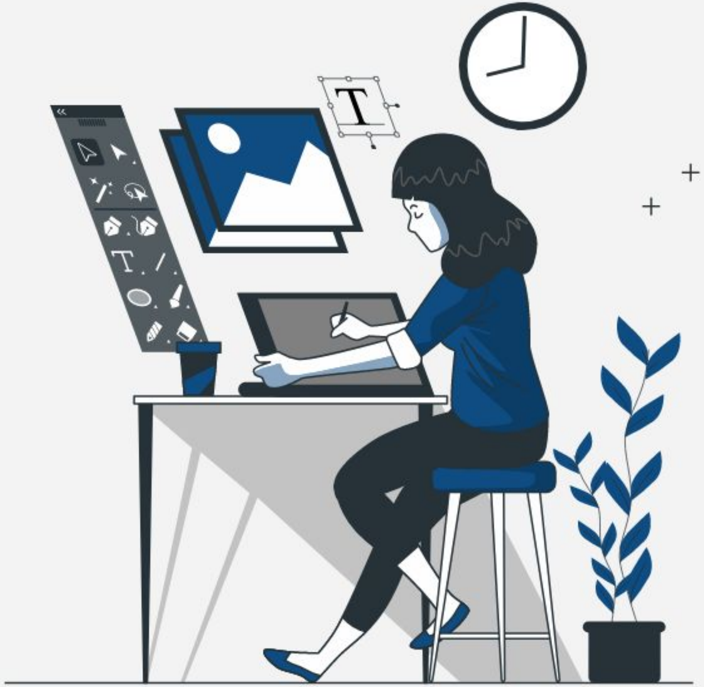
SPX vs SPY

OPTIONS PRICING PROJECT



QF101 G1T3

JUAN, ENJIE, DOMINIQUE, IAN

01	OVERVIEW	“Why is it interesting”
02	METHODOLOGY	Data, Model used
03	BASIC RESULTS	Daily, Total P/L
04	IMPROVEMENTS	Limiting when to buy + 10% Deviation cap
05	IMPROVED RESULTS	Limiting when to buy + 10% Deviation cap
06	CONCLUSION	

ABOUT THE PROJECT

A

Based on: S&P 500
SPX - European
SPY - American

B

SPY is the most traded option

C

Large difference in trade volume
SPX: 1-3 Million daily
SPY: 60-140 Million daily

Overview - Steps

1. Create our own Options Pricing Model
2. Compare the predicted option prices against actual market prices of the options
3. Compare how our model does in predicting SPX options (European) vs SPY options (American)

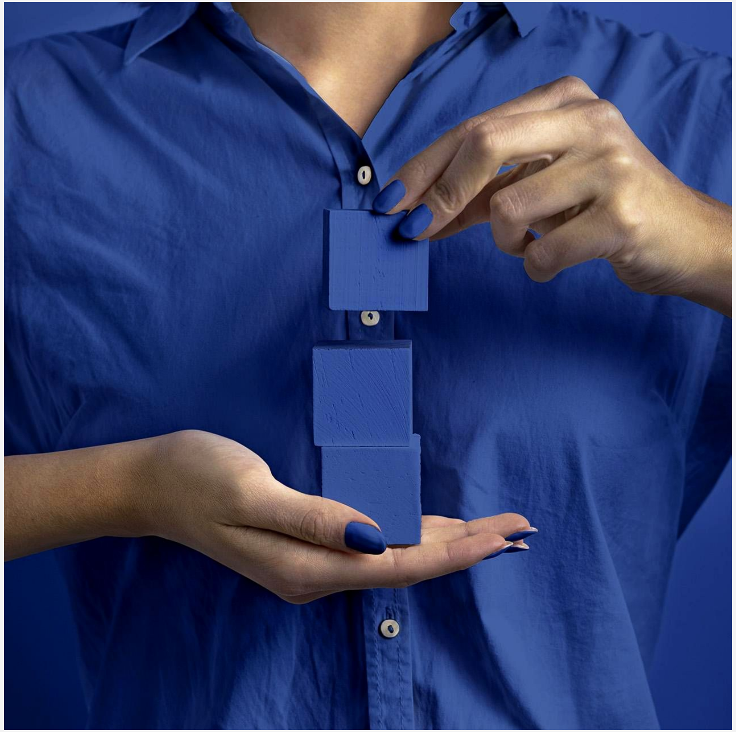
Improvement:

