YASP: Yet Another Stock Predictor

Umang Bhalla

November 17, 2021

Abstract

In this research paper we dive deep into webscrapping , flask server setup and Optionchain stocks visualisation

Contents

List of Tables

List of Listings

1 Introduction

Optionchain visualiser is a collection of multiple technologies: Python as the programming language, Bash as a scripting language, JSON as a database, Flask as a software-based server, Raspberry Pi as hardware and Arch Linux as operating system.

Optionchain visualiser is a collection of *multiple* technologies, like **python programming language**, **bash scripting**, **json** as database, **flask** as software based server , **Raspberry pi 4b** as hardware server and **arch linux** as operating system [1].

Using web-scraping we extract data from NSE [2] which gets saved in database (as json file) then using matplotlib in first varient and dash(component in plotty [3]) we generate static pngs and interactive graphs.

classoption: - twocolumn

1.1 Mechanism and how it works.

Both **varients**[4] of this project scrape **NSE** [5] to collect relevant data about user provieded stocks or all the public stocks listed on the exchange. After this is done, the collected json data is add to *data* subdirectory and named as *stockname*-fno.json

1.2 Methods of Data Scraping used

[1]arch linux is used as it provides arch user repository which hosts a lot of commandline tools used in this project

[2]National Stock Exchange www.nseindia.com

[3]lllllllllllllll

[4]see sec. 2.2

[5]National Stock Exchange www.nseindia.com

2 Probability by normal distribution

Strike price : current price of stock

Implied volatility is not directly observable, so it needs to be solved using the five other inputs of the Black-Scholes model, which are: - The market price of the option. https://www.investopedia.com/terms/m/market-price.asp - The underlying stock price. - The strike price. https://www.investopedia.com/terms/s/strikeprice.asp - The time to expiration. - The risk-free interest rate.

2.0.1 Market Price

The market price is the current price at which an asset or service can be bought or sold. The market price of an asset or service is determined by the forces of supply and demand. The price at which quantity supplied equals quantity demanded is the market price.

Shocks to either the supply or the demand for a good or service can cause the market price for a good or service to change. A supply shock is an unexpected event that suddenly changes the supply of a good or service. A demand shock is a sudden event that increases or decreases the demand for a good or service. Some examples of supply shock are interest rate cuts, tax cuts, government stimulus, terrorist attacks, natural disasters, and stock market crashes. Some examples of demand shock include a steep rise in oil and gas prices or other commodities, political turmoil, natural disasters, and breakthroughs in production technology.

2.0.2 Strike Price

A strike price is the set price at which a derivative contract can be bought or sold when it is exercised. For call options, the strike price is where the security can be bought by the option holder; for put options, the strike price is the price at which the security can be sold.

Strike prices are used in derivatives (mainly options) trading. Derivatives are financial products whose value is based (derived) on the underlying asset, usually another financial instrument. The strike price is a key variable of call and put options. For example, the buyer of a call option would have the right, but not the obligation, to buy the underlying security in the future at the specified strike price.

2.1 Black-Scholes

Implied volatility is calculated by taking the market price of the option, entering it into the Black-Scholes formula, and back-solving for the value of the volatility. But there are various approaches to calculating implied volatility. One simple approach is to use an iterative search, or trial and error, to find the value of implied volatility. Iv calculation done by nse

Call iv -> call implied volatility (given by nse) put iv -> put option implied volatility (given by nse) Days of expiry -> expiryDate - today

Strike value win probability =

NORMSDIST(LN(STRIKEPRICE/VALUE)/CALLIV*SQRT(DAYSTOEXPIRATION/365))

