Estratégia de médias móveis.

February 15, 2021

A ideia central desse trabalho é apresentar a estratégia de cruzamento de médias móveis para definir quando comprar e quando vender ações. Para analisar a eficácia de tal estratégia, será feito o backtest da mesma em uma ação do índice bovespa.

[1]:
```
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import yfinance as yf
import pyfolio as pf
import warnings
warnings.filterwarnings('ignore')
```

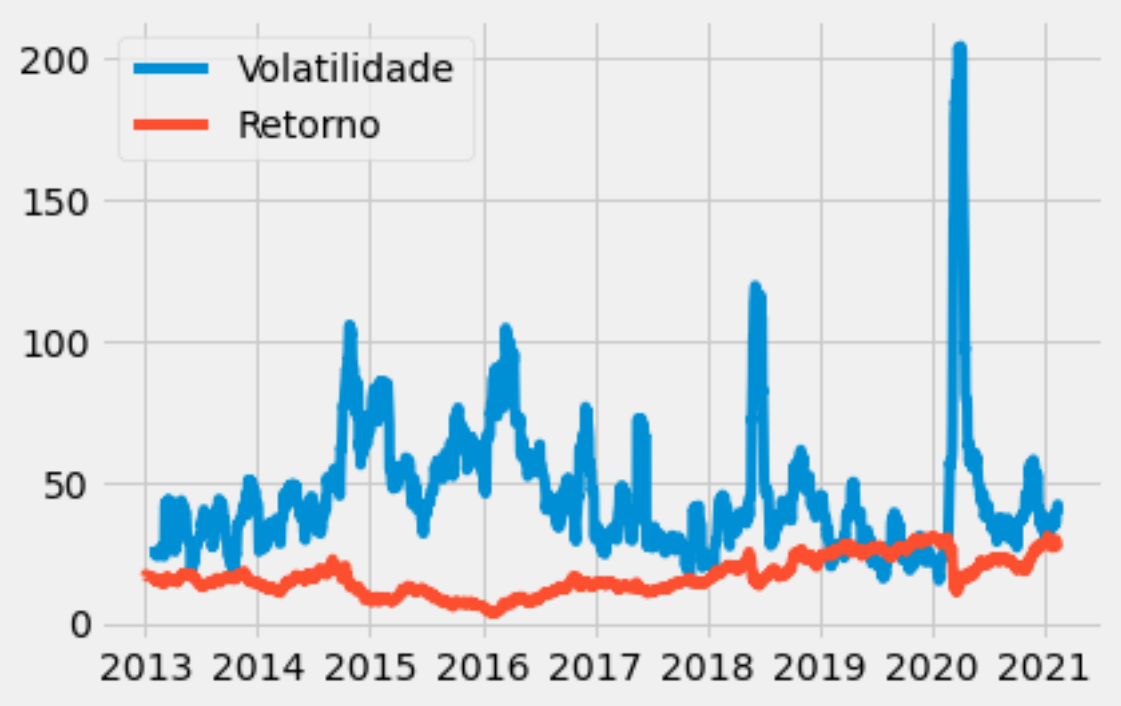
```
C:\Users\Flavio\anaconda3\lib\site-packages\pyfolio\pos.py:26: UserWarning:
Module "zipline.assets" not found; multipliers will not be applied to position
notionals.
  warnings.warn(
```

0.0.1 Definindo os ativos que serão utilizados

Os dados coletados dão inicio em 01-01-2013 com os valores de fechamento ajustado.