Go long when our predicted value is more than 0% to 10% of the market value, Go short when our predicted value is less than 0% to 10% of the market value

DATA

Data range: 3/1/2020 to 30/12/2022

Options: 30 Days to Expiry



Code

```
# calculate the risk-neutral probability
if method == 'crr':
    u = np.exp(sigma * np.sqrt(dt))
    d = 1 / u
    p = (np.exp((r - div) * dt) - d) / (u - d)
elif method == 'jr':
    u = np.exp((r-div-0.5*sigma**2)*dt+sigma*np.sqrt(dt))
    d = np.exp((r-div-0.5*sigma**2)*dt-sigma*np.sqrt(dt))
    p = 0.5
```

Models

JR

$$p = \frac{1}{2}$$

$$u = e^{(r-\frac{1}{2}\sigma^2)dt+\sigma\sqrt{dt}}$$

$$d = e^{(r-\frac{1}{2}\sigma^2)dt-\sigma\sqrt{dt}}$$

CRR

$$u = e^{\sigma\sqrt{dt}}$$

$$d = e^{-\sigma\sqrt{dt}}$$

$$p = \frac{e^{rdt} - d}{u - d}$$

Buy Signals

Long Call/Put

Enter +1 position when predicted price > market price.

Short Call/Put

Enter -1 position when predicted price < market price.

```
df_input['LONG_CALL'] = np.where(df_input['CALL'] > df_input['C_PRICE'], long_call, short_call)
df_input['LONG_PUT'] = np.where(df_input['PUT'] > df_input['P_PRICE'], long_put, short_put)
dcall = (df_input['CALL'] - df_input['C_PRICE'])/df_input['C_PRICE']
dput = (df_input['PUT'] - df_input['P_PRICE'])/df_input['P_PRICE']
```

Calculating PnL

1. Calculate cost of buying / premium received from entering into a position (PROFIT_LOSS_ON_PURCHASE)
2. Calculate options payoff (PROFIT_LOSS_ON_SALE):
   a. Call: Max(Stock Price - Strike Price, 0)
   b. Put: Max(Strike Price - Stock Price, 0)
3. PnL = PROFIT_LOSS_ON_PURCHASE + PROFIT_LOSS_ON_SALE

```
df_input['PROFIT_LOSS_ON_PURCHASE_CALL'] = -(df_input['LONG_CALL']*df_input['C_PRICE'])
df_input['PROFIT_LOSS_ON_PURCHASE_PUT'] = -(df_input['LONG_PUT']*df_input['P_PRICE'])
df_input['PROFIT_LOSS_ON_PURCHASE'] = df_input['PROFIT_LOSS_ON_PURCHASE_CALL'] + df_input['PROFIT_LOSS_ON_PURCHASE_PUT']
df_input['PROFIT_LOSS_ON_SALE_CALL'] = (df_input['LONG_CALL']*np.maximum(df_input['UNDERLYING_PRICE_EXPIRE'] - df_input['STRIKE'], 0))
df_input['PROFIT_LOSS_ON_SALE_PUT'] = (df_input['LONG_PUT']*np.maximum(df_input['STRIKE'] - df_input['UNDERLYING_PRICE_EXPIRE'], 0))
df_input['PROFIT_LOSS_ON_SALE'] = df_input['PROFIT_LOSS_ON_SALE_CALL'] + df_input['PROFIT_LOSS_ON_SALE_PUT']
df_input['PROFIT_LOSS_CALL'] = df_input['PROFIT_LOSS_ON_SALE_CALL'] + df_input['PROFIT_LOSS_ON_PURCHASE_CALL']
df_input['PROFIT_LOSS_PUT'] = df_input['PROFIT_LOSS_ON_SALE_PUT'] + df_input['PROFIT_LOSS_ON_PURCHASE_PUT']
df_input['PROFIT_LOSS'] = df_input['PROFIT_LOSS_ON_SALE'] + df_input['PROFIT_LOSS_ON_PURCHASE']
```

