BT 4016
Risk Analytics for Financial Services
Semester II AY22/23

Group 16

Group Members:
Edwin Lim Ze Xin (A0203808J)
Heng Jee Joo (A0196695E)
Tianbo Sun (A0210985B)
Ng Chek Khau (A0234465A)
Tan Chong Ren (A0234511U)

Table of Contents

1. Basic Analytics and Visualisations of the Portfolio

a. Retrieve the daily prices of the 4 cryptos online (2020-2-1 to 2022-2-1) and calculate the daily log returns. (The same process as how we retrieved the stock prices in the tutorial.)

```
symbols = ["BTC-USD", "ETH-USD", "DOGE-USD", "MATIC-USD"]
cryptos = get_data_for_multiple_cryptos(symbols, start_date='2020-02-01',end_date= '2022-02-02')
cryptos['BTC-USD'].dropna(inplace = True)
cryptos['ETH-USD'].dropna(inplace = True)
cryptos['DOGE-USD'].dropna(inplace = True)
cryptos['MATIC-USD'].dropna(inplace = True)

BTC = cryptos['BTC-USD']
ETH = cryptos['ETH-USD']
DOGE = cryptos['DOGE-USD']
MATIC = cryptos['MATIC-USD']
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

```
log_returns = pd.DataFrame({'BTC-USD':BTC['log_return'],
                            'ETH-USD':ETH['log_return'],
                            'DOGE-USD':DOGE['log_return'],
                            'MATIC-USD':MATIC['log_return']})

