FINANCE

PART A

BITCOIN STOCK TO FLOW MODEL AND WHY IT IS BAD?

The Bitcoin stock to flow model was created and published by the twitter contact PlanB@100trillionUSD on March 2019. He proposed that the stock-to-flow ratio can quantify scarcity, and the stock-to-flow model can be used to determine bitcoin's market value. Stock-to-Flow model is defined as,

Stock-to-Flow, $SF = \frac{Current\ Stock}{Supp \quad Flow}$

Stock refers to the total amount of existing stock of a commodity, while Flow refers to the yearly production of the commodity. The SF serves as an indicator for the number of years it will take for the current production rate to yield the existing stock. As at the time of publication, Gold had 62 SF with a Cap market of $8.4 trillion and value of $1,300 while Silver had 22 SF with a Cap market of $308 billion and value of $16. The Bitcoin had 22 SF with a market value of $70 billion as calculated from the existing stock of 17.5M coin and 0.7M supply flow.

The author, PlanB emphasize on the role of Store of Values (SoV) commodities such as Gold and Silver which have unforgeable costliness and low supply rate that translate to high SF, thus, having monetary relevance. The author further added that Bitcoin is very similar to the SoV commodities (Gold and Silver) as it also has unforgeable costliness because of the high cost of production, the authenticity of transaction, hashing, Proof of concept and the fixed 21 million coins expected to be mined. Bitcoin halving also plays a vital role in the supply growth rate as it occurs every four years thus doubling the flow production. PlanB model estimate that the Bitcoin SF will rise to 50 after the halving on May 2020 which makes it very close to the Gold SF of 62 and will translate to a market value of $1 trillion and a price of $55,000. The model which plot the relationship of Bitcoin SF to its market value also indicates a Power law relationship with R-square of 95% accuracy.

Why is it Bad?

Although, the PlanB model equation which was related to the Gold SF provides a high R-square that signifies that the hypothesis that SF correlates with the market value cannot be rejected, the projection of the model to keep increasing exponentially shouldn't be trusted as the Gold Cap SF to market values held market capitalization from $60 Billion to $9 trillion at the same SF value of 60 over the last 115 years (Cordeiro 2020). Thus argue that the SF doesn't have a relationship with the current or future capitalization as a further analysis done on other cryptocurrencies further solidify the argument.

Further studies on the model indicate that each Bitcoin is projected to value as much as $235 Billion as at 2045 and approach infinity as Bitcoin flow approach 0 in 2140. The model is far from good because SF is the only parameter used in determining the market capitalization and doesn't represent the overall factors attributed to market capitalization for Bitcoin.

In conclusion, while Bitcoin has a growing high SF, the present trend may support the PlanB model but for a very short period as many other factors that determine the market capitalization such as Electricity consumption, miners' perception and activities between the halving play a major role in its evaluation.

Works Cited

Cordeiro, Nico. *Why the Stock-to-Flow Bitcoin Valuation Model Is Wrong*. 30 June 2020, www.coindesk.com/why-the-stock-to-flow-bitcoin-valuation-model-is-wrong.

CryptoWhale. *Here's Why the (S2F) Stock-to-Flow Model Is Completely Wrong*. 2 July 2020, medium.com/in-bitcoin-we-trust/heres-why-the-s2f-stock-to-flow-model-is-completely-wrong-f77d539c147c.

Emblow, Nick. *Falsifying Stock-to-Flow As a Model of Bitcoin Value*. 11 Aug. 2019, medium.com/coinmonks/falsifying-stock-to-flow-as-a-model-of-bitcoin-value-b2d9e61f68af.

PlanB. *Modeling Bitcoin Value with Scarcity*. 22 Mar. 2019, medium.com/@100trillionUSD/modeling-bitcoins-value-with-scarcity-91fa0fc03e25.

Redman, Jamie. *S2F Hopium: Report and Twitter Critics Find Flaws With Bitcoin's Stock-to-Flow Ratio*. 17 May 2020, news.bitcoin.com/s2f-hopium-report-and-twitter-critics-find-flaws-with-bitcoins-stock-to-flow-ratio/.

Part B

Yara Inc is listed on the NYSE with a stock price of $40 - the company is not known to pay dividends. We need to price a call option with a strike of $45 maturing in 4 months. The continuously-compounded risk-free rate is 3%/year, the mean return on the stock is 7%/year, and the standard deviation of the stock return is 40%/year. What is the Black-Scholes call price?