SPX

Root Mean Squared Error of CRR Call model: 20.894495881960978
Root Mean Squared Error of JR Call model: 20.892574928621944
Root Mean Squared Error of CRR Put model: 21.61877256944796
Root Mean Squared Error of JR Put model: 21.6169893240666
Mean Absolute Error of CRR Call model: 12.179502242149622
Mean Absolute Error of JR Call model: 12.178586841592729
Mean Absolute Error of CRR Put model: 13.381022873010346
Mean Absolute Error of JR Put model: 13.3795543630229
Percentage Mean Absolute Error of CRR Call model: 89.90169561712081 %
Percentage Mean Absolute Error of JR Call model: 89.7479891380906 %
Percentage Mean Absolute Error of CRR Put model: 52.08037360182608 %
Percentage Mean Absolute Error of JR Put model: 52.073426146778615 %

Mean Profit/Loss of CRR model: 8.959887290942877
Mean Profit/Loss of JR model: 8.900853060784788
Volatility of CRR model: 207.2139025439408
Volatility of JR model: 207.2486516839786
Total Profit/Loss of CRR model: 332689.57499999995
Total Profit/Loss of JR model: 330497.57499999995

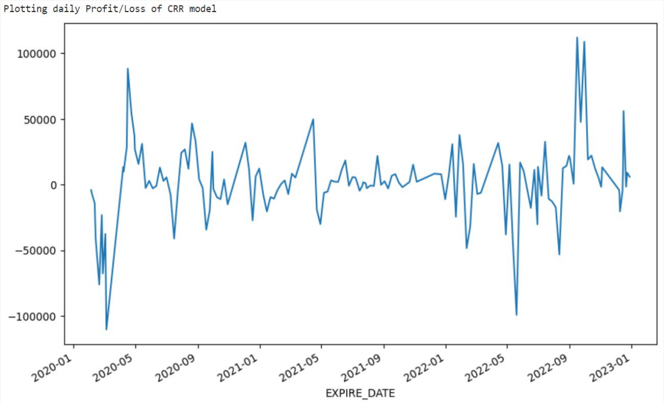
SPY

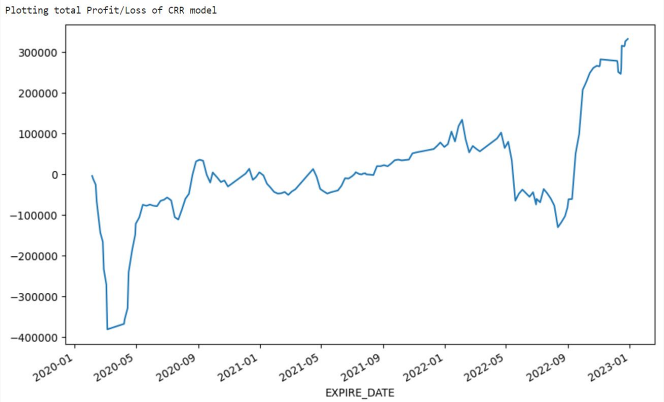
Root Mean Squared Error of CRR Call model: 2.2014083365619896
Root Mean Squared Error of JR Call model: 2.201059135659596
Root Mean Squared Error of CRR Put model: 2.22104470891091
Root Mean Squared Error of JR Put model: 2.2208519746157314
Mean Absolute Error of CRR Call model: 1.33057266941219
Mean Absolute Error of JR Call model: 1.3303482365065098
Mean Absolute Error of CRR Put model: 1.398773103360051
Mean Absolute Error of JR Put model: 1.398604977617586
Percentage Mean Absolute Error of CRR Call model: inf %
Percentage Mean Absolute Error of JR Call model: inf %
Percentage Mean Absolute Error of CRR Put model: 46.934619008242365 %
Percentage Mean Absolute Error of JR Put model: 46.93023731003161 %