[2]:
```
petr = yf.download(tickers='PETR4.SA', start='2013-01-01')[['Adj Close']]
petr

petr['LogReturn'] = np.log(petr['Adj Close'] / petr['Adj Close'].shift(1)) #␣
 ↪Retorno␣
petr['Volatility'] = petr['LogReturn'].rolling(21).std()*np.sqrt(252)


plt.plot(petr['Volatility']*100, label = 'Volatilidade')
plt.plot(petr['Adj Close'], label = 'Retorno')
plt.legend(loc='best')
plt.show()
```

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

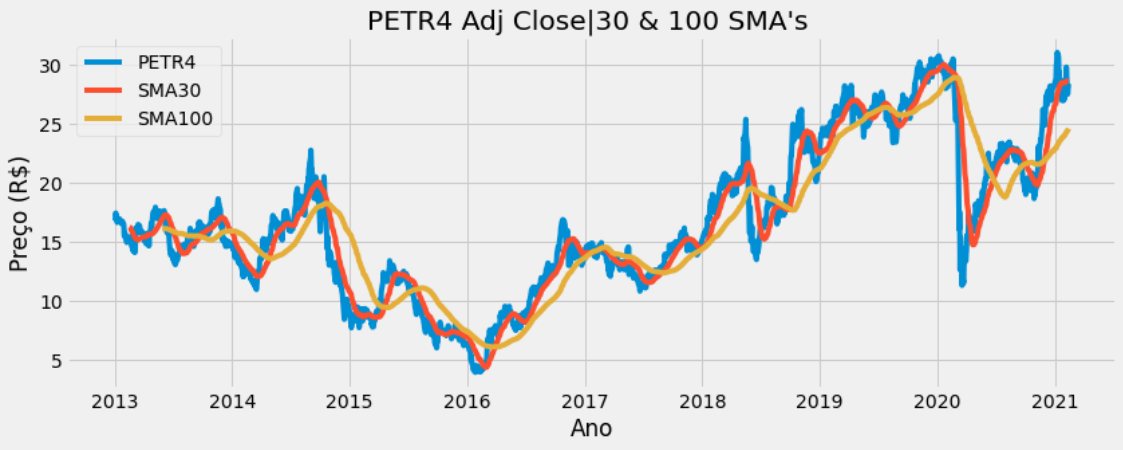


0.0.2 Criando as médias móveis simples

Para desenvolver a análise vamos adicionar ao modelo as médias móveis simples de 30 dias e de 100 dias.

[3]:
```
# Criando a de 30 dias

SMA30 = pd.DataFrame()
SMA30 = petr.rolling(window=30).mean()

# Criando a de 100 dias

SMA100 = pd.DataFrame()
SMA100 = petr.rolling(window=100).mean()
```

[4]:
```
# Vizualizando
plt.figure(figsize = (12.5, 4.5))
plt.plot(petr['Adj Close'], label = 'PETR4')
plt.plot(SMA30['Adj Close'], label = 'SMA30')
plt.plot(SMA100['Adj Close'], label = 'SMA100')
plt.xlabel('Ano')
plt.ylabel('Preço (R$)')
plt.legend(loc = 'best')
plt.title("PETR4 Adj Close|30 & 100 SMA's ")
plt.show()
```



Começando a análise Apartir de agora, criaremos um dataframe para abrigar as 3 variáveis criadas. Após a criação do dataframe, definiremos uma função para dar o sinal quando comprar e quando vender a ação

[5]:
```
data = pd.DataFrame()
data['petr'] = petr['Adj Close']
data['SMA30'] = SMA30['Adj Close']
data['SMA100'] = SMA100['Adj Close']

data
```

[5]:

Date	petr	SMA30	SMA100
2013-01-02	16.807058	NaN	NaN
2013-01-03	17.413103	NaN	NaN
2013-01-04	17.438713	NaN	NaN
2013-01-07	17.139957	NaN	NaN
2013-01-08	16.644878	NaN	NaN
…	…	…	…
2021-02-08	28.110001	28.674000	24.3866
2021-02-09	27.540001	28.682667	24.4408
2021-02-10	27.799999	28.677667	24.5000
2021-02-11	28.080000	28.674333	24.5640
2021-02-12	28.420000	28.679333	24.6315

[2013 rows x 3 columns]