NORMSDIST = $(0.5*\text{pi})^2 * e^{(-(z^2)/2)}$

```python
def probability(scrip):
  nearest = []
  perf = fetch_data(scrip)
  underlyingValue = perf['underlyingValue']
  strikePrices = perf['strikePrices']
  strikePrices = list(dict.fromkeys(strikePrices))

  label = labels(scrip)
  exp = expiry(scrip)

  l = 0
  for i in strikePrices:
    nearest.append(i - underlyingValue)

  for i in nearest:
    if i < 0:
        nearest[l] = nearest[l] * -1
        l = l + 1

  value = strikePrices[nearest.index(min(nearest))]

  for j in perf['stocks']:
    if j['metadata']['expiryDate'] == exp and j['metadata']['strikePrice'] == value:
        if j['metadata']['optionType'] == 'Call':
            iv_call = j['marketDeptOrderBook']['otherInfo']['impliedVolatility']
        if j['metadata']['optionType'] == 'Put':
            iv_put = j['marketDeptOrderBook']['otherInfo']['impliedVolatility']

  prob = []
  daysexpiry = (datetime.datetime.strptime(exp, '%d-%b-%Y') -
                datetime.datetime.strptime(today, '%Y-%m-%d')).days

  for i in label:
#STRIKE<VALUE WIN PROB => NORMSDIST(LN(STRIKEPRICE/VALUE)/CALLIV*SQRT(DAYSTOEXPIRATION/365))
#normsdist = (1/2pi)^2 * e^(-(z^2)/2)
    if i < value and i > 0:
      try:
          normsdist = math.log(i/underlyingValue) / \
              ((iv_call/100)*math.sqrt(daysexpiry/365))
          result = integrate.quad(lambda q: (
              (1/math.sqrt(2 * math.pi))*((math.e)**(q**2/-2.0))), -np.inf, normsdist)
          prob.append(100 - result[0]*100)
        except:
            prob.append(0)
    elif i > value and i > 0:
      try:
          normsdist = math.log(i/underlyingValue) / \
              ((iv_put/100)*math.sqrt(daysexpiry/365))
          result = integrate.quad(lambda q: (
              (1/math.sqrt(2 * math.pi))*((math.e)**(q**2/-2.0))), -np.inf, normsdist)
          prob.append(result[0]*100)
      except:
          prob.append(0)
    else:
        prob.append(0)

  return prob
```

2.1.1 What nse use for IV calculation and we get by api

Calculation of theoretical base price of contracts as per Black –Scholes formula: The options price for a Call option shall be computed as follows:

C = S * N (d1) – X * e ˆ (- rt)* N (d2)

and the price for a Put option is :

P = X * e ˆ (- rt) * N (-d2) – S * N (-d1)

where :

d1 = ln (S / X) + (r + s ˆ 2 / 2) * t

s * vt

d2 = ln (S / X) + (r – s 2 / 2) * t

s * vt

= d1 - s * vt

and C = price of a Call option

P = price of a Put option

S = price of the underlying asset

X = Strike price of the option

r = rate of interest (Rate of interest shall be the relevant MIBOR rate for the day)

t = time to expiration

s = volatility (Volatility shall be the higher of the underlying volatility or the the near month futures contact volatility on the relevant day.)

N represents a standard normal distribution with mean = 0 and standard deviation = 1, and ln represents the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).

2.2 Different varients of the project

2.2.1 User input based

This varient primarily relies on user to decide stock about which he/she has to scrape information. Using the run [6] script provided in the repository user call the fzf [7] prompt. In the fzf [8] prompt user is supposed to write name of the stock they want to search, the search usses fuzzy algorithm [9] , hence if we write *SBI* in the search prompt, **SBIN** , **SBILIFE** , **SBICARD** gets displayed as results ; then user can press tab to select the symbol [10] on the promt and press it again to select the enxt prompt or search for other symbol [11] to add the array that will be fed to python script to start the scrape.

```
#!/bin/bash
rm -rf data
mkdir -p data
fno_list_Arr=$(cat fno_main.json | jq ".[]" | jq ".[].symbol" | fzf --reverse -m -i --height=80%);
arr=($fno_list_Arr)


final="["
for i in "${arr[@]}"
do
  final=$final"{\"label\": $i, \"value\": $i} ,"
done

final=${final::-1}"]"

```

[6] sssssssssss

[7] oooooooooooo

[8] oooooooooooo

[9] fffffffffff

[10] jjjjjjjjjjjj

[11] jjjjjjjjjjjj

```
echo $final
echo $final > fno.json

python3 optionchain.py
```

2.2.2 Server based (Flask)

2.3 Difference between both varients.

3 Propose work

Make easy visualize most difficult concept inside financial markets i.e. OC OC https://www.investopedia.com/terms/o/optionchain.asp

Use basic data to find probability to take informed decision

3.1 Block diagram

4 Data Set and visualization

Call Put bta denge Maxpain https://www.investopedia.com/ter https://zerodha.com/varsity/chapter/max-pain-pcr-ratio/

5 Result and Implementation

minimum requirement, setup

6 Conclusion and Future

7 References