Crypto-trading

Idea and Methodology

By Clifford

Idea

- Aim: Create an automated rebalancing strategy for managing cryptocurrencies portfolio

Criterial of strategy:

1. Able to rebalance the weight of assets through a model
2. import past data(i.e close data, return) to the model and output a optimal portfolio weight

Idea

Criterial of programme:

1. Able to retrieve cryptos' data through api and record data
2. Able to buy or sell cryptos automatically by following the optimal portfolio
3. Generate buy or sell signal to user, notifying them amount to buy or sell of certain type of cryptos
4. recording the account balance and amount of cryptos holding daily
5. Generate graphs of accumulated return for better data visualization

Methodology: Tools to use

Programme: Python

- Easy coding
- Can import lots of libraries

Platform: jupyter

Methodology: Tools to use

Programme: Python

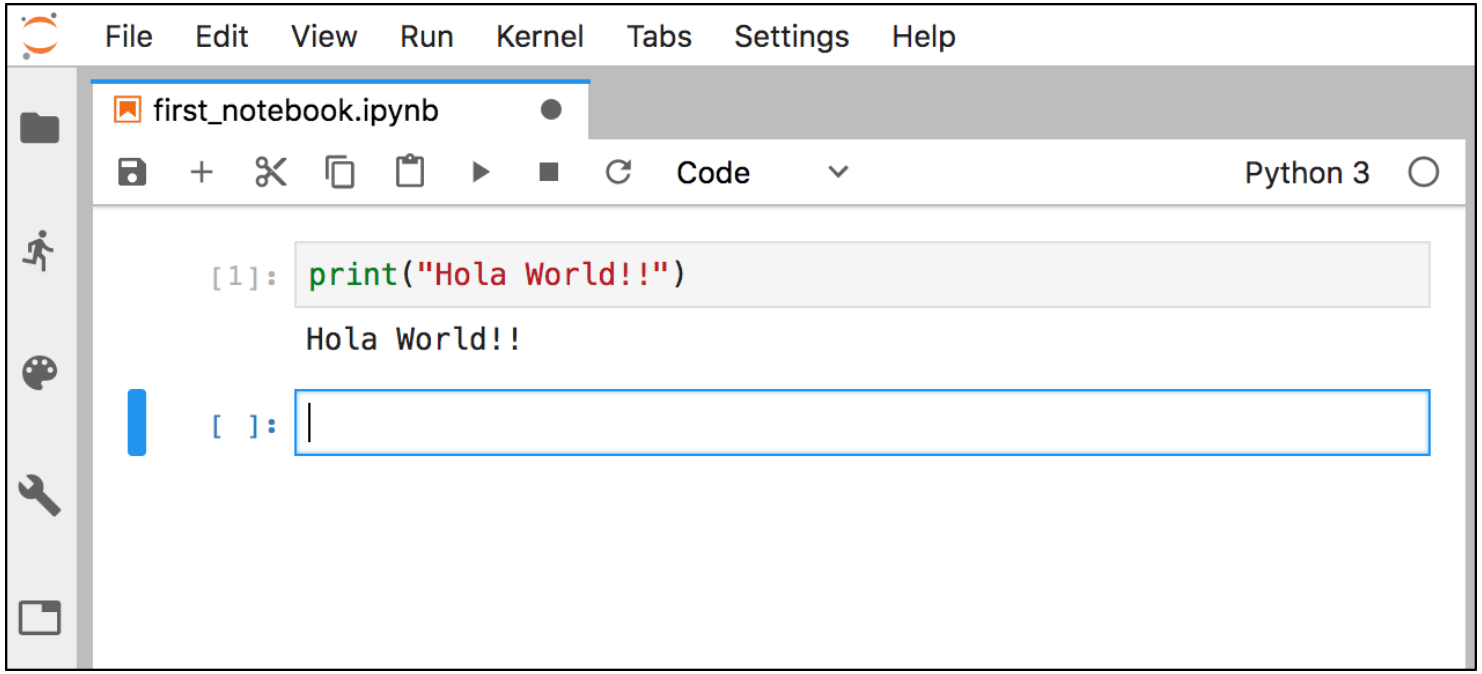
Easy coding

Can import lots of libraries

Platform: jupyter notebook

Portable

Able to run on cell, easy for testing



Methodology: getting data from api

Use python library 'requests' to get data from api link

```
import pandas as pd
import requests

#importing data from url with list of cryptos info
headers = {
        'x-api-key' : "SwWOMt7bUi12i31OpjeAaayb2KQ99GP41EPANKOO",
}
list_url='https://83i8cudoh7.execute-api.ap-east-1.amazonaws.com/dev/cryptos'

#get the crypto_id
list_info = requests.get(list_url, headers=headers).json()
crypto_id = pd.json_normalize(list_info,'data')
crypto_id
```

```
api_url = "https://83i8cudoh7.execute-api.ap-east-1.amazonaws.com/dev/crypto/"