[6]:
```
# Criando a função

def buy_sell(data):
```

```
    sigbuy = []
    sigsell = []
    flag = 1 # Quando as médias cruzarem

    for i in range(len(data)):
        if data['SMA30'][i] > data['SMA100'][i]:
            if flag != 1:
                sigbuy.append(data['petr'][i])
                sigsell.append(np.nan)
                flag = 1
            else:
                sigbuy.append(np.nan)
                sigsell.append(np.nan)

        elif data['SMA30'][i] < data['SMA100'][i]:
                if flag != 0:
                    sigbuy.append(np.nan)
                    sigsell.append(data['petr'][i])
                    flag = 0
                else:
                    sigbuy.append(np.nan)
                    sigsell.append(np.nan)
        else:
                    sigbuy.append(np.nan)
                    sigsell.append(np.nan)

    return(sigbuy, sigsell)
```

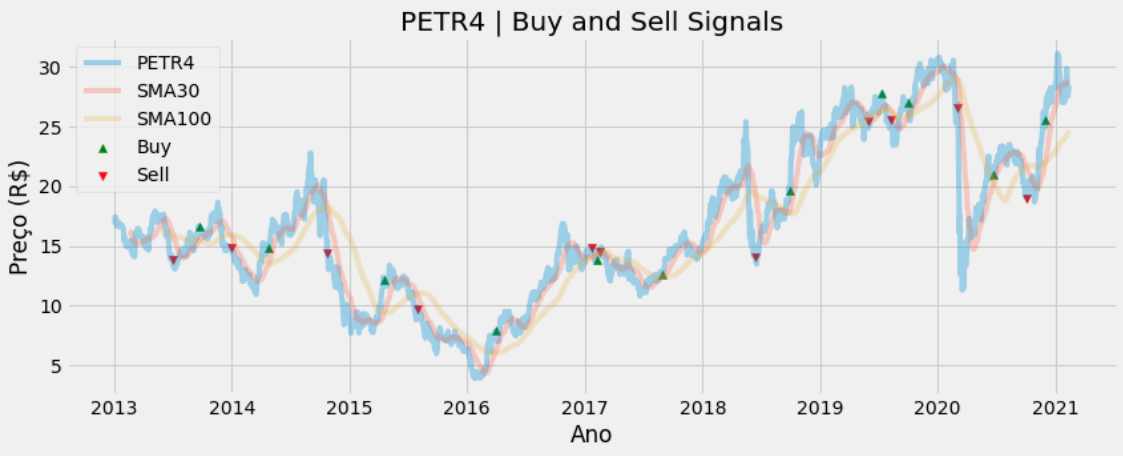
Agora quardaremos os dados de compra e venda em uma variável

[7]:
```
buy_sell = buy_sell(data)
data['buy_signal'] = buy_sell[0]
data['sell_signal'] = buy_sell[1]
```

Vizualizando a estratégia

[8]:
```
plt.figure(figsize = (12.6, 4.6))
plt.plot(data['petr'], label = 'PETR4', alpha = 0.35)
plt.plot(data['SMA30'], label = 'SMA30', alpha = 0.25)
plt.plot(data['SMA100'], label = 'SMA100', alpha = 0.25)
plt.scatter(data.index, data['buy_signal'], label = 'Buy', marker='^', color =␣
↪'green')
plt.scatter(data.index, data['sell_signal'], label = 'Sell', marker='v', color␣
↪= 'red')
plt.title('PETR4 | Buy and Sell Signals')
plt.xlabel('Ano')
plt.ylabel('Preço (R$)')
```