log_returns
```

Date	BTC-USD	ETH-USD	DOGE-USD	MATIC-USD
2020-02-01	0.004518	0.019316	0.011628	0.022230
2020-02-02	-0.005178	0.026559	0.018814	0.041880
2020-02-03	-0.005456	0.006592	-0.011410	0.144471
2020-02-04	-0.012185	-0.003242	0.025492	-0.044297
2020-02-05	0.046028	0.076176	0.031074	0.014285
...	...	...	...	...
2022-01-28	0.017248	0.049945	0.002891	0.053536
2022-01-29	0.009321	0.019437	0.009835	0.009334
2022-01-30	-0.005800	0.002454	-0.025380	-0.054332
2022-01-31	0.014804	0.032057	0.016596	0.023755
2022-02-01	0.006737	0.037899	0.005808	0.002231

732 rows × 4 columns

b. For this period, plot the asset return correlations using a matrix heatmap, and plot the annualized Sharpe ratios of the four crypto assets using a bar chat.

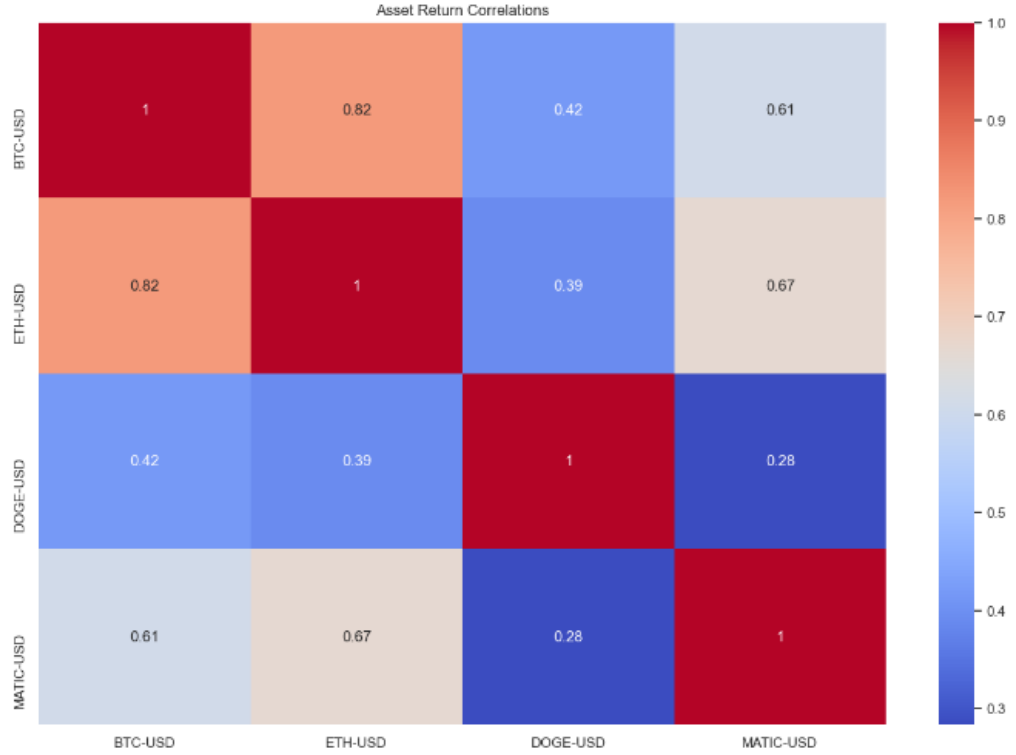

Figure 1: Asset Returns correlations of 4 assets

We annualised the sharpe ratio with 365 days instead of the usual 250 since cryptocurrencies trade 24/7 so there are 365 trading days in a year.

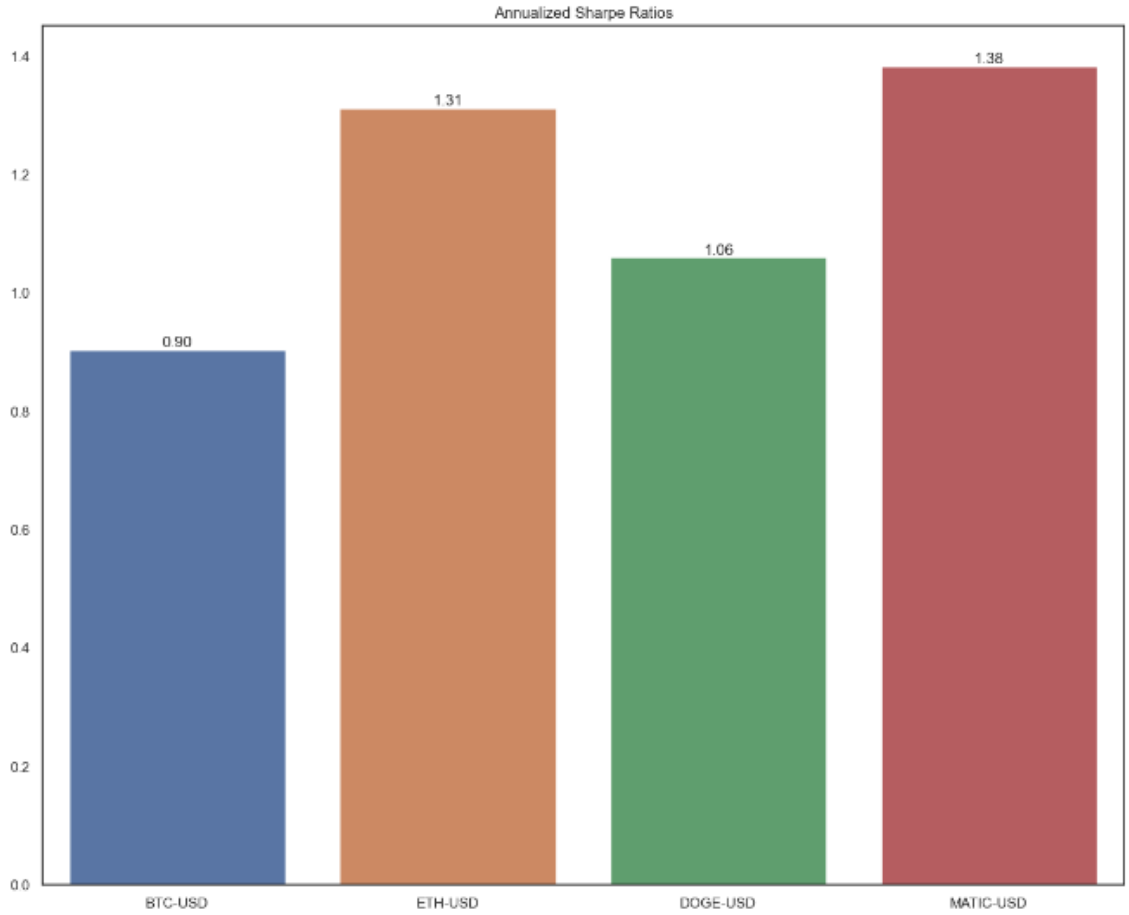

Figure 2: Bar Graph of annualised Sharpe Ratios

c. What can you observe? (< 100 words)

From the correlation matrix, it can be observed that most of the cryptocurrencies have a higher correlation with BTC as compared to the others, which is expected since BTC has the highest market capitalisation in the crypto market. As for the Sharpe Ratios, it seems that MATIC has the highest potential return given the same level of risk out of all the cryptocurrencies, followed by ETH. This could be explained by their smaller market capitalisation as compared to BTC and better establishment and fundamentals compared to DOGE, resulting in a higher potential for large growth.

2. Markowitz MVO and simulated portfolio value

a. Assume we are long only, with total weights equal to 1, run a Mean-Variance Optimization on the crypto assets to minimize risk. Plot the resultant holding weights for two constraint settings, one with the maximum holding weight limit to 0.5, and one with no limit on maximum holding weight. So you will have two graphs, each has 4 lines of different colors. Each line has 731 values between 0-1, representing the holding weight for that asset on each day.

We utilised the cvxpy library to compute the optimal weights for each day for both the restricted and unrestricted portfolio. In both cases, we used a rolling window of the past 183 days of data of returns of each asset to determine the optimal weight of asserts for that day. We chose 183 days (semi-annual) since MATIC-USED only started trading in Apr 2019, and there is no data before that so a 365-day rolling window is not possible. In both cases, we assumed an initial capital of 1million USD.

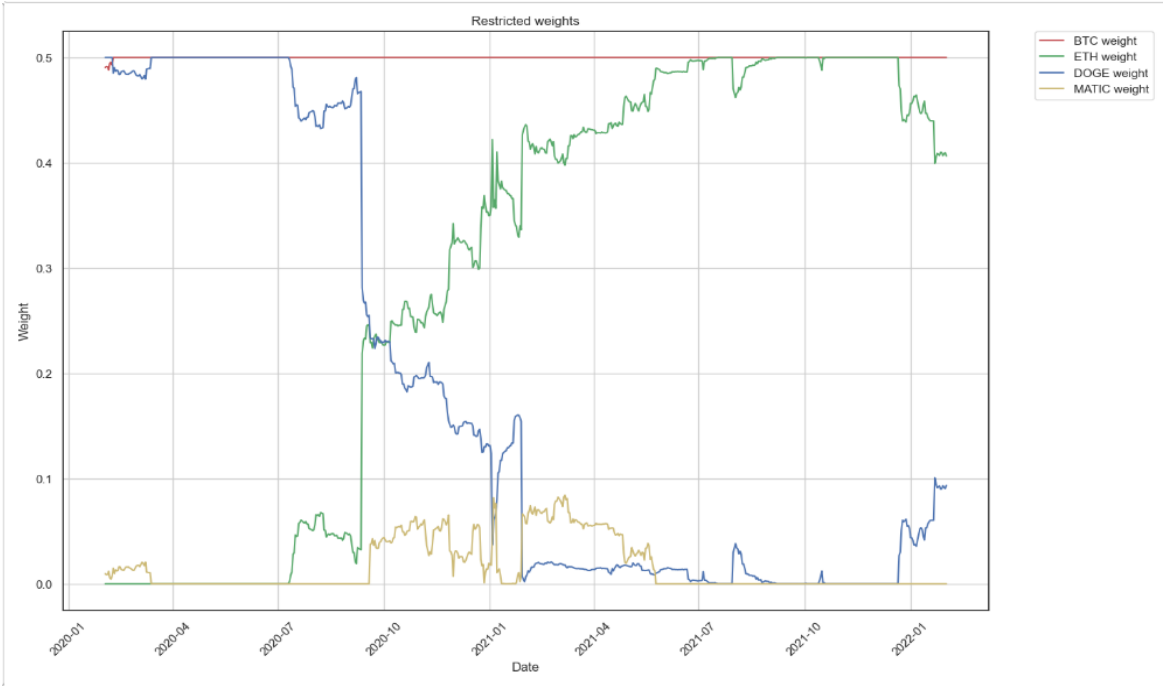

Figure 3: Graph of restricted holding weights (0.5)

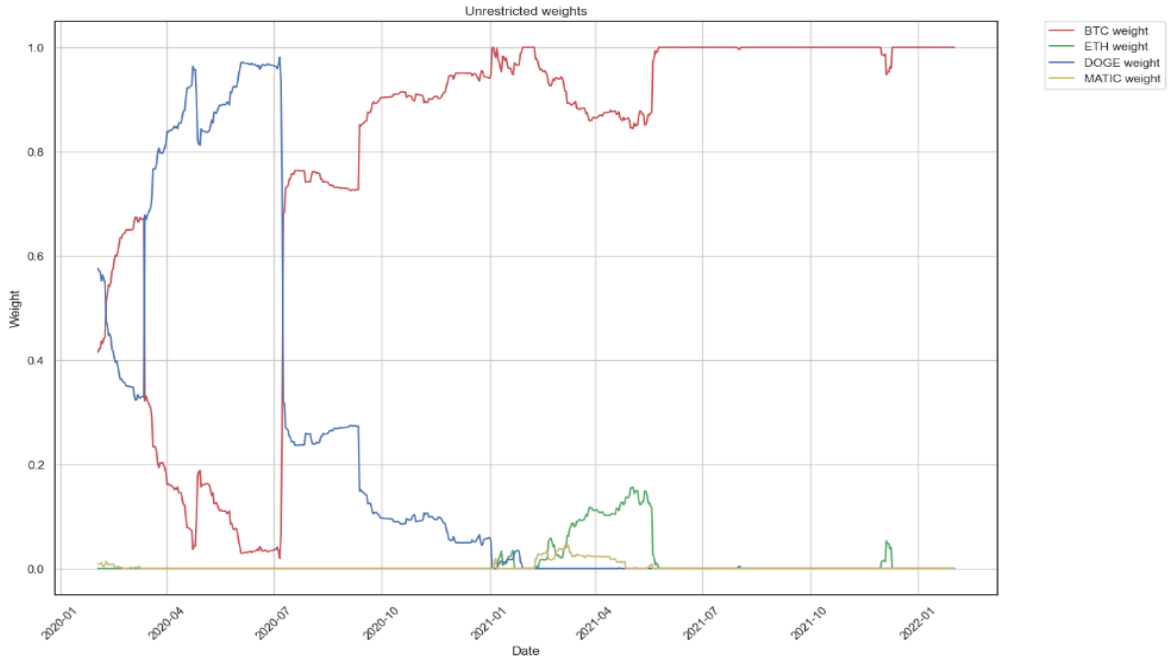

Figure 4: Graph of unrestricted holding weights

b. Report the annualized returns and volatilities for the two portfolio settings and compare the results. (comparison<50 words)