prod_url = []
for i in crypto_id['product_id']:
        prod_url.append(f'{api_url}{i}')

#get the info of respective cryptos with its crypto_id
eth = requests.get(prod_url[0],headers=headers).json()
btc = requests.get(prod_url[1],headers=headers).json()
usdc = requests.get(prod_url[2],headers=headers).json()

#get the daily return of respective cryptos and save it as pandas DataFrame format
btc_d = pd.DataFrame(btc['data']['dod']).T
eth_d = pd.DataFrame(eth['data']['dod']).T
usdc_d = pd.DataFrame(usdc['data']['dod']).T

#get the monthly return of respective cryptos and save it as pandas DataFrame format
btc_m = pd.DataFrame(btc['data']['mom']).T
eth_m = pd.DataFrame(eth['data']['mom']).T
usdc_m = pd.DataFrame(usdc['data']['mom']).T
```

Methodology: way to store data

Use python library 'pandas' to help formatting data as dataframe

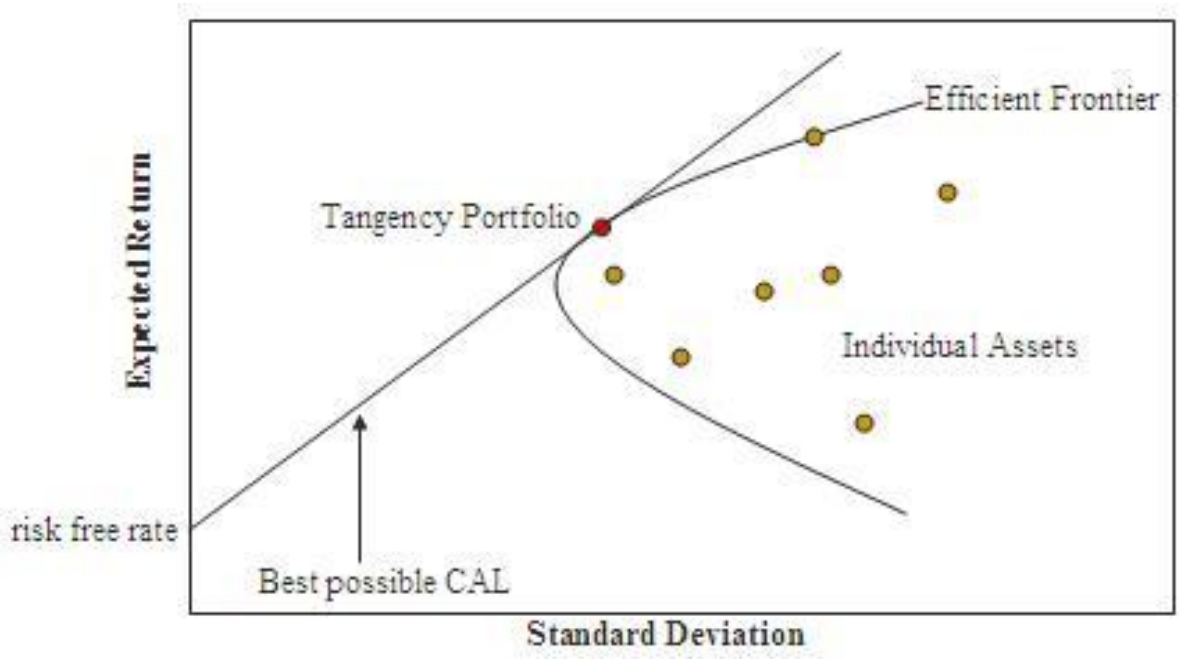
```
import numpy as np
import pandas as pd
#merging the monthly data of cryptos and renaming the columns
basedata = pd.DataFrame(data=btc_m,index=btc_m.index)
basedata = basedata.rename(columns={'amount':'BTC_close','converted_date':'date','mom':'BTC_return'})
newdata = pd.DataFrame(data=eth_m,index=eth_m.index)
newdata = newdata.rename(columns={'amount':'ETH_close','converted_date':'date','mom':'ETH_return'})
basedata = pd.merge(basedata, newdata, on='date')
newdata2 = pd.DataFrame(data=usdc_m,index=usdc_m.index)
newdata2 = newdata2.rename(columns={'amount':'USDC_close','converted_date':'date','mom':'USDC_return'})
basedata = pd.merge(basedata, newdata2, on='date')
basedata['date'] = pd.to_datetime(basedata['date'])
```