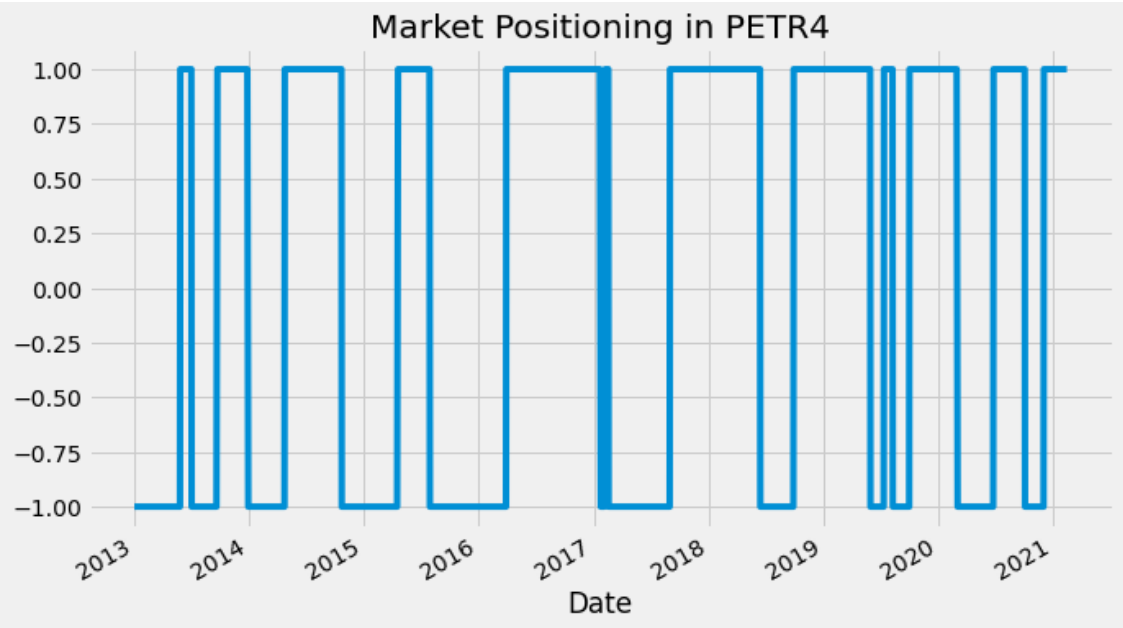
```
plt.legend(loc = 'best')
plt.show()
```



0.0.3 Iniciando o Backtest e algumas estratégias de risco

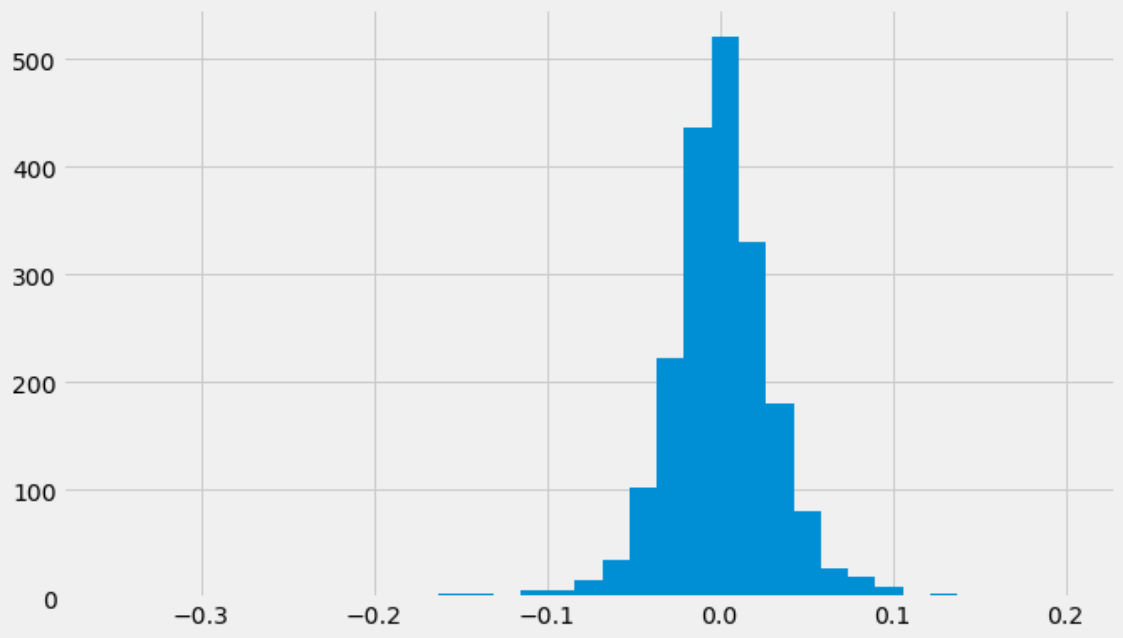
[9]:
```
data['position'] = np.where(data['SMA30'] > data['SMA100'], 1, -1)
```