```
annualised_returns = (np.exp(portfolio_df["log_return"].sum()/len(portfolio_df["log_return"]) * 365) -1) * 100
annualised_volatility = portfolio_df['log_return'].std()* 365 ** 0.5 * 100
print("annualised % compound return of restricted portfolio is "  + str(annualised_returns))
print("annualised % volatility of restricted portfolio is "  + str(annualised_volatility))

annualised_returns_unrestricted = (np.exp(portfolio_df_unrestricted["log_return"].sum()/len(portfolio_df_unrestricted["log_returr
annualised_volatility_unrestricted = portfolio_df_unrestricted['log_return'].std()* 365 ** 0.5 * 100

print("annualised % compound return of unrestricted portfolio is "  + str(annualised_returns_unrestricted ))
print("annualised % voltaility of restricted portfolio is "  + str(annualised_volatility_unrestricted ))
```

```
annualised % compound return of restricted portfolio is 184.35210530046186
annualised % volatility of restricted portfolio is 83.20874876859608
annualised % compound return of unrestricted portfolio is 168.1593859090962
annualised % voltaility of restricted portfolio is 81.5445010735886
```

Annualised compound return and volatility of restricted is slightly higher than unrestricted. Annualised volatility of restricted portfolio is unexpectedly higher at 83% compared to unrestricted at 81%. This could be due to having other more volatile cryptocurrencies in the restricted while unrestricted portfolio consists of BTC only in the long run, which also contributes to the higher returns.

3. Calculate the VaR(5%) for the component assets

a. At α = 5% level, estimate daily VaR and ES of the percentage return (not value) of: all the individual component crypto assets and the portfolio (no limit to holding weights setting). Plot and show the VaR and ES curves for one year (2021-2-1 to 2022-2-1). So you will have 5 graphs, each has two lines of different colors (VaR and ES).

We performed the Historical Simulation method to calculate the VaR and ES at α = 5% level with a rolling window of 183 days (semi-annually). In order to get valid values for the first 183 days, we fetched more data from yahoo finance to get data up to 183 days before 1 Feb 2020. We chose 183 days as we are unable to using a rolling window of 365 since MATIC-USD only started trading in Apr 2019, so there is no information on its trading price before this period.

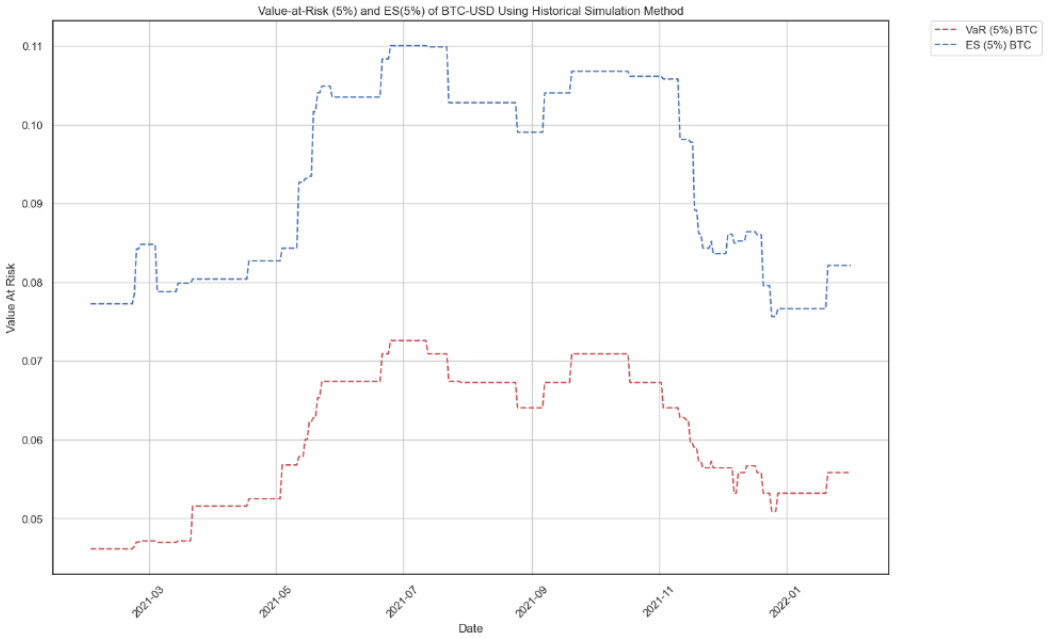

Figure 5: VaR and ES graph of BTC