Methodology: modelling data

using efficient frontier to find out the optimal asset weight allocation to minimize the risk of the portfolio with highest attainable return

Library 'scipy' have tools for us to find minimize risk

```
from scipy.optimize import minimize
def Efficient_frontier(data,period):
    meanRet = data.mean()
    sigma   = data.cov()
    def get_metrics(w) ->np.array:
        w = np.array(w)
        R = np.sum(meanRet*w)*period
        V = np.sqrt(np.dot(w.T,np.dot(sigma*period,w)))
        SR = R/V
        return np.array([R,V,SR], dtype="object")
    def neg_sharpe(w) ->np.array:
        return get_metrics(w)[2] * -1

    def volatility(w) ->np.array:
        return get_metrics(w)[1]

    def check_sum(w) ->float:
        return np.sum(w) - (1 - portfolio_reserve)

    guess = [0.5,0.5]
    bounds = tuple((0.0,1.0) for i in range(2))
    constraints = ({'type':'eq', 'fun': check_sum})
    optimized_volatility = minimize(volatility,guess,method='SLSQP',bounds=bounds,constraints=constraints)
    return optimized_volatility.x
```

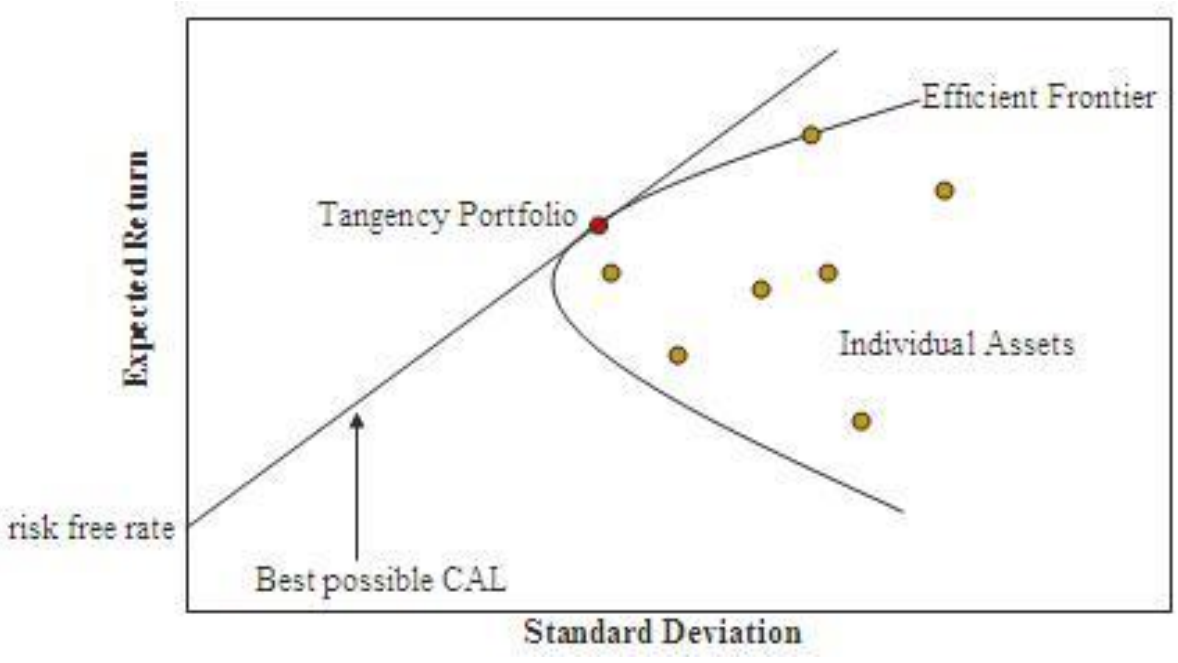


Methodology: modelling data

The efficient frontier to used to calculate weights of Bitcoin and Ethereum as their price movement is similar. USD Coin is used as capital reserve in this simulation as it has very minimum price change.

Assuming no commission taken during trading

```
from scipy.optimize import minimize
def Efficient_frontier(data,period):
    meanRet = data.mean()
    sigma   = data.cov()
    def get_metrics(w) ->np.array:
        w = np.array(w)
        R = np.sum(meanRet*w)*period
        V = np.sqrt(np.dot(w.T,np.dot(sigma*period,w)))
        SR = R/V
        return np.array([R,V,SR], dtype="object")
    def neg_sharpe(w) ->np.array:
        return get_metrics(w)[2] * -1

    def volatility(w) ->np.array:
        return get_metrics(w)[1]

    def check_sum(w) ->float:
        return np.sum(w) - (1 - portfolio_reserve)

    guess = [0.5,0.5]
    bounds = tuple((0.0,1.0) for i in range(2))
    constraints = ({'type':'eq', 'fun': check_sum})
    optimized_volatility = minimize(volatility,guess,method='SLSQP',bounds=bounds,constraints=constraints)