[11]:
```
# Momentos comprados (1) e vendidos (-1)

data['position'].plot(ylim = [-1.1, 1.1],
                      title = 'Market Positioning in PETR4',
                      figsize = (10, 6));
```



[12]:
```
# Histograma dos retornos da ação

data['returns'] = np.log(data['petr']/ data['petr'].shift(1))
data['returns'].hist(bins=35, figsize=(10,6))
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925a8eaa30>
```



[13]:
```
# Deriva os retornos de log da estratégia dados os posicionamentos e retornos␣
↪de mercado

data['strategy'] = data['position'].shift(1)*data['returns']
```

[14]:
```
# Soma os valores de retorno de log único para o estoque e a estratégia

data[['returns', 'strategy']].sum()
```

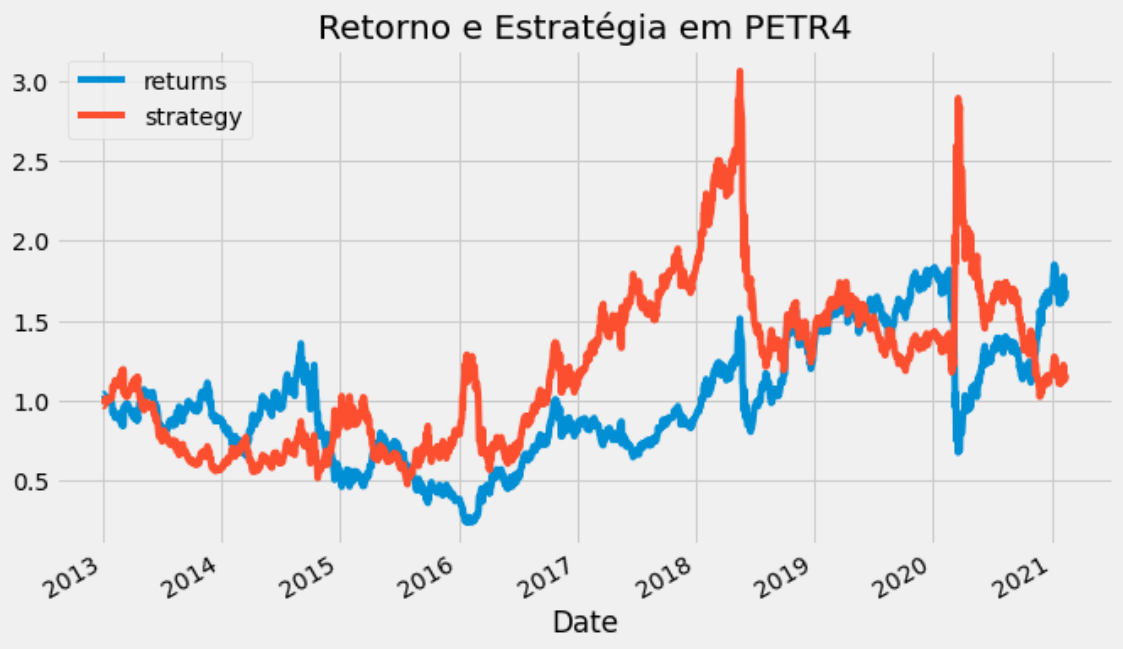
[14]:
```
returns     0.525294
strategy    0.150605
dtype: float64
```

[15]:
```
# Aplica a função exponencial à soma dos retornos do log para calcular o␣
↪desempenho bruto.

data[['returns', 'strategy']].sum().apply(np.exp)
```