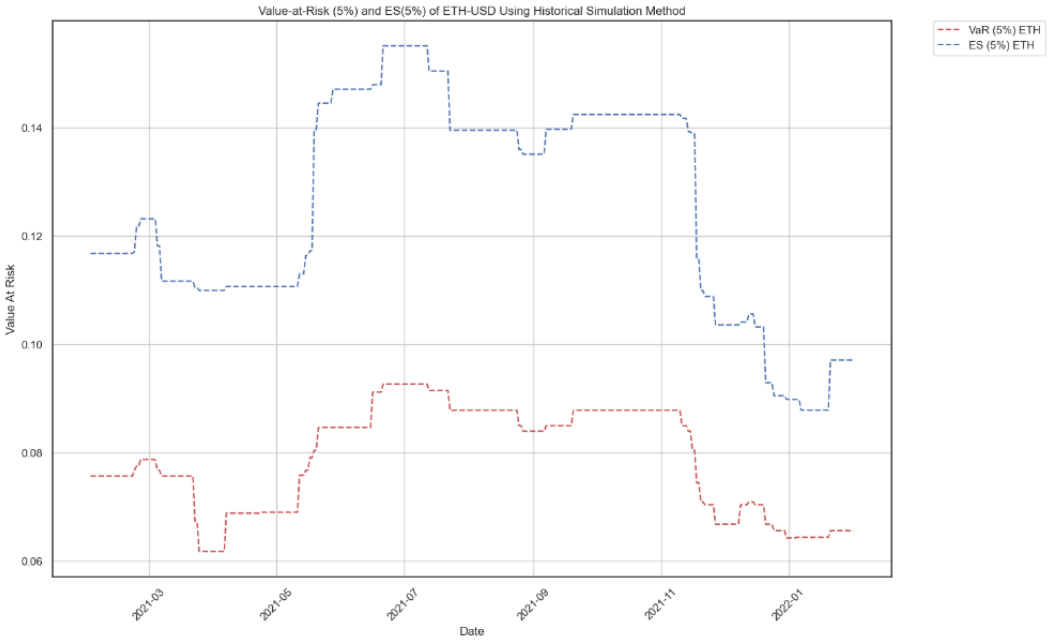

Figure 6: VaR and ES graph of ETH

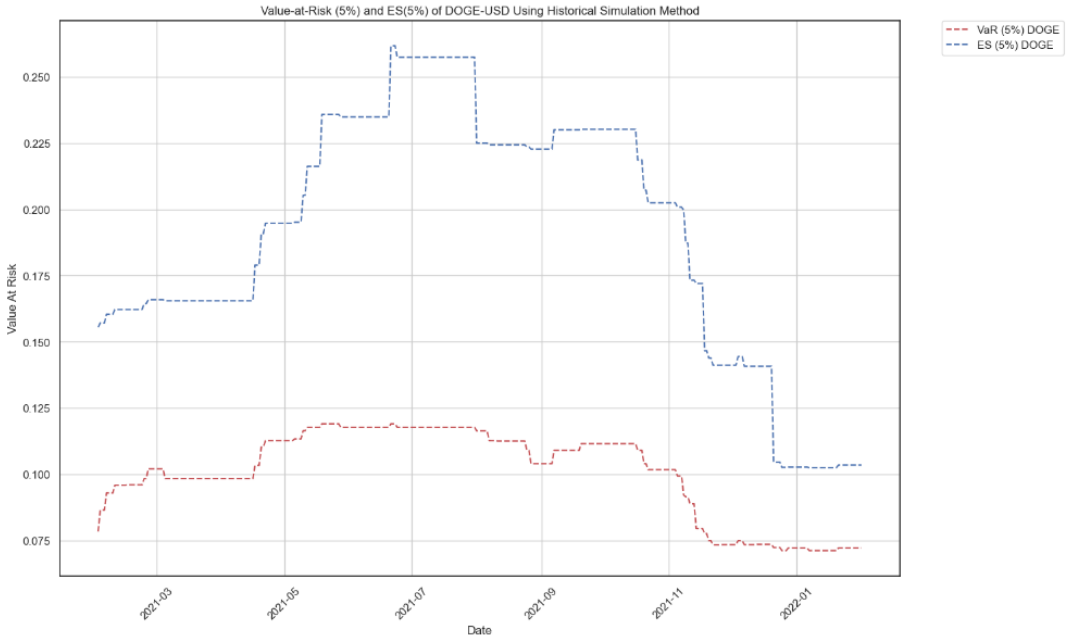

Figure 7: VaR and ES graph of DOGE

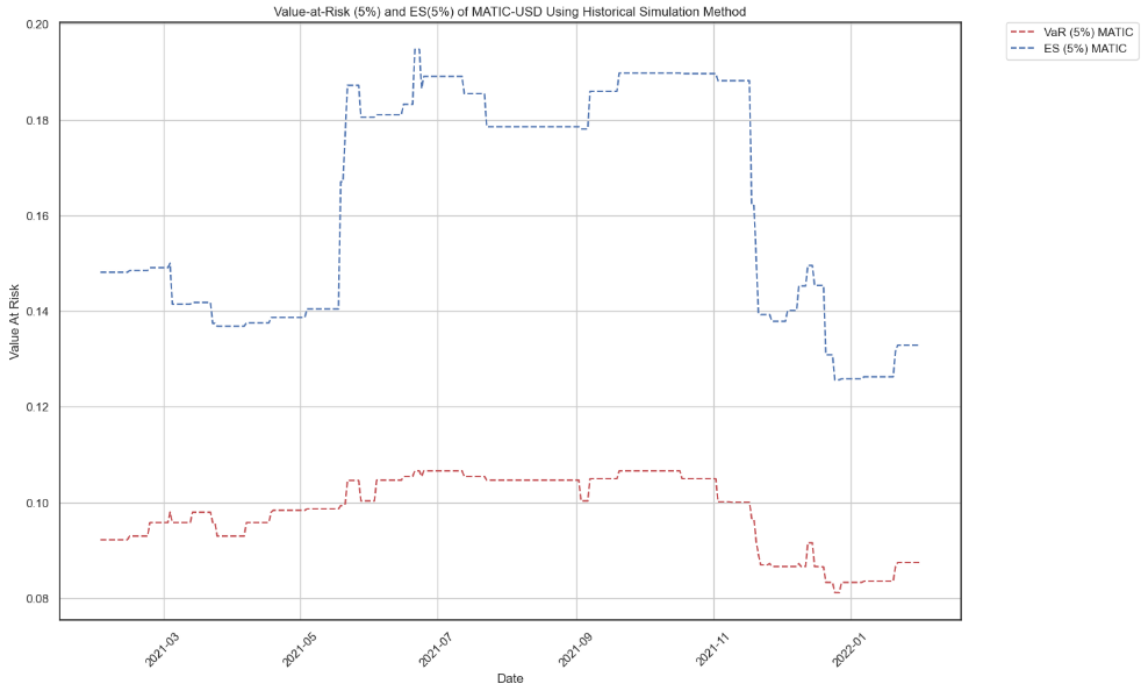

Figure 8: VaR and ES graph of MATIC

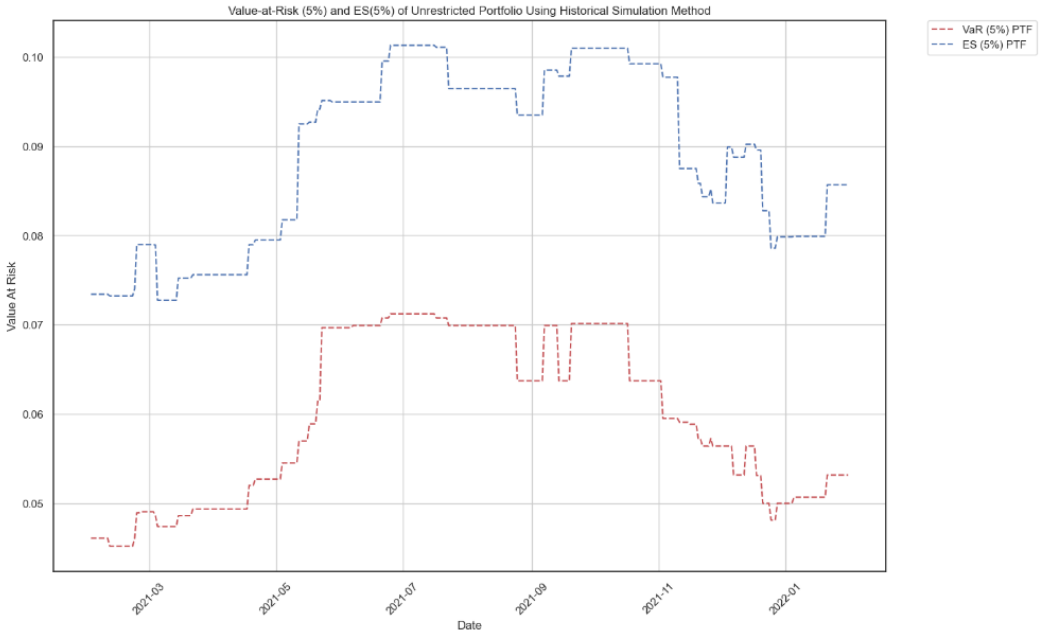

Figure 9: VaR and ES graph of Portfolio

b. **Identify the riskiest component asset from the above evidence and justify your choice. (<100 words)**

Using the unrestricted weights for the unrestricted portfolio as well as plotting the other assets, the riskiest component from the graphs above is DOGE-USD due to its highest percentage of VaR and ES as seen in the graph. MATIC comes close at second by having an ES and VaR percentage of more than 0.15 for the year 2021 to 2022, but DOGE still has the highest risk by averaging more than 0.15 but peaking at 0.22, the highest among the other assets.

4. Hedge by Options

a. Compare Bitcoin (BTC-USD) and Ether (ETH-USD), which coin's prices do you think is more volatile? (You can choose to also consider the upside and downside risks separately.)

From our calculations,

ETH log volatility annualized percentage: 104.31%
BTC log volatility annualized percentage: 78.56 %

ETH has a higher log volatility annualized percentage than BTC so it is more volatile.
This conclusion is also supported by overall higher VaR we observed in Q3.

b. For your no limit setting portfolio from Q2, on 2021/01/01, for each one unit of the asset your team deemed to be the most volatile, buy one straddle (buy one put + buy one call). That expires on 2021/01/08. You can choose the exercise prices as close as to minus/plus one standard deviation of asset return (minus one standard deviation for buying a put and plus one standard deviation for buying a call option). You can determine your cost using the last transacted option prices on 2021/01/01 as the market price. Suppose your cost is externally financed and to be returned/deducted on 2021/01/08, with no interest rate.
What are the instrument_ids that you decided to buy?

Before proceeding with the question, the following assumptions were made:

- Options contracts are non-retradeable
- Simple closing price is used for each contract each day
- Exercise price of put option does not have to be equal to exercise price of call option
- Risk-free rate is zero

Firstly, the ETH options contracts were filtered based on those with a start date of 2021/01/01 and an expiry date of 2021/01/08.