    return optimized_volatility.x
```



Methodology: generate signals

After finding optimal asset weights, the program will determine whether to buy or to sell and setting amount for buying or selling.

The program will print out the info to signal users after executing

```
def log(signal, symbol, amount, price, date):
    data = [date, signal, symbol, amount, price]
    if signal == 'Buy':
        print(f"At {date} : Buy {amount} {symbol} @${price}")
    elif signal == 'Sell':
        print(f"At {date} : Sell {amount} {symbol} @${price}")
    log_data.append(data)
```

```
At 2021-06-30 : Sell 0.3340914 Bitcoin @$35045.0
At 2021-06-30 : Buy 5.0452356 Ethereum @$2275.679931640625
At 2021-06-30 : Buy 226.9144707 USD Coin @$0.9998999834060669
```

```
At 2021-07-02 : Buy 0.4408709 Bitcoin @$33786.55078125
At 2021-07-02 : Sell 6.7683605 Ethereum @$2154.1298828125
At 2021-07-02 : Sell 315.6115075 USD Coin @$0.9998999834060669
```

Methodology: record trades

At the end of the day, the program will record that day's data and print out log of the portfolio

```
#initializing the portfolio
def portfolio_init():
    data = pd.DataFrame(columns=[['date','BTC_price','BTC_amount','ETH_price','ETH_amount','USDC_price','USDC_amount','Total_value']])
    empty_list = ['','0.0','0.0','0.0','0.0','0.0','0.0','0.0']
    data.loc[len(data)] = empty_list
    return data
#updating new data to portfolio
def portfolio_update(df,new_list):
    new_list_series = pd.Series(new_list, index=df.columns)
    df = df.append(new_list_series, ignore_index=True)
    return df
#printing latest info of portfolio
def portfolio_log(value):
    crypto = ["Bitcoin","Ethereum','USD Coin']
    symbol = ['BTC','ETH','USDC']
    print('-'*80)
    print(f'Portfolio weight at {value[0]}:')
    x = -1
    for i in range(len(crypto)):
        x += 2
        if value[x+1] == 0:
            print(f'{crypto[i]} Amount Held : None')
        else:
            print(f'{crypto[i]} Amount Held : {value[x+1]}{symbol[i]} at @${value[x]}')
    print(f'Total value of the portfolio = ${value[-1]}')
    print('-'*80)
```