[15]:
```
returns     1.690956
strategy    1.162537
dtype: float64
```

[17]:
```
data[['returns','strategy']].cumsum().apply(np.exp).plot(figsize=(10,6))
plt.title('Retorno e Estratégia em PETR4')
```

[17]:
```
Text(0.5, 1.0, 'Retorno e Estratégia em PETR4')
```



0.0.4 Estatísticas de risco-retorno

[18]:
```
# Retorno médio anualizado em espaço logarizado

data[['returns','strategy']].mean()*252
```

[18]:
```
returns     0.065792
strategy    0.018863
dtype: float64
```

[19]:
```
# Retorno médio anualizado em espaço regular

np.exp(data[['returns','strategy']].mean()*252) - 1
```

[19]:
```
returns     0.068005
strategy    0.019042
dtype: float64
```

[21]:
```
 # Desvio padrão anualizado logarizado

data[['returns','strategy']].std()*252**0.5
```

[21]:
```
returns     0.521758
strategy    0.521773
dtype: float64
```

[22]:
```
# Desvio padrão anualizado regulazizado

(data[['returns','strategy']].apply(np.exp) - 1).std()*252**0.5
```

[22]:
```
returns     0.516678
strategy    0.528698
dtype: float64
```

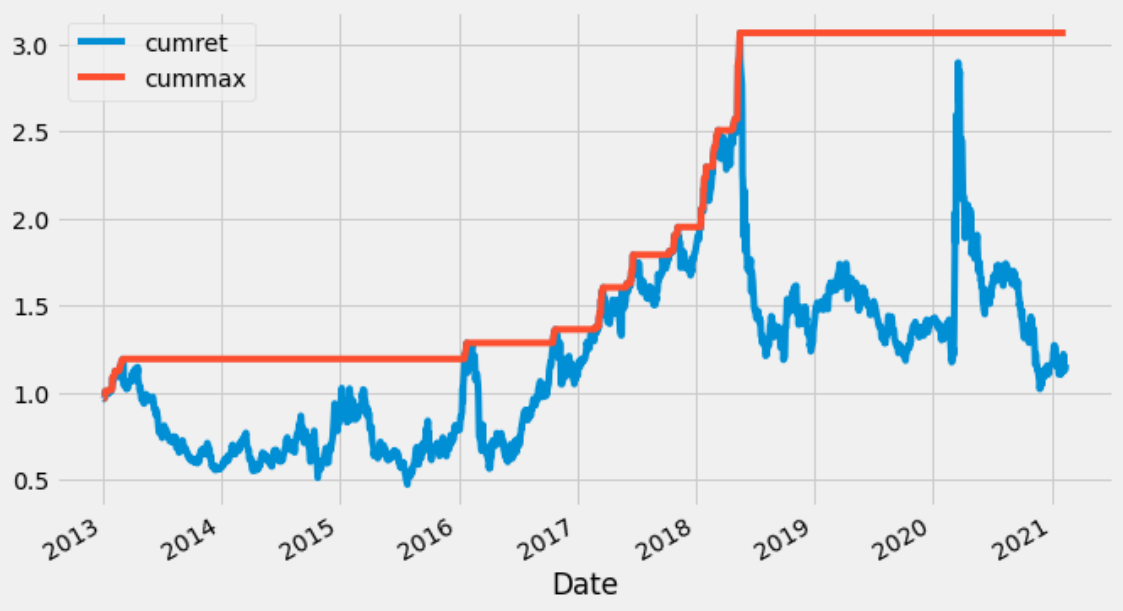
0.0.5 Outras estatísticas de risco frequentemente de interesse no contexto de desempenhos de estratégia de negociação são o 'maximum drawdown' e o período de 'longest drawdown'.