Solution

Stock price, S_t = $40

Strike price, K = $45

Time to maturity, t = 4 months or 4/12 year = 1/3 yr

Risk free rate, r = 3% or 0.03

Mean return on stock, μ = 7% or 0.07

Standard deviation of stock return or annualized volatility, σ = 40% or 0.4

To determine the Black-Scholes Call price, we make use of the formula below,

$$C = S_t N(d_1) - K e^{rt} N(d_2)$$

$$d_1 = \frac{In\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right) t}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

Where $N(d_1)$ and $N(d_2)$ is normal distribution

$$d_1 = \frac{In\left(\frac{40}{45}\right) + \left(0.03 + \frac{0.07^2}{2}\right) * \frac{1}{3}}{0.07\sqrt{\frac{1}{3}}}$$

d_1 =-0.35124

$$d_2 = -0.35124 - 0.4\sqrt{\frac{1}{3}}$$

$$d_2 = -0.58218$$

The $N(d_1)$ and $N(d_2)$ is the cumulative standard normal distribution function with a standardize mean of 0 and standard deviation of 1.

$N(d_1) = 0.36271916$

$N(d_2)$ =0.28021599

$C = 40(0.36271916) - 45e^{-0.03*\frac{1}{3}} *0.28021599$

$$\therefore Call\ price, C = \$2.0245$$

Computer Science

Part A

Why is it a bad idea to use recursion method to find the fibonacci of a number?

Recursion method involves the multi-call function within the function in the evaluation of an algorithm problem. While the recursive method seems as a naïve or brute solution for Fibonacci number, the technique has a time complexity of ≈$O(n^2)$ (exponential time) and space complexity of O(n) (linear space) as this depend on the depth of Fibonacci tree. Thus, the Fibonacci recursive method has a poor computation performance as observed when the n is higher than 100 as this would requires more internal call of the function. A better approach to solve the problem is an iterative method which approach the problem by using two variables with the first storing the previous number and the second to store the current number and iterated for the N time. The Iterative method has the time complexity of O(n) (linear time) and Space complexity of O(1) (constant space) as it uses less memory.

A sample of Recursion method for Fibonacci of a number

```
def calc_fib(n):
    if (n <= 1):
        return n

    return calc_fib(n - 1) + calc_fib(n - 2)
```

A sample of iterative method for Fibonacci of a number

```
def calc_fib_fast(n):
    if n <= 1:
        return n

    previous = 0
    current  = 1

    for _ in range(n - 1):
        previous, current = current, (previous + current)

    return current
```

Part B

Write a function that takes in a Proth Number and uses Proth's theorem to determine if said number is prime? You can write this in any programming language but C/C++/Golang are preferred

Solution

My Preferred programming language is Python, thus, I first solved using python and rewrote the code in C++. The approach used in solving the problem is based on Proth's theorem, that state a number is refer as Proth number based on three conditions;

- The number is equal to $k2^n + 1$
- K and n are positive real number and K is Odd number

- $2^n > K$

The algorithm developed was to firm ascertain if the input number is a Proth number by the following process;

- If the number is even then it is never a Proth number
- Subtract 1 from the number
- Then, find out the occurrence for 2^n
- Get the equivalent value of K
- Compare the result of 2^n and K, if $2^n > K$ then it is a Proth number

Finally, I determine if the Proth number is also a Prime number.

Python Solution

```
def proth_prime(n):
    if n%2 == 0 or n<=2:
        return "Not Proth Number"
    elif isProth(n):
        if isPrime(n):
            return "Proth Prime"
        else:
            return "Proth Number but Not Proth Prime"
    else:
        return "Not Proth Number"

def isProth(n):
        a=n-1
        count=0
        while a%2==0:
            count+=1
            a/=2
        w=2**count
        if w>(n-1)/w:
            return True
        else:
            return False
def isPrime(n):
    i =2
    while i*i<=n:
        if n%i ==0:
            return False
        i+=1
    return True
```

Code rewritten in C++

```
#include <iostream>
#include <cmath>
using namespace std;
bool isProth(int n){
    int a=n-1;
    int count=0;
    int w;
    while (a%2==0){
```

```
        count+=1;
        a/=2;

    }
    w=pow(2,count);
    if (w>(n-1)/w)
        return true;
    else
        return false;

}
bool isPrime(int n){
    int i =2;
    while (i*i<=n){
        if (n%i ==0)
            return false;
        i++;

    }
    return true;
}
void proth_prime(int n){
    if (n%2==0 || n<=2){
        cout<< "Not Proth Number";
    }
    else if (isProth(n)){
        if (isPrime(n)){
            cout<<"Proth Prime";
        }
        else{
            cout<<"Proth Number but not Prime";
        }
    }
    else{
        cout<<"Not Proth Number";
    }

}
int main()
{
    int n;
    cout<<"Enter the number to check for Proth prime >>>";
    cin>>n;
    proth_prime(n);

    return 0;
}
```