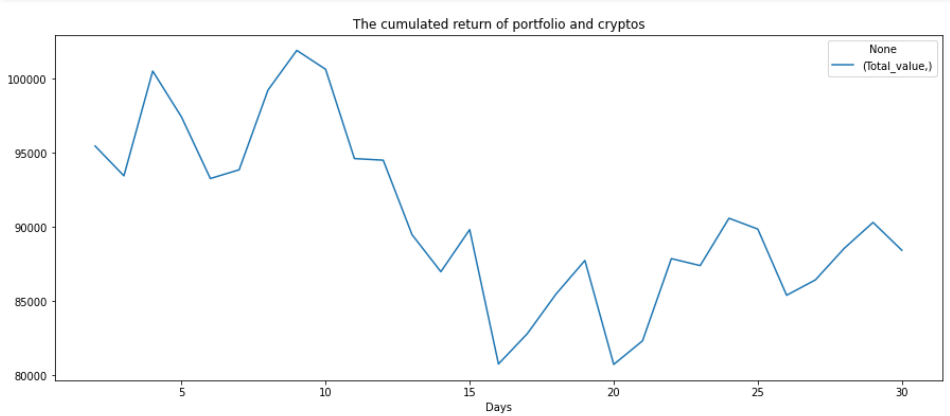
```
--------------------------------------------------------------------------------
Portfolio weight at 2021-06-30:
Bitcoin Amount Held : 1.33955900120410O8BTC at @$35045.0
Ethereum Amount Held : 12.09405053393679ETH at @$2275.679931640625
USD Coin Amount Held : 8274.942441505242USDC at @$0.9998999834060669
Total value of the portfolio = $82741.1480994725
--------------------------------------------------------------------------------
```

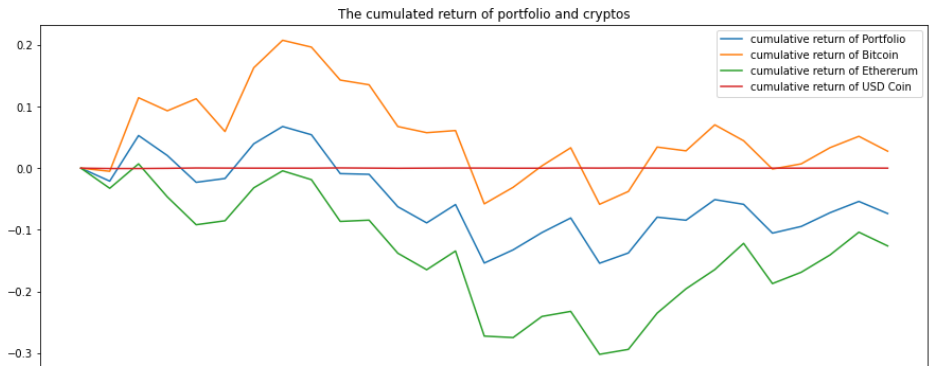
```
--------------------------------------------------------------------------------
Portfolio weight at 2021-07-01:
Bitcoin Amount Held : 1.33955900120410O8BTC at @$33504.69140625
Ethereum Amount Held : 12.09405053393679ETH at @$2106.409912109375
USD Coin Amount Held : 8274.942441505242USDC at @$0.9998999834060669
Total value of the portfolio = $78630.65368799125
--------------------------------------------------------------------------------
```