```
startDate = '2021-01-01' #<----PARAM
endDate = '2021-01-08' #<----PARAM

# reverse merge since option dates might be missing (not trading everyday)
optionDF = optionDF_raw[optionDF_raw.ticker=='ETH-USD']
optionDF = optionDF[optionDF.expDate == endDate]

#filter for options with a price on startDate (tradeable), can keep only startDate data for option
optionDF = optionDF[optionDF.Date==startDate]
# issue: each option could get traded multiple times, this raw file is tick data
optionDF = optionDF.drop_duplicates(subset=['Date','instrument_id'],keep='last')

optionDF.instrument_id.nunique()


# cross-merge on instrument_id vs ETH, so every instrument_id is populated for each Date with all necessary data
# create backbone DF first
df1 = ETH_raw[(ETH_raw.Date >= startDate) & (ETH_raw.Date <= endDate)][['Date']]
df2 = optionDF[['instrument_id']]
optionDF_ts_full = df1[['Date']].merge(df2[['instrument_id']],how='cross')
```

A dataframe was then generated to include the daily ETH options contract (instrument_id, exercise price, type, price, expiry date) as well as the ETH price. To factor in the volatility of ETH price, the upward and downward ETH prices were calculated. A zero dummy was also created to consider the case where the option contract was not exercised which would lead to zero payoff.

```
optionDF_ts_full = optionDF_ts_full.merge(optionDF[['instrument_id','exPX','type','price','expDate']],on='instrument_id'
                                          ,how='left')
optionDF_ts_full = optionDF_ts_full.merge(ETH_raw[['Date','price_USD']],on='Date',how='left')
# note original price is in corresponding crypto, converting
optionDF_ts_full['price_option_USD'] = optionDF_ts_full['price']*optionDF_ts_full['price_USD']

# price of option and price of underlying at purchase
optionDF_ts_full = optionDF_ts_full.sort_values(by='Date')
optionDF_ts_full['price_option_first'] = optionDF_ts_full.groupby('instrument_id')['price_option_USD'].transform('first')
optionDF_ts_full['price_underlying_first'] = optionDF_ts_full.groupby('instrument_id')['price_USD'].transform('first')

print(ETH_price_vol)
# for filtering later # ETH_return_vol # ETH_price_vol
optionDF_ts_full['PX_minus_vol'] = optionDF_ts_full['price_underlying_first'] - ETH_price_vol
optionDF_ts_full['PX_plus_vol'] = optionDF_ts_full['price_underlying_first'] + ETH_price_vol
optionDF_ts_full['zero_dummy'] = 0.0
```

Using the generated dataframe, the optimal call and put contracts were identified based on the calculated profits and losses. The instrument ids to buy for an optimal straddle would thus be **ETH-USD-210108-700-P** and **ETH-USD-210108-740-C**

```
# filter and keep, PUT
optionDF_P = optionDF_ts_full[optionDF_ts_full['type']=='P']
optionDF_P = optionDF_P[(optionDF_P.Date >= startDate) & (optionDF_P.Date <= endDate)]

optionDF_P["PnL"] = (optionDF_P['exPX'] - optionDF_P['price_USD'])
optionDF_P["PnL"] = optionDF_P[["PnL",'zero_dummy']].max(axis=1) - optionDF_P['price_option_first']

mask_P_lower_bound = optionDF_P['exPX'] >= optionDF_P['PX_minus_vol']
mask_P_upper_bound = optionDF_P['exPX'] <= optionDF_P['price_underlying_first']

optionDF_P = optionDF_P[mask_P_lower_bound & mask_P_upper_bound]
optionDF_P_end = optionDF_P[optionDF_P.Date==endDate]

print(optionDF_P_end.reset_index(drop=True).loc[optionDF_P_end["PnL"].argmax(),])
optimal_P_row = optionDF_P_end.reset_index(drop=True).loc[optionDF_P_end["PnL"].argmax(),]
```

```
Date                    2021-01-08
instrument_id    ETH-USD-210108-700-P
exPX                           700.0
type                               P
price                         0.0255
expDate                 2021-01-08
price_USD                1224.197144
price_option_USD           31.217027
price_option_first         18.624373
price_underlying_first    730.367554
PX_minus_vol              690.013062
PX_plus_vol               770.722045
zero_dummy                       0.0
PnL                       -18.624373
```

```
# filter and keep, CALL
optionDF_C = optionDF_ts_full[optionDF_ts_full['type']=='C']
optionDF_C = optionDF_C[(optionDF_C.Date >= startDate) & (optionDF_C.Date <= endDate)]

optionDF_C["PnL"] = (-optionDF_C['exPX'] + optionDF_C['price_USD'])
optionDF_C["PnL"] = optionDF_C[["PnL",'zero_dummy']].max(axis=1) - optionDF_C['price_option_first']

mask_P_lower_bound = optionDF_C['exPX'] >= optionDF_C['price_underlying_first']
mask_P_upper_bound = optionDF_C['exPX'] <= optionDF_C['PX_plus_vol']
optionDF_C = optionDF_C[mask_P_lower_bound & mask_P_upper_bound]
optionDF_C_end = optionDF_C[optionDF_C.Date==endDate]

print(optionDF_C_end.reset_index(drop=True).loc[optionDF_C_end["PnL"].argmax(),])
optimal_C_row = optionDF_C_end.reset_index(drop=True).loc[optionDF_C_end["PnL"].argmax(),]
```

```
Date                    2021-01-08
instrument_id    ETH-USD-210108-740-C
exPX                           740.0
type                               C
price                           0.06
expDate                 2021-01-08
price_USD                1224.197144
price_option_USD           73.451829
price_option_first         43.822053
price_underlying_first    730.367554
PX_minus_vol              690.013062
PX_plus_vol               770.722045
zero_dummy                       0.0
PnL                       440.37509
```

c. **Report the two-year portfolio performance (annualized return and Sharpe ratio) with or without this one-off straddle hedging.**

Portfolio	Compound Annualized Return	Sharpe Ratio
Without straddle hedging	168.16%	2.06
With straddle hedging	168.16%	2.06