Mean Profit/Loss of CRR model: -0.5089830091975012
Mean Profit/Loss of JR model: -0.5032064873973185
Volatility of CRR model: 19.9622973690234
Volatility of JR model: 19.96649186704153
Total Profit/Loss of CRR model: -14498.89000000002
Total Profit/Loss of JR model: -14334.340000000015

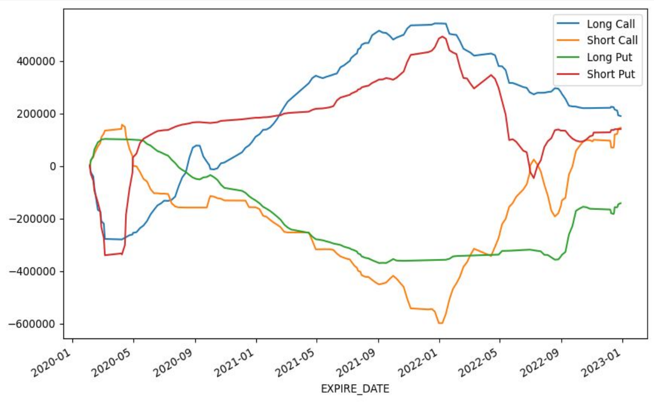
SPX (P/L)

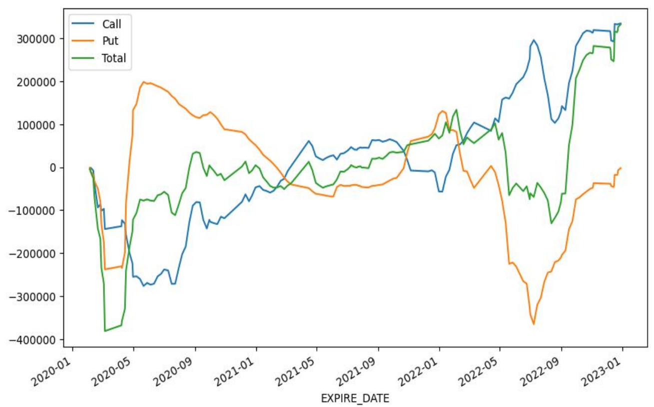
Daily/Total





Call vs Put

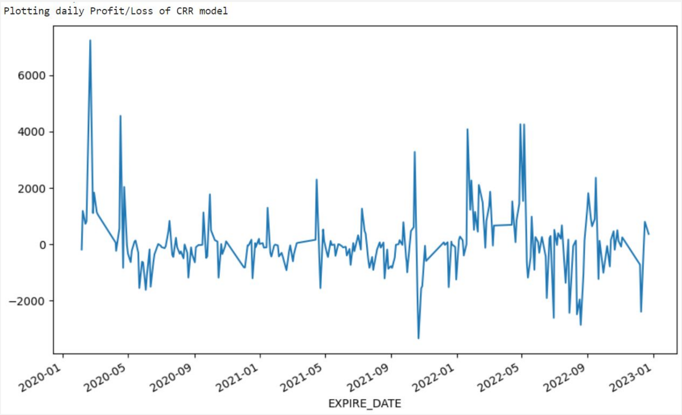


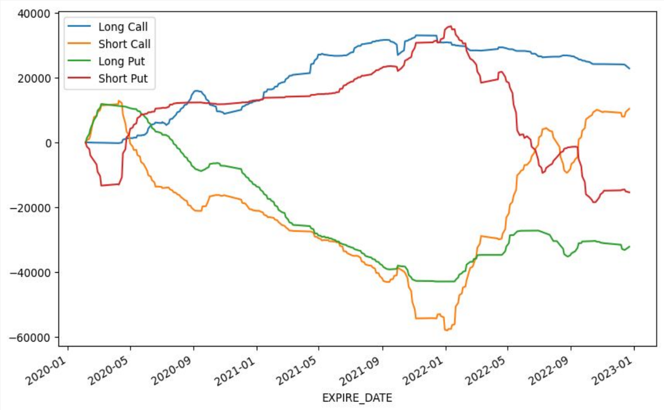


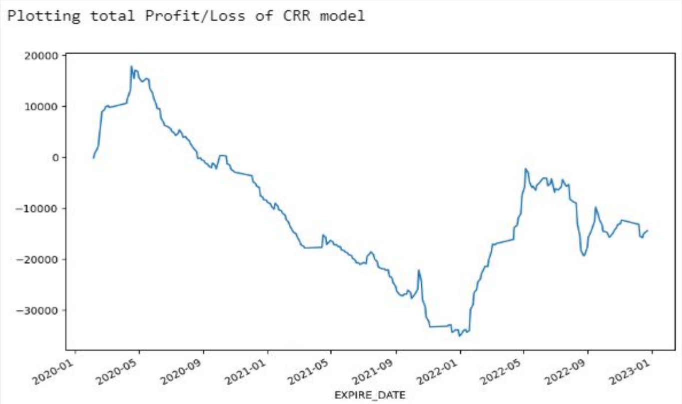
SPY (P/L)

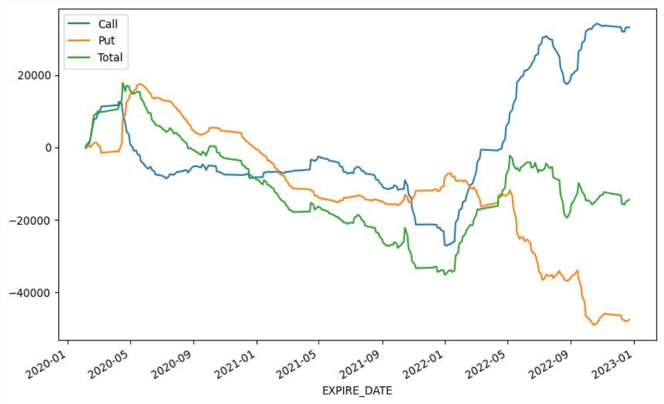
Daily/Total

Call vs Put









LIMITING WHEN TO BUY & 10% PRICE DEVIATION CONSTRAINT

```
# df_input['LONG_CALL'] = np.where(df_input['CALL'] > df_input['C_PRICE'], long_call, short_call)
# df_input['LONG_PUT'] = np.where(df_input['PUT'] > df_input['P_PRICE'], long_put, short_put)
dcall = (df_input['CALL'] - df_input['C_PRICE'])/df_input['C_PRICE']
dput = (df_input['PUT'] - df_input['P_PRICE'])/df_input['P_PRICE']

df_input['LONG_CALL'] = np.where(dcall > 0, np.where(dcall <= 0.1, long_call, 0), np.where(dcall >= -0.1, short_call, 0))