MATHS

Over all real numbers, find the minimum value of a positive real number, y such that

$$y = \sqrt{((x+6)^2+25)} + \sqrt{((x-6)^2+121)}$$

Solution

The minimum value of y is determined by the value of y corresponding to the value of x when dy/dx = 0.

Thus,

$$\frac{dy}{dx} = \frac{d(\sqrt{((x+6)^2+25)} + \sqrt{((x-6)^2+121)})}{dx}$$

Let u= $\sqrt{((x+6)^2+25)}$, and $w = \sqrt{((x-6)^2+121)}$

$$\frac{dy}{dx} = \frac{du}{dx} + \frac{dw}{dx}$$

Also, let a = $((x+6)^2+25)$, b = $((x-6)^2+121)$

$$\therefore u = a^{1/2}$$

$$\therefore w = b^{1/2}$$

Thus,

$$\frac{du}{dx} = \frac{du}{da} * \frac{da}{dx}$$

$$\frac{dw}{dx} = \frac{dw}{db} * \frac{db}{dx}$$

$$\frac{da}{dx} = 2(x+6)$$

$$\frac{db}{dx} = 2(x-6)$$

$$\frac{du}{da} = \frac{1}{2}a^{-\frac{1}{2}}$$

$$\frac{dw}{db} = \frac{1}{2}b^{-\frac{1}{2}}$$

$$\therefore \frac{du}{dx} = 2(x+6) * \frac{1}{2}a^{-\frac{1}{2}} = \frac{(x+6)}{\sqrt{((x+6)^2+25)}}$$

Also,

$$\frac{dw}{dx} = 2(x-6) * \frac{1}{2}b^{-\frac{1}{2}} = \frac{(x-6)}{\sqrt{((x-6)^2+121)}}$$

Therefore,

$$\frac{dy}{dx} = \frac{(x+6)}{\sqrt{((x+6)^2+25)}} + \frac{(x-6)}{\sqrt{((x-6)^2+121)}}$$

$$\frac{dy}{dx} = \frac{(x+6)\sqrt{((x-6)^2+121)} + (x-6)\sqrt{((x+6)^2+25)}}{\sqrt{((x+6)^2+25)((x-6)^2+121)}}$$

Equating dy/dx = 0 and solving for x;

$$(x+6)\sqrt{((x-6)^2+121)} + (x-6)\sqrt{((x+6)^2+25)} = 0$$

Or

$$(x+6)\sqrt{((x-6)^2+121)} = -(x-6)\sqrt{((x+6)^2+25)}$$

Squaring both sides;

$$\left((x+6)\sqrt{((x-6)^2+121)}\right)^2 = \left(-(x-6)\sqrt{((x+6)^2+25)}\right)^2$$

$$(x+6)^2 * ((x-6)^2+121) = (x-6)^2 * ((x+6)^2+25)$$

$$(x+6)^2(x-6)^2 + 121(x+6)^2 = (x+6)^2(x-6)^2 + 25(x-6)^2$$

$$\therefore 121(x^2+12x+36) = 25(x^2-12x+36)$$

$$96x^2 + 1752x + 3456 = 0$$

$$x^2 + 18.25x + 36 = 0$$

Solving the quadratic equation using mathematic formula, x = -16 and x= -9/4

Thus, to determine minimum y, we solve for y for each cases of x;

For x=-9/4;

$$y = \sqrt{((x+6)^2+25)} + \sqrt{((x-6)^2+121)}$$

$$y = \sqrt{\left(\left(-\frac{9}{4}+6\right)^2+25\right)} + \sqrt{\left(\left(-\frac{9}{4}-6\right)^2+121\right)}$$

$$y = 20$$

For x = -16;

$$y = \sqrt{((x+6)^2+25)} + \sqrt{((x-6)^2+121)}$$

$$y = \sqrt{((-16+6)^2+25)} + \sqrt{((-16-6)^2+121)}$$

$$y = 35.777$$

Therefore, the minimum real positive number of y is 20 at x=-9/4