```
# with straddle, option size - one straddle for one unit underlying asset
portfolio_df_unrestricted_straddled = portfolio_df_unrestricted.copy()

notional_underlying = ETH_raw[ETH_raw.Date ==startDate].notional.values[0]
print('underlying coin quantity of unrestricted port on',startDate,notional_underlying)
straddle_PnL = (optimal_P_row.PnL + optimal_C_row.PnL) * notional_underlying

portfolio_df_unrestricted_straddled.loc[portfolio_df_unrestricted_straddled.Date == endDate,'Portfolio_value'] += straddle_PnL

portfolio_df_unrestricted_straddled['Prev value'] = portfolio_df_unrestricted_straddled['Portfolio_value'].shift(1)
portfolio_df_unrestricted_straddled['log_return'] = np.log(portfolio_df_unrestricted_straddled['Portfolio_value']/portfolio_df_ur
portfolio_df_unrestricted_straddled['perc_return'] = (portfolio_df_unrestricted_straddled['Portfolio_value']/portfolio_df_unrestr


annualised_returns_unrestricted_original = (np.exp(portfolio_df_unrestricted_original["log_return"].sum()/len(portfolio_df_unrest
annualised_volatility_unrestricted_original = portfolio_df_unrestricted_original['log_return'].std()* 365 ** 0.5 * 100
sharpe_ratio_unrestricted_original = annualised_returns_unrestricted_original/annualised_volatility_unrestricted_original
print("annualised % compound return of original unrestricted portfolio is "  + str(annualised_returns_unrestricted_original))
print("sharpe ratio of original unrestricted portfolio is "  + str(sharpe_ratio_unrestricted_original))


annualised_returns_unrestricted_straddled = (np.exp(portfolio_df_unrestricted_straddled["log_return"].sum()/len(portfolio_df_unre
annualised_volatility_unrestricted_straddled = portfolio_df_unrestricted_straddled['log_return'].std()* 365 ** 0.5 * 100
sharpe_ratio_unrestricted_straddled = annualised_returns_unrestricted_straddled/annualised_volatility_unrestricted_straddled
print("annualised % compound return of straddled unrestricted portfolio is "  + str(annualised_returns_unrestricted_straddled))
print("sharpe ratio of straddled unrestricted portfolio is "  + str(sharpe_ratio_unrestricted_straddled))
```

```
underlying coin quantity of unrestricted port on 2021-01-01 -1.986732020270357e-16
annualised % compound return of original unrestricted portfolio is 168.16036712638927
sharpe ratio of original unrestricted portfolio is 2.062191175347263
annualised % compound return of straddled unrestricted portfolio is 168.16036712638987
sharpe ratio of straddled unrestricted portfolio is 2.0621911753472673
```

To generate the portfolio with straddle hedging, the ETH notional size was calculated to derive the straddle P&L, which would ultimately determine the portfolio value. From there, we calculated the annualized return of the portfolio and the sharpe ratio. Since the ETH notional

size is very small (-1.9867e-16), there is not much difference between the annualized return and sharpe ratio for both portfolios.

5. Design your own hedging strategies

a. For your no limit setting portfolio from Q2, on any date throughout the two years, you are allowed to decide your favored frequency, and buy or sell any quantity of options as long as that option has a trading record on that date based on the data file. (Again, similar to Q4, assuming that you can successfully buy the instrument of any quantity at the last transacted prices. E.g., if your strategy decide to buy one call at 4:00PM, you may use the price at 3:58PM as the market price.) Clearly explain your strategy: what is the intuition? why do you design it in such a way.

For question 5, there are certain things to consider and factor so we will go through them systematically.

Firstly, to understand more about our options available, we looked into the trading date range over the past 2 years as well as the expiry of the options list available for us

```
# note original price is in corresponding crypto
optionDF_raw['price'] = optionDF_raw['price'].astype(np.float)

optionDF_raw = optionDF_raw.sort_values(by='Date').reset_index(drop=True)
print('trade date range',optionDF_raw['Date'].min(),optionDF_raw['Date'].max())
print('exp date range',optionDF_raw['expDate'].min(),optionDF_raw['expDate'].max())

BTC_raw['ticker'] = 'BTC-USD'
ETH_raw['ticker'] = 'ETH-USD'
coin_raw = pd.concat([BTC_raw,ETH_raw])

# filter to align and merge
optionDF_raw = optionDF_raw[(optionDF_raw.Date>=startDate) & (optionDF_raw.Date<=endDate)]
optionDF_raw = optionDF_raw.merge(coin_raw[['Date','ticker','notional','price_USD']],\
                                  on=['Date','ticker'],how='left')
```

```
trade date range 2020-01-27 2021-12-01
exp date range 2020-02-07 2022-03-25
```

For 2 assets (ETH and BTC), we also checked for the size of data available to us, and further check on the unique instrument id to comprehend the data better.

```
# work out possibility space
print("List of all instruments available: {}".format(optionDF_raw.instrument_id.size))
print("possibility space is: {}".format(optionDF_raw.instrument_id.nunique()))
```

```
List of all instruments available: 100322
possibility space is: 4180
```

There are certain considerations set based on question 5.

1) The weight optimisation for Q2 earlier must be used here as well.
2) We are allowed to trade at any favoured frequency but we assumed that
   - Cash is limited to the portfolio position of Q2
   - Buy/Sell option to further level up is allowed.

The also introduce the question of risk control. Since any frequency is allowed, we should not utilise a strategy to buy all available asset that we deem fit and not have any liquidity for subsequent trading days throughout the 2 years period. Our group has thus look into the idea of doing risk control by looked at the tenor of the options presented to us which determined the max spend ratio as well.

Our strategy revolves around the use of Black-Scholes Formula to compare with the option pricing given to us and attempt to capture any option mispricing. We used the volatility that is calculated with a rolling window of 183 (semi-annual) once again to calculate the theoretical option value.

```python
def black_scholes(S, K, T, r, sigma, option_type):
    """
    Computes the Black-Scholes price of an option.

    Parameters:
    S (float): Current stock price
    K (float): Strike price
    T (float): Time to expiration (in years)
    r (float): Risk-free interest rate
    sigma (float): Volatility of the underlying asset
    option_type (str): 'C' or 'P'

    Returns:
    float: The Black-Scholes option price
    """

    d1 = (math.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*math.sqrt(T))
    d2 = d1 - sigma*math.sqrt(T)

    if option_type == 'C':
        option_price = S*norm.cdf(d1) - K*math.exp(-r*T)*norm.cdf(d2)
    elif option_type == 'P':
        option_price = K*math.exp(-r*T)*norm.cdf(-d2) - S*norm.cdf(-d1)
    else:
        raise ValueError("Option type must be 'call' or 'put'.")

    return option_price
```

For simplicity. We will do utilise the strategy with the following rules and steps

1) Compute of the option in each row is done to determine if option valuation is above or below the market price.
2) We assumed that the risk-free rate used on the Black-Scholes option price is 0
3) The focus of the strategy is a buy and hold onto call and puts. Hence, we do not look into selling call and puts to reduce complexity and complications in the strategy
4) We assumed that no writing or selling option is done to other counterparty, so there is only long call/put option only.
5) We assumed that there is no margin on option.
6) We disregard daily mark-to-market on option value, only calculate exercise and non-exercise value.
7) Although portfolio buy and sell comes from the portfolio, we are only looking at 1 contract here, so it will not break what is available in the portfolio, so we would run a simple check to see if the portfolio balance is enough to fund the option that is undervalued by the market. If yes, we buy, else we skip that contract.

We proceed to calculate the valuation of all the options on the available date, and assign an undervalued tag to long if price option – valuation < 0. To test the code is working we also printed every 1000th option valuation to sense check its value.

