**Investment Portfolio Optimization using Stock Options**

***Jonathan McWilliams***

***February 2020***

***Analysis Introduction and Approach***

Building an optimum portfolio within constraints is a common investment problem. Given a finite amount of imaginary investment capital, this model attempts to demonstrate that exact concept; to find the optimal selection of investments to maximize profit via stock options.

The analysis follows a basic four step process:

1. *Market movements are predicted for a user-defined timeframe*
2. *Future prices of three preselected stocks are forecasted (using step 1 as input)*
3. *Profitability of each option type (buy or sell, call or put) is assessed at each strike price*
4. *Funds are allocated, via linear programming, to maximize investment profit*

***Model Execution***

While the underpinning of this analysis is investment related, the emphasis was put on creating a step-by-step *methodology* for the user to visually explore and learn from – not creating a state of the art investment algorithm. The use of built-in functions is limited and leans more on rudimentary Python code that offers a *viewable* progression through solutions. The user will be able to see examples of how data is scraped, manipulated, visualized, and finally – acted upon. The few pre-made functions that are used in the model are cited at the end of this document and also in the code block itself. Please note, there are a few assumptions that the user must supply to the model for an accurate output:

1. *Current month and option expiration month [code block 2]*
2. *Date to start the market simulation [code block 10]*

***Scope of Investment Opportunity***

This analysis assumes all options are held until, and exercised on, their expiration date – the European model. Consequently, the options’ price fluctuations throughout the investment timeframe is not important. The core of this project is assigning an *intrinsic* value to each option and acting accordingly on the stock at time of expiration; short-term trading of only option positions themselves – without the buying and selling of the underlying asset – is not addressed in this model. Furthermore, the model evaluates both the buying and *selling (aka “writing”)* strategy for both *call* and *put* positions, resulting in 12 possible investments to be made:

[3] stocks (Google, Apple, or GE)

x [2] option types (call or put)

x [2] transaction types (buy or sell)

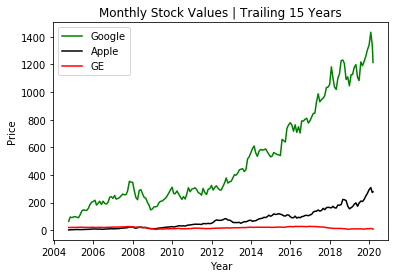
**= 12 possible investments**

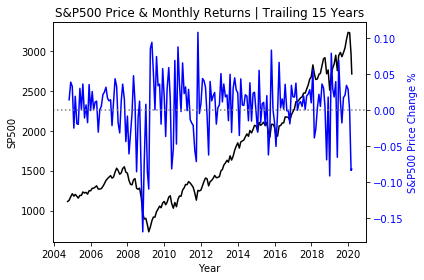
***Data Sources***

Historical market prices and stock prices were downloaded from *Alpha Vantage* and aggregated at a *monthly* level *[code block 3]*. All stock option data was scraped and downloaded from *Yahoo Finance* using the *Beautifulsoup1* package *[code blocks 12a-12c]*. Monthly data slices were decided upon since the extra nuance that daily data would provide would be outweighed by its cumbersomeness. The aim was to keep the analysis as simple as possible and monthly data allows that. An *Alpha Vantage* login is required to acquire the market and stock data and is provided along with the python code in an *‘ENV’* file2.

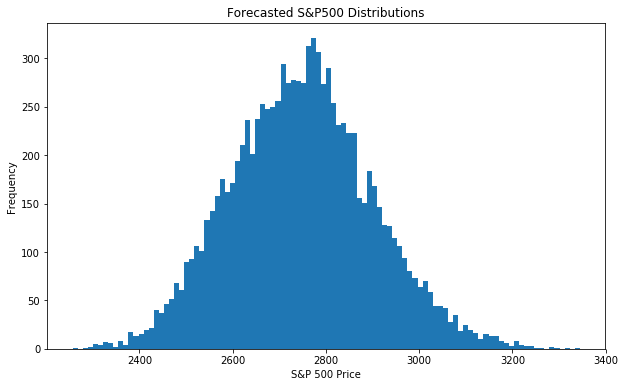
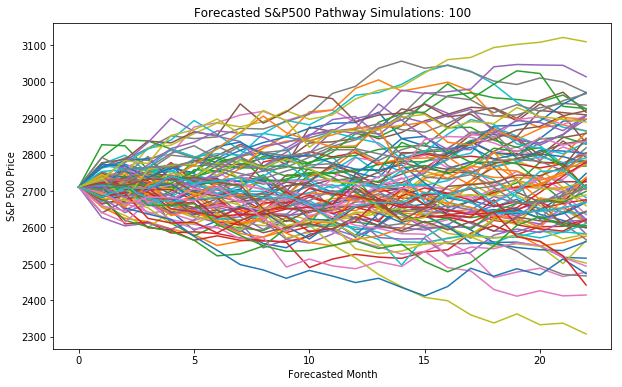
***Market and Stocks in Scope***

The *market* referred to in this analysis is the S&P 500 as it is a common benchmark for the overall health of the domestic equity market. In the past two decades, there have been four pronounced movements: the dotcom crash in the early 2000s, the recovery from 2002 to 2007, the 2008 financial collapse, and the strong and steady recovery from 2009 until early 2020. At the time of this analysis (February 2020), the S&P 500 has experienced fairly uninterrupted growth for more than a decade; even advancing past the 3,000 marker – a historically significant marker.