df_input['LONG_PUT'] = np.where(dput > 0, np.where(dput <= 0.1, long_put, 0), np.where(dput >= -0.1, short_put, 0))
```

SPX

```
Root Mean Squared Error of CRR Call model: 20.894495881960978
Root Mean Squared Error of JR Call model: 20.892574928621944
Root Mean Squared Error of CRR Put model: 21.61877256944796
Root Mean Squared Error of JR Put model: 21.6169893240666
Mean Absolute Error of CRR Call model: 12.179502242149622
Mean Absolute Error of JR Call model: 12.178586841592729
Mean Absolute Error of CRR Put model: 13.381022873010346
Mean Absolute Error of JR Put model: 13.3795543630229
Percentage Mean Absolute Error of CRR Call model: 89.90169561712081 %
Percentage Mean Absolute Error of JR Call model: 89.7479891380906 %
Percentage Mean Absolute Error of CRR Put model: 52.08037360182608 %
Percentage Mean Absolute Error of JR Put model: 52.07342614677861 5 %
```

```
Mean Profit/Loss of CRR model: 14.05162047884517
Mean Profit/Loss of JR model: 13.985484500821414
Volatility of CRR model: 198.01085248543916
Volatility of JR model: 198.03539355460376
Total Profit/Loss of CRR model: 521750.72
Total Profit/Loss of JR model: 519295.0249999999
```