```python
# step 1, calculate valuation of all options on each available date
# and assign a long-only tag: if undervalued, buy and hold
for i in optionDF_raw.index:
    row = optionDF_raw.loc[i,]

    # using rolling vol
    if row['ticker'] == 'BTC-USD':
        return_vol =  BTC_raw_full[BTC_raw_full.Date == row.Date]['return_vol'].values[0]
    elif row['ticker'] == 'ETH-USD':
        return_vol = ETH_raw_full[ETH_raw_full.Date == row.Date]['return_vol'].values[0]

    # S - stock price, K - strike price, T - year to expiration,
    # r - risk-free rate, sigma - underlying asset volatility, option_type C or P
    # black_scholes(S, K, T, r, sigma, option_type)
    optionDF_raw.loc[i,'option_valuation'] = black_scholes(row['price_USD'], row['exPX'], row['tenor'], 0.000001, return_vol/100,
    if i%1000 == 0:
        print(i,'option_valuation',optionDF_raw.loc[i,'option_valuation'])


optionDF_raw['price_option_minus_valuation'] = optionDF_raw['price_option_USD'] - optionDF_raw['option_valuation']
optionDF_raw['price_option_minus_valuation_abs'] = optionDF_raw['price_option_minus_valuation'].abs()
optionDF_raw['underValued'] = optionDF_raw['price_option_minus_valuation'] < 0
```

```
0 option_valuation -3179.4765527611708
1000 option_valuation 1606.7764904751693
2000 option_valuation 0.0
```

```
<ipython-input-7-cfa05c755096>:55: RuntimeWarning: divide by zero encountered in double_scalars
  d1 = (math.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma*math.sqrt(T))
```

```
3000 option_valuation 0.0
4000 option_valuation 0.0
5000 option_valuation 1.1134198393755526
6000 option_valuation 114.58672860975824
7000 option_valuation 0.0
8000 option_valuation 74.58674482893304
9000 option_valuation 95.55466177694237
10000 option_valuation 3073.4669550941762
11000 option_valuation 4323.467333861281
12000 option_valuation 25.554658735846488
```

Next, for the undervalued options that we have identified, we proceed to sort them by date and set the price option by the first instance of each row grouped by instrument id, and calculated in USD based on the exercise prices done in Q4. Filtering the results shows that we have 736 contracts identified to be undervalued and can be implemented in our strategy.

```
# for every underValued option, find its PnL on expDate
# buy on first underValued date, need price_option_first processing as well
optionDF_underValued = optionDF_raw[optionDF_raw['underValued']]
optionDF_underValued = optionDF_underValued.sort_values(by='Date')
optionDF_underValued['zero_dummy'] = 0.0

# remember here we have holding period coin exposure, so this is different vs Q4 (paying coin to get crypto)
optionDF_underValued['price_option_first'] = optionDF_underValued.groupby('instrument_id')['price'].transform('first')
optionDF_underValued['price_option_first_USD'] = optionDF_underValued['price_option_first']*optionDF_underValued['price_USD']

# to see only PnL on expiration
optionDF_underValued = optionDF_underValued[optionDF_underValued.Date == optionDF_underValued.expDate]

optionDF_underValued = optionDF_underValued.reset_index(drop=True)
# 736 contracts
```

```
print(optionDF_underValued[['trade_id','instrument_id','ticker','expDate','exPX','type','price_option_first','price_option_first_
```

```
    trade_id           instrument_id   ticker     expDate     exPX type  \
0        238   BTC-USD-200501-8500-C  BTC-USD  2020-05-01   8500.0    C
1         86   BTC-USD-200501-8000-P  BTC-USD  2020-05-01   8000.0    P
2        156   BTC-USD-200501-7000-P  BTC-USD  2020-05-01   7000.0    P
3        153   BTC-USD-200501-6500-P  BTC-USD  2020-05-01   6500.0    P
4         99   BTC-USD-200501-8500-P  BTC-USD  2020-05-01   8500.0    P
..       ...                     ...      ...         ...      ...  ...
731        3  BTC-USD-211201-57000-C  BTC-USD  2021-12-01  57000.0    C
732        3  BTC-USD-211201-56500-P  BTC-USD  2021-12-01  56500.0    P
733        2  BTC-USD-211201-56500-P  BTC-USD  2021-12-01  56500.0    P
734        1  BTC-USD-211201-56500-P  BTC-USD  2021-12-01  56500.0    P
735        1  BTC-USD-211201-56000-P  BTC-USD  2021-12-01  56000.0    P

     price_option_first  price_option_first_USD
0                0.0300              265.942998
1                0.0005                4.432383
2                0.0005                4.432383
3                0.0005                4.432383
4                0.0005                4.432383
..                  ...                     ...
731              0.0030              171.689484
732              0.0015               85.844742
733              0.0015               85.844742
734              0.0015               85.844742
735              0.0005               28.614914
```

We proceed to compute the profit and loss for each of the rows above. The profit is either the difference in the exercise price and the price of the asset in the exercise by the right call (long/short) or worthless upon expiry. To get a quick check on whether the strategy worked, we also printed the sum of the profit and loss on all the undervalued contracts identified by our model and compute the sum.

```
#   exPX in [price_USD-ETH_vol, price_USD] if P
#   exPX in [price_USD, price_USD+ETH_vol] if C
optionDF_underValued['PnL'] = 0 # initialize
for i in optionDF_underValued.index:
    row = optionDF_underValued.loc[i,]

    if row.type == 'P':
        row["PnL"] = (row['exPX'] - row['price_USD'])
        row["PnL"] = row[["PnL",'zero_dummy']].max() - row['price_option_first_USD']

    elif row.type == 'C':
        row["PnL"] = (-row['exPX'] + row['price_USD'])
        row["PnL"] = row[["PnL",'zero_dummy']].max() - row['price_option_first_USD']

    optionDF_underValued.loc[i,"PnL"] = row["PnL"].copy()

# for quick sense checks
print('optionDF_underValued.PnL.sum()',optionDF_underValued.PnL.sum())
```

```
optionDF_underValued.PnL.sum() 145116.95052499388
```

The total flattened PnL of one contract each amount to $145,116, which is positive and a good indication of the strategy working.

For a better understanding of the period in which of strategy impacted our portfolio, we cumulatively summed the Profit and Loss and printed the date where the profit and loss impacted our portfolio.