Methodology: creating graphs

Use library 'matplotlib' to plot the value of portfolio and cumulated return

```
import matplotlib.pyplot as plt
def draw_graph(days):
    global save_portfolio
    graph_portfolio = pd.DataFrame(data=save_portfolio.iloc[1+days:])
    graph_portfolio[['Total_value']].plot(figsize = (15,6))
    plt.title("The cumulated return of portfolio and cryptos")
    plt.ylabel("USD ($)")
    plt.xlabel("Days")
    graph_return = pd.DataFrame()
    fig,ax = plt.subplots(figsize = (15,6))
    graph_return[['daily_return']] = graph_portfolio[['Total_value']]/graph_portfolio[['Total_value']].shift() - 1
    graph_return[['cumulated_return']] = np.cumprod(1 + graph_return[['daily_return']]) - 1

    graph_return[['btc_daily_return']] = graph_portfolio[['BTC_price']]/graph_portfolio[['BTC_price']].shift() - 1
    graph_return[['btc_cumulated_return']] = np.cumprod(1 + graph_return[['btc_daily_return']]) - 1

    graph_return[['eth_daily_return']] = graph_portfolio[['ETH_price']]/graph_portfolio[['ETH_price']].shift() - 1
    graph_return[['eth_cumulated_return']] = np.cumprod(1 + graph_return[['eth_daily_return']]) - 1

    graph_return[['usdc_daily_return']] = graph_portfolio[['USDC_price']]/graph_portfolio[['USDC_price']].shift() - 1
    graph_return[['usdc_cumulated_return']] = np.cumprod(1 + graph_return[['usdc_daily_return']]) - 1

    graph_return = graph_return.replace(np.nan,0)
    ax.plot(graph_return[['cumulated_return']], label = 'cumulative return of Portfolio')
    ax.plot(graph_return[['btc_cumulated_return']], label = 'cumulative return of Bitcoin')
    ax.plot(graph_return[['eth_cumulated_return']], label = 'cumulative return of Ethererum')
    ax.plot(graph_return[['usdc_cumulated_return']], label = 'cumulative return of USD Coin')
    plt.legend()
    plt.title("The cumulated return of portfolio and cryptos")
    plt.ylabel("Cumulated return (%)")
    plt.xlabel("Days")
```





Methodology: backtesting

Using last 30 days price change to calculate optimal weights and simulate the trades