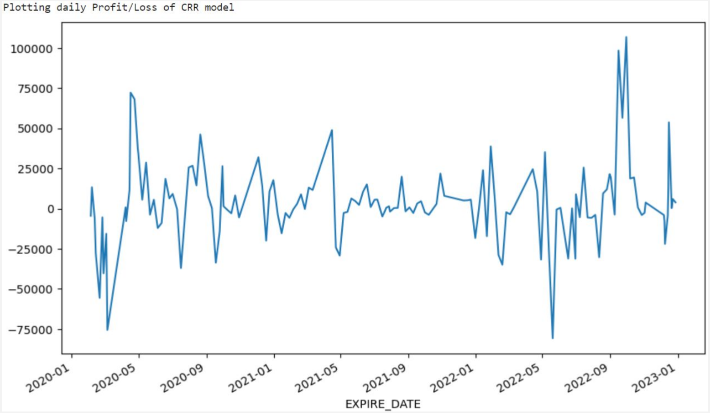
SPY

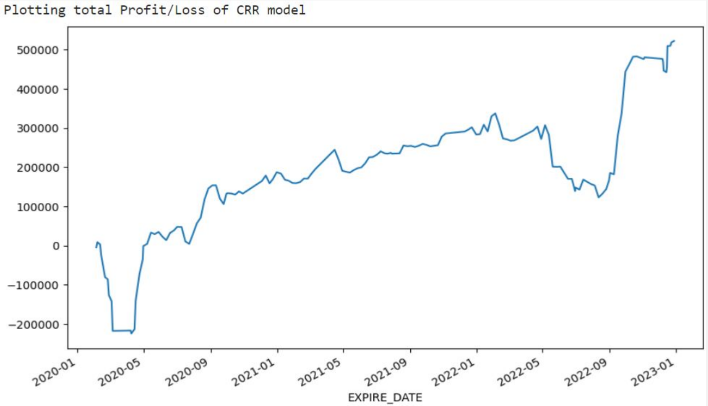
```
Root Mean Squared Error of CRR Call model: 2.2014083365619896
Root Mean Squared Error of JR Call model: 2.201059135659596
Root Mean Squared Error of CRR Put model: 2.221104470891091
Root Mean Squared Error of JR Put model: 2.2208519746157314
Mean Absolute Error of CRR Call model: 1.33057266941219
Mean Absolute Error of JR Call model: 1.3303482365065098
Mean Absolute Error of CRR Put model: 1.398773103360051
Mean Absolute Error of JR Put model: 1.398604977617586
Percentage Mean Absolute Error of CRR Call model: inf %
Percentage Mean Absolute Error of JR Call model: inf %
Percentage Mean Absolute Error of CRR Put model: 46.934619008242365 %
Percentage Mean Absolute Error of JR Put model: 46.9302373100316 1 %
```

```
Mean Profit/Loss of CRR model: 0.2437407849469913
Mean Profit/Loss of JR model: 0.2512162114723018 3
Volatility of CRR model: 19.076043535318924
Volatility of JR model: 19.080055467270356
Total Profit/Loss of CRR model: 6943.199999999994
Total Profit/Loss of JR model: 7156.144999999999
```

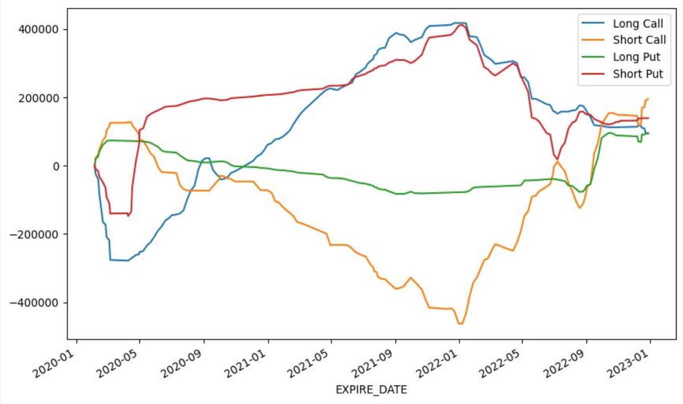
SPX (P/L)