Three stocks are used in this analysis and are preselected for the user: Google, Apple, and General Electric (GE). These stocks are very different in their industry as well as their performance; Google is a high-growth software giant that has seen its price rise in astounding multiples every year; Apple is a technology hardware company that has seen solid growth over the past decade; GE is a mature conglomerate that has seen horrific equity performance ever since the 2000 crash. This diversity in industry and stock performance is not a coincidence; in a real-world scenario we would want to diversify our capital to make sure an unforeseen market movement against one company’s industry would not affect our entire portfolio.



**Forecasting Market Level and Stock Prices**

The model predicts monthly *market* movements until the *option expiration date [code block 8]*, which the user is required to select from three dates: June 2020, January 2021, or January 2022 *[code block 2]*. These particular dates were selected because all three stocks shared these common expiration dates. Visualizations of the market simulation forecast are below.

The chosen forecasting method is *Geometric Brownian Motion3* (GMB); a mathematical simulation that shares the general approach of the Black Scholes model. Before its adoption to finance, Brownian Motion (BM) was first used in the scientific field to explain the random movement of solid matter through fluid pathways; e.g. pollen grains in water or dust particles in air4. This idea of randomness is adopted in our model by applying itself to each monthly return, thereby simulating volatility that an investor would expect to experience. It is important to note that the difference between GMB and basic MB is that the former includes an assumed *growth rate* (also called *drift*). The model assumes that this growth rate is the *risk free rate* in the market – which is assigned to be the month-to-date yield of monthly treasury securities [code block 5]. By including this concept of *drift*, the simulated market climbs steadily upward – but with random variation of returns. The GMB equation is shown below:

Where:

*St = price level at time period t*

*t = time period*

*Δt = the change between current and previous time states*

*r = the risk free rate corresponding to the assumed “drift”*

*σ = standard deviation of returns*

*z = standard normally distributed random variable*

After locking in a final prediction for market level, the model moves on to predict the three individual stock prices at the end of the expiration date *[code block 11]*. The stock predictions track the market’s movement using each stock’s *beta* – a measure of correlation between the market and a given stock *[the betas are scraped from Yahoo finance in code block 6]*.

**Investing Optimization**

The model relies on linear programming5 to select the optimal stock options for the investment portfolio construction *[code block 14]*. The objective of this linear programming model is to maximize profitability, which it does by assigning a *future value* to an option position based on how far it is *in the money* – a term pertaining to *profitability [code block 13].*  However, the model can only maximize profitability after it satisfies specific parameters; one very important parameter being *cash outflow*. Each position’s cashflow profile affects *how much* of that position the investment model is able to buy (because of finite investment capital). The cash outflow profile differs by the option type:

* *Buying a call | cashflow = premium paid out for the option*
* *Buying a put | cashflow = premium paid out for the option*
* *Selling a call | cashflow = premium gained for the option less the current price of the underlying stock33*
* *Selling a put | cashflow = premium gained for the option less the strike price of the underlying stock34*

After assigning a future profit value (based on the prediction of each respective stock price) and cashflow cost to the options, the optimization is executed subject to the following parameters:

* *The max dollar amount that can be invested is set at $1MM (i.e. a cash outflow)*
* *All investment amounts must be more than or equal to zero (i.e. there can be no negative amount invested)*
* *Investment exposure to each of the three respective companies must be at least 15%; this ensures that the model does not overexpose the portfolio to one particular option position*
* *At least 30% of the investment portfolio must be invested in selling options (as opposed to buying); this is to encourage portfolio diversification*

**Model Results and Final Portfolio**

The final block of code *[code block 15]* consolidates, summarizes, and visualizes the final optimized portfolio. The user is able to print the overall portfolio performance as well as the allocation, profitability, and cashflow toward each option type. Please note that these numbers will change continually since the market, stock, and option data is live, and all three will continually fluctuate. As of February 2020, the results are as follows:

*Model performance: a profit of $1.175MM off of a $1MM portfolio* 🡪 *a 117.5% ROI*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Option Allocation** | Buy | Sell | **Profitability** | Buy | Sell | **Cash Out Flow** | Buy | Sell |
| Google Call | - | - | Google Call | - | - | Google Call | - | - |
| Google Put | 475 | - | Google Put | $136,677 | - | Google Put | -$150,000 | - |
| Apple Call | - | 14,632 | Apple Call | - | $955,357 | Apple Call | - | -$700,000 |
| Apple Put | - | - | Apple Put | - | - | Apple Put | - | - |
| GE Call | - | - | GE Call | - | - | GE Call | - | - |
| GE Put | - | 33,333 | GE Put | - | $83,333 | GE Put | - | -$150,000 |

**Model Summary**

As noted in the *Model Execution* section above, the goal of this analysis was to demonstrate basic option valuation. If the readers goal is to fully simulate a real-life investment scenario, a more nuanced approach should be built into this model – particularly viewing all investment decisions through a *risk-adjusted* lens. While the model is removed from a real-life scenario, it successfully shows how to effectively break a multi-step valuation process into respective pieces: data scraping, data manipulation, data aggregation, and portfolio construction.

*Citations & Notes:*

1: Richardson, L. (April 2007). Beautiful soup documentation.

2: The ENV file must be copied and saved to the same folder as the accompanying python code

3: Hilpisch, 2019, *Python for Finance*, Chapter 12: Stochastics

4: Encyclopedia Britannica, <https://www.britannica.com/science/Brownian-motion>

5: Linear programming is executed via the GEKKO package:

Beal, L.D.R., Hill, D., Martin, R.A., and Hedengren, J. D., GEKKO Optimization Suite, Processes, Volume 6, Number 8, 2018, doi: 10.3390/pr6080106