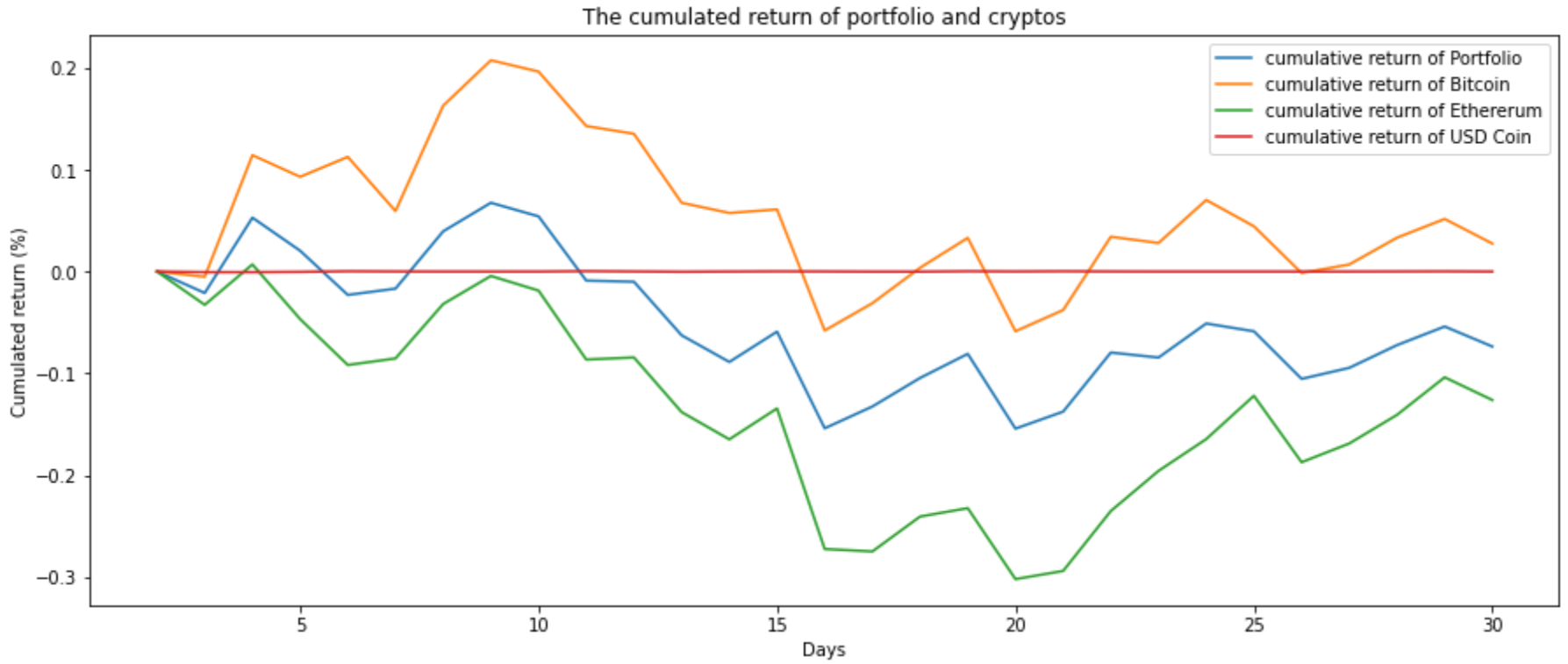
The bascktest also allows few customize like when to do trades, amount to reserve for trade

```
save_portfolio  =  portfolio_init()
start_backtest(trade_when,reserve,data_range)
```

```
Portfolio weight at 2021-07-01:
Bitcoin Amount Held : 1.454707184448861BTC at @$33504.69140625
Ethereum Amount Held : 13.133652332589447ETH at @$2106.409912109375
USD Coin Amount Held : 8986.254588068252USDC at @$0.9998999834060669
Total value of the portfolio = $85389.72657047084
--------------------------------------------------------------------------------
********************************************************************************
Bitcoin: 75.58500000000001%
Ethereum: 14.415%
USD Coin: 10.0%
********************************************************************************
At 2021-07-02 : Buy 0.4787681 Bitcoin @$33786.55078125
At 2021-07-02 : Sell 7.3501672 Ethereum @$2154.1298828125
At 2021-07-02 : Sell 342.7414 USD Coin @$0.9998999834060669
--------------------------------------------------------------------------------
Portfolio weight at 2021-07-02:
Bitcoin Amount Held : 1.9334752627198495BTC at @$33786.55078125
Ethereum Amount Held : 5.783485104946643ETH at @$2154.1298828125
USD Coin Amount Held : 8643.513188031064USDC at @$0.9998999834060669
Total value of the portfolio = $86426.48693282381
--------------------------------------------------------------------------------
********************************************************************************
Bitcoin: 76.458%
Ethereum: 13.542000000000002%
USD Coin: 10.0%
********************************************************************************
--------------------------------------------------------------------------------
Portfolio weight at 2021-07-03:
Bitcoin Amount Held : 1.9334752627198495BTC at @$34669.12890625
Ethereum Amount Held : 5.783485104946643ETH at @$2226.989990234375
USD Coin Amount Held : 8643.513188031064USDC at @$1.0
Total value of the portfolio = $88555.17974569688
--------------------------------------------------------------------------------
```

Result

From the return performance, the simulated portfolio has a cumulated return between Bitcoin and Ethereum. It is acceptable as the simulation does not allow short action which the return must be negative when both assets' price drop.



Result

Sometimes the backtest suggests full position on either cryptos.
This maybe due to insufficient price data since only last 30 days are counted whil efficient frontier applies on annual data.
Resulting inaccurate output

```
**********
Ethereum: 90.0%
USD Coin: 10.0%
**********
```

Area of improvement

1. Can allow short action while price of bitcoin falls so that profit can still be made
2. Use other strategies can applicable on daily changes or short time interval
3. Allowing adjustment on reserving amount of USD Coin to avoid risk while price of crypto drops