Daily/Total



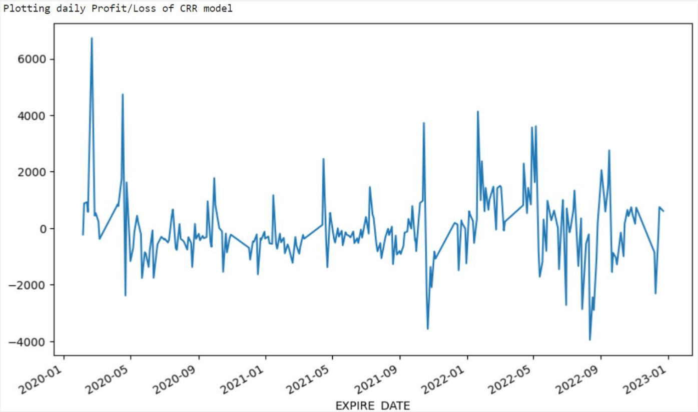


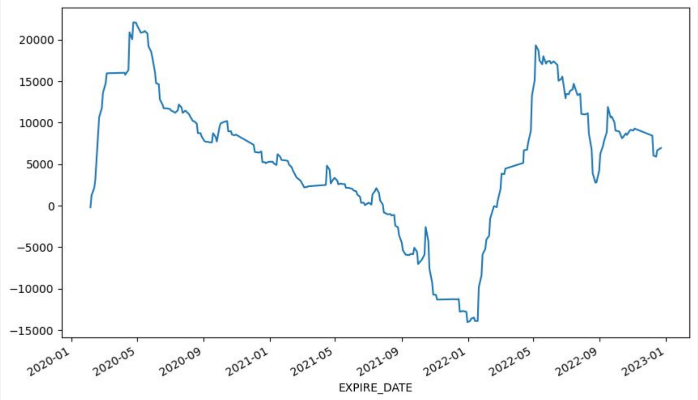
Call vs Put



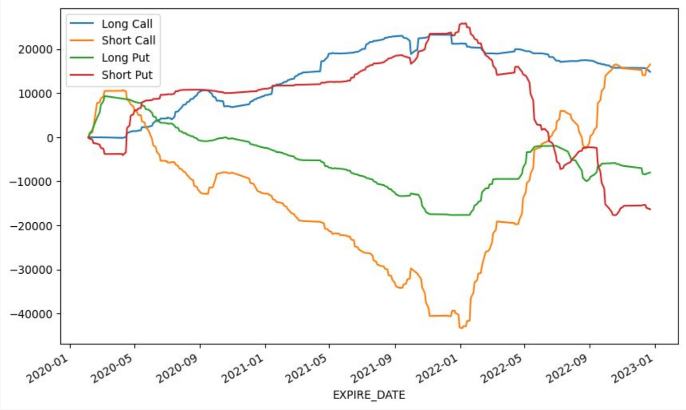
SPY (P/L)

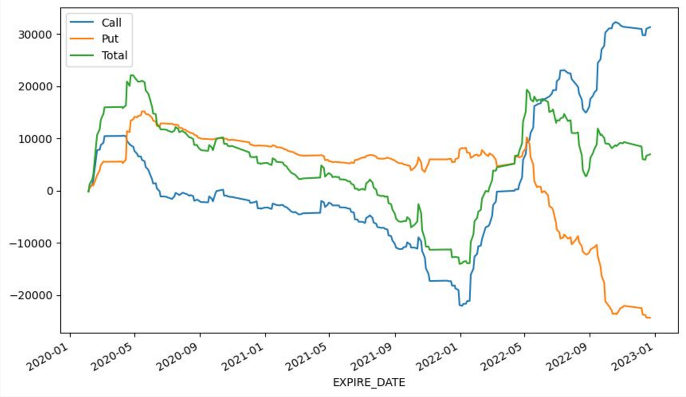
Daily/Total





Call vs Put





Lessons Learnt

- SPX has better returns, probably due to the lower trade volume.
- Lower liquidity = Higher volatility = More gains :-)
- SPX = 10 * SPY
- CRR & JR gives similar results

THANKS!