job is "to make money". In the previous items you have collected BLS model prices along with the associated market prices. If you take the model seriously, then whenever $put_{BLS,i} < put_{mkt,i}$ the market is overvaluing the option relative to the model. The correct trading strategy would be to sell (write) the put options and thereby collect the option premium $put_{mkt,i}$, which could be held as cash while you wait for the option to expire worthlessly according to the BLS model. Would you feel comfortable going to your boss and indicating that the firm could make a lot of money by selling (writing) a $10 million of out-of-the-money put options on SPY?