```
optionDF_underValued['PnL_sized'] = optionDF_underValued['PnL'] * optionDF_underValued['notional']
optionDF_underValued_pivoted = optionDF_underValued.pivot_table(index='Date',values='PnL_sized',aggfunc='sum').reset_index()

print('days when this strategy impact portfolio')
print(optionDF_underValued_pivoted)
```

```
days when this strategy impact portfolio
         Date     PnL_sized
0   2020-05-01  8.185826e+04
1   2020-08-01  3.316898e+06
2   2020-09-01  8.436454e+04
3   2021-01-01  1.180196e+04
4   2021-02-01 -4.218705e+05
5   2021-03-01  1.555535e+07
6   2021-04-01 -1.955469e+04
7   2021-05-01  2.247720e+05
8   2021-08-01  1.489156e-01
9   2021-09-01  1.879699e+06
10  2021-10-01  4.624184e+05
11  2021-11-01 -1.679907e+00
12  2021-12-01 -8.211654e+04
```

b. Simulate, present, and evaluate your portfolio's performance with your strategy (highlight the annualized return and Sharpe ratio, you can also choose to plot the portfolio value curve).

Here, we created a new data frame portfolio_df_unrestricted_overlay that keeps track of our profit and loss arising from our option trading, separate from our original unrestricted portfolio. After calculating the profits and losses from our option strategy, we combined this value with our original portfolio value on each day to arrive at a data frame portfolio_df_unrestricted_COMBINED that considers the value of original portfolio as well as the profits from our option trading strategy.

To get an evaluation of the portfolio's performance with our strategy, we recalculated the annualised returns and sharpe ratio of the portfolio. Assuming there is no portfolio impact before the option exercise and looking at 2020-02-01 to 2022-02-01, we overlay the option performance on our portfolio, and calculated the annualised returns and sharpe ratio.

```
# b) performance evaluation
# assuming no portfolio value impact before option exercise

portfolio_df_unrestricted = pd.read_csv('portfolio_df_unrestricted_updated.csv',index_col=0)
#2020-2-1 to 2022-2-1
portfolio_df_unrestricted['Date'] = ETH_raw['Date']

# overlaying option performance
portfolio_df_unrestricted_overlay = portfolio_df_unrestricted.copy()
portfolio_df_unrestricted_overlay = portfolio_df_unrestricted_overlay.merge(optionDF_underValued_pivoted,on='Date',how='left')
portfolio_df_unrestricted_overlay.loc[0,'PnL_sized_cum'] = 0  # the added part
portfolio_df_unrestricted_overlay['PnL_sized_cum'] = portfolio_df_unrestricted_overlay['PnL_sized_cum'].fillna(method='ffill')
portfolio_df_unrestricted_overlay['Portfolio_value'] += portfolio_df_unrestricted_overlay['PnL_sized_cum']


portfolio_df_unrestricted_overlay['Prev value'] = portfolio_df_unrestricted_overlay['Portfolio_value'].shift(1)
portfolio_df_unrestricted_overlay['log_return'] = np.log(portfolio_df_unrestricted_overlay['Portfolio_value']/portfolio_df_unrest
portfolio_df_unrestricted_overlay['perc_return'] = (portfolio_df_unrestricted_overlay['Portfolio_value']/portfolio_df_unrestricte
```

```
portfolio_df_unrestricted_COMBINED = portfolio_df_unrestricted_overlay.copy()

annualised_returns_unrestricted_COMBINED = (np.exp(portfolio_df_unrestricted_COMBINED["log_return"].sum()/len(portfolio_df_unrest
annualised_volatility_unrestricted_COMBINED = portfolio_df_unrestricted_COMBINED['log_return'].std()* 365 ** 0.5 * 100
sharpe_ratio_unrestricted_COMBINED = annualised_returns_unrestricted_COMBINED/annualised_volatility_unrestricted_COMBINED
print("annualised % compound return of strategy unrestricted portfolio is "  + str(annualised_returns_unrestricted_COMBINED))
print("sharpe ratio of strategy unrestricted portfolio is "  + str(sharpe_ratio_unrestricted_COMBINED))
```

```
annualised % compound return of strategy unrestricted portfolio is 457.4926246261862
sharpe ratio of strategy unrestricted portfolio is 4.013586179349791
```

Based on the results, we see that our strategy was able to yield and annualised compound percentage return of around 457% and have a good sharpe ratio value of around 4.01, which is a positive sign that our strategy works and makes for a good portfolio strategy.

We also plotted the graph of our portfolio value against the date range to better understand how our portfolio value changes due to the strategy implemented.

In [25]:
```
portfolio_df_unrestricted_COMBINED
```

Out[25]:

	Portfolio_value	Prev value	log_return	perc_return	Date	PnL_sized	PnL_sized_cum
0	1.009192e+06	NaN	NaN	NaN	2020-02-02	NaN	0.000000e+00
1	1.001657e+06	1.009192e+06	-0.007495	-0.007467	2020-02-03	NaN	0.000000e+00
2	1.010893e+06	1.001657e+06	0.009179	0.009221	2020-02-04	NaN	0.000000e+00
3	1.049453e+06	1.010893e+06	0.037435	0.038144	2020-02-05	NaN	0.000000e+00
4	1.071942e+06	1.049453e+06	0.021203	0.021430	2020-02-06	NaN	0.000000e+00
...	...	...	...	...	...	...	...
726	3.068819e+07	3.056684e+07	0.003962	0.003970	2022-01-28	NaN	2.359156e+07
727	3.075465e+07	3.068819e+07	0.002163	0.002166	2022-01-29	NaN	2.359156e+07
728	3.071322e+07	3.075465e+07	-0.001348	-0.001347	2022-01-30	NaN	2.359156e+07
729	3.081943e+07	3.071322e+07	0.003452	0.003458	2022-01-31	NaN	2.359156e+07
730	3.086830e+07	3.081943e+07	0.001584	0.001585	2022-02-01	NaN	2.359156e+07

731 rows × 7 columns

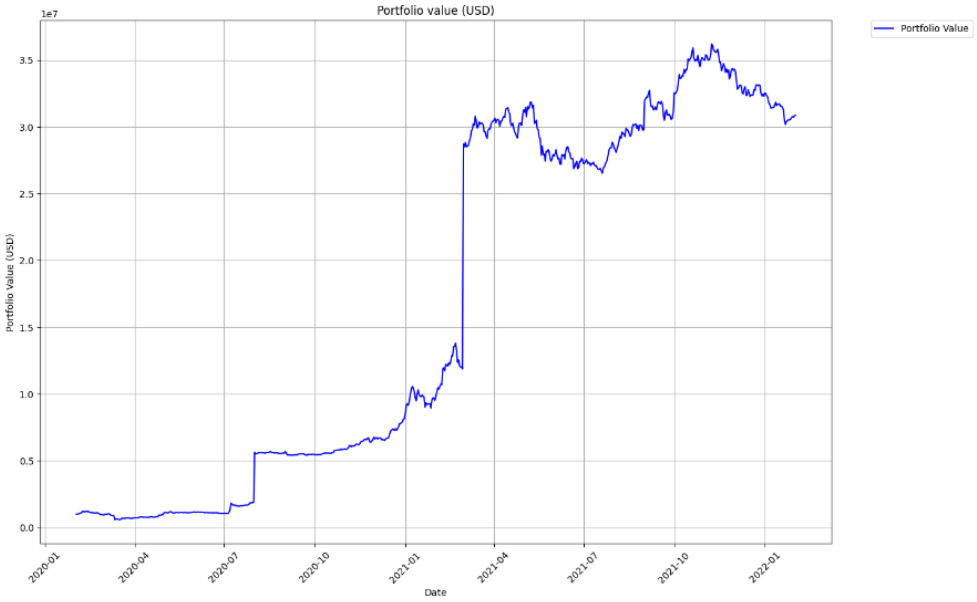

Figure 10: Portfolio Value after implementing strategy.