[23]:
```
# Define uma nova coluna com a performance bruta de todo o período
data['cumret'] = data['strategy'].cumsum().apply(np.exp)
```

[24]:
```
# Defina ainda outra coluna com o valor máximo de execução do desempenho bruto
data['cummax'] = data['cumret'].cummax()
```

[25]:
```
# plot
data[['cumret','cummax']].dropna().plot(figsize=(10,6))
```

[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1925a4ba190>
```



Determinando o drawdown máximo Drawdown máximo é simplesmente calculado como o máximo da diferença entre as duas colunas relevantes. O drawdown máximo no exemplo é de cerca de 204 pontos percentuais:

[26]:
```
drawdown = data['cummax'] - data['cumret']
drawdown.max()
```

[26]:
```
2.043617250313512
```

Determinando o longest drawdown period Exige as datas em que o desempenho bruto é igual a seu máximo cumulativo (ou seja, onde um novo máximo é definido). Essas informações são armazenadas em um objeto temporário. Em seguida, as diferenças em dias entre todas essas datas são calculadas e o período mais longo é selecionado. Esses períodos podem ser de apenas um dia ou mais de 100 dias.

[27]:
```
temp = drawdown[drawdown == 0] # Onde a diferença é igual a zero?
```

[29]:
```
# Calcula os valores de timedelta entre todos os valores de índice.
periods = (temp.index[1:].to_pydatetime() - temp.index[:-1].to_pydatetime())
```

[29]:
```
array([datetime.timedelta(days=4), datetime.timedelta(days=1),
       datetime.timedelta(days=13), datetime.timedelta(days=7),
       datetime.timedelta(days=1), datetime.timedelta(days=1),
       datetime.timedelta(days=1), datetime.timedelta(days=4),
       datetime.timedelta(days=2), datetime.timedelta(days=1),
       datetime.timedelta(days=14), datetime.timedelta(days=1),
       datetime.timedelta(days=3), datetime.timedelta(days=1),
```

```
 datetime.timedelta(days=2), datetime.timedelta(days=4),
 datetime.timedelta(days=1052), datetime.timedelta(days=2),
 datetime.timedelta(days=4), datetime.timedelta(days=266),
 datetime.timedelta(days=1), datetime.timedelta(days=1),
 datetime.timedelta(days=1), datetime.timedelta(days=3),
 datetime.timedelta(days=123), datetime.timedelta(days=6),
 datetime.timedelta(days=4), datetime.timedelta(days=2),
 datetime.timedelta(days=1), datetime.timedelta(days=1),
 datetime.timedelta(days=4), datetime.timedelta(days=3),
 datetime.timedelta(days=4), datetime.timedelta(days=71),
 datetime.timedelta(days=1), datetime.timedelta(days=13),
 datetime.timedelta(days=1), datetime.timedelta(days=1),
 datetime.timedelta(days=4), datetime.timedelta(days=1),
 datetime.timedelta(days=111), datetime.timedelta(days=10),
 datetime.timedelta(days=4), datetime.timedelta(days=1),
 datetime.timedelta(days=1), datetime.timedelta(days=1),
 datetime.timedelta(days=10), datetime.timedelta(days=71),
 datetime.timedelta(days=1), datetime.timedelta(days=5),
 datetime.timedelta(days=2), datetime.timedelta(days=1),
 datetime.timedelta(days=1), datetime.timedelta(days=6),
 datetime.timedelta(days=21), datetime.timedelta(days=1),
 datetime.timedelta(days=3), datetime.timedelta(days=7),
 datetime.timedelta(days=4), datetime.timedelta(days=45),
 datetime.timedelta(days=3), datetime.timedelta(days=1),
 datetime.timedelta(days=3), datetime.timedelta(days=9),
 datetime.timedelta(days=1), datetime.timedelta(days=4),
 datetime.timedelta(days=1), datetime.timedelta(days=1)],
dtype=object)
```

[31]:
```
periods.max()  #O maior drawdown foi de 1052 dias
```

[31]:
```
datetime.timedelta(days=1052)
```

0.0.6 Referências

Python for Algorthmic trading - Hilpisch, Yves

https://www.youtube.com/watch?v=SEQbb8w7VTw