GABRIEL FANINI SILVA

ROBÔ TRADE DE CRIPTOMOEDAS NA BINANCE

Departamento de Matemática, UFPR

Departamento de Matemática, UFPR

Curitiba, 25 de novembro de 2019.

Sumário

1 Introdução

O objetivo deste programa criado em python é automatizar indicadores técnicos do mercado de ações afim de nos proporcionar indicações de compra ou venda de pares de criptomoedas na corretora Binance. Extrai-se uma base de dados da Binance, composta pelas informações contidas nos candlesticks, como preço de fechamento(Close), preço de abertura(Open), preço máximo(High), preço mínimo(Low) e volume(Volume) e a partir destes dados extraídos, calculamos os indicadores técnicos.

Foram utilizados indicadores de variados tipos: oscilação, força, bandas, momento, volatilidade, volume, peso. Porém, todos eles tem uma característica em comum: indicam posições de sobre-compra e sobre-venda. Todos eles medem as forças compradoras e vendedoras do mercado e tentam encontrar um extremo em cada uma delas. Com tal característica em mãos, a automatização dos indicadores se torna fácil.

O programa então se propõe a "tirar uma média"de todos os indicadores, afim de observarmos qual a melhor decisão a ser tomada em relação ao mercado de criptos. Com os resultados dos indicadores em mãos, o programa então compra ou vende automaticamente a criptomoeda analisada na corretora Binance. Ou seja, o programa automatiza a análise técnica e a compra e venda de criptomoedas em uma determinada corretora.

2 Binance

Criptomoedas são meios de trocas, centralizados ou não, utilizados no meio digital que utilizam a blockchain e criptografia para assegurar as transações. Uma criptomoeda centralizada é uma moeda que está submetida à algum orgão central. O real, por exemplo, quem o regula é o banco central do Brasil, ou seja, o governo pode criar leis e regulações sobre os usuários da moeda.

Porém, em 2009, surgiu uma moeda que era diferente de tudo que a humanidade já havia visto, o Bitcoin. Criação do(pseudônimo) Satoshi Nakamoto, o Bitcoin é uma criptomoeda descentralizada. Na prática isto significa que quando ocorre uma transação entre uma pessoa e outra, não existe uma regulação direta neste ato. Você pode enviar seus bitcoins para qualquer pessoa em qualquer lugar do mundo sem que nenhum governo, banco, empresa ou pessoa interrompa tal transação. Hoje temos inúmeras corretoras de criptomoedas pelo mundo onde voce pode transicionar criptomoedas entre elas.

Binance é uma corretora virtual de criptomoedas sediada em Hong Kong. Nela(*https://www.binance.com/e*
podemos encontrar inúmeros pares de criptomoedas para compra e venda. Por exemplo, "BT-

CUSDT"representa o par Bitcoin por Tether(moeda digital que tem praticamente o mesmo valor que o dólar americano). O interessante é que existe uma biblioteca em linguagem python da Binance(*binance.client*).

Na imagem abaixo temos duas as funções que foram utilizadas para extrair os dados da binance

```
def minutos_da_nova_data(data_inicio, simbolo, intervalo, data, corretora):

    '''
     Esta função tem como objetivo converter um determinado intervalo selecionado para minutos. Por exemplo:
    1h = 60m, 1d = 1440m

    https://medium.com/better-programming/easiest-way-to-use-the-bitmex-api-with-python-fbf66dc38633
    '''

    if corretora == 'binance':
        old = datetime.strptime(data_inicio, '%d %b %Y') #Ponto inicial(primeiro dia, por exemplo) da base de dados extraíd
        new = pd.to_datetime(binance_client.get_klines(symbol=simbolo, interval=intervalo)[-1][0], unit='ms') #Ponto final(

    return old, new

def data_binance(data_inicio, simbolo, intervalo):

    '''
     Esta função extrai todos os dados que queremos da corretora Binance, especialmente a data que contém
    os preços de abertura(Open), fechamento(Close), máximos(High), mínimos(Low) e volume(Volume). Além da data
    (Date) em que estamos trabalhando

    https://medium.com/better-programming/easiest-way-to-use-the-bitmex-api-with-python-fbf66dc38633
    '''

    data_df = pd.DataFrame() #Extraíndo da base da dados em formato DataFrame

    ponto_antigo, ponto_novo = minutos_da_nova_data(data_inicio, simbolo, intervalo, data_df, corretora = 'binance')

    print('Fazendo Download da base no intervalo de %s para o par de cripto %s.' % (intervalo, simbolo)) #Mensagem enquanto

    dados = binance_client.get_historical_klines(simbolo, intervalo, ponto_antigo.strftime('%d %b %Y %H:%M:%S')
                                                 , ponto_novo.strftime('%d %b %Y %H:%M:%S')) #Esta função, propria da bi

    data = pd.DataFrame(dados, columns = ['timestamp', 'open', 'high', 'low', 'close', 'volume', 'close_time', 'quote_av'
                                          , 'trades', 'tb_base_av', 'tb_quote_av', 'ignore' ])
    data['timestamp'] = pd.to_datetime(data['timestamp'], unit='ms') #Criando coluna timestamp, formada pelas datas da data

    data_df = data #Renomeando a data
    data_df.set_index('timestamp', inplace=True) #Substitui a coluna de índices pela coluna timestamp

    return data_df
```

Figura 1: Comentários completos encontram-se no código

Usando as duas funções acima, basta então definirmos o início e fim da data, intervalo de tempo entre candles, o par de criptomoeda e inserir a chave API

```
#API's extraídos da Binance:

binance_api_key = '' #Coloque seu API KEY dentro das aspas ''
binance_api_secret = '' #Coloque seu APy SECRET KEY dentro das aspas ''

#Intervalos de tempo:

binsizes = {'1m': 1, '3m': 3, '5m': 5, '15m': 15, '30m': 30, '1h': 60
            , '2h': 120, '4h': 240, '6h': 360, '8h': 480, '12h': 720 , '1d': 1440
            , '1w': 10080} #m representa minuto(1 minuto), h hora(60 minutos), d dia(1440 minutos) e w sem

batch_size = 250
binance_client = Client(api_key=binance_api_key, api_secret=binance_api_secret)

#Escolhendo os parametros para criar a base:

data_inicio = '1 Oct 2019' #Escolha a Data de início para a criação da base de dados do par de moedas esco
intervalo = '4h' #Intervalo de tempo, opções encontradas no binsizes acima

Comprar = [] #Lista de moedas recomendadas para compra na binance(por enquanto vazia)

#Abaixo segue uma lista de simbolos de pares de moedas listadas na binance. O programa irá
#analisar cada uma delas. Voce pode escolher quantas quiser

binance_simbolos = ['RVNBTC', 'ETHBTC', 'XLMBTC', 'NEOBTC', 'LINKBTC', 'VETBTC', 'XRPBTC'
                    , 'TRXBTC', 'EOSBTC', 'XMRBTC', 'BCHABCBTC', 'LTCBTC', 'CHZBTC', 'FETBTC'
                    , 'ADABTC', 'ALGOBTC', 'ATOMBTC', 'BATBTC', 'BNBBTC', 'ONTBTC', 'XTZBTC'
                    , 'QTUMBTC', 'ARPABTC', 'KAVABTC', 'IOTABTC', 'TNTBTC', 'BQXBTC', 'VIBBTC'

                    , 'BTCUSDT', 'ETHUSDT', 'BNBUSDT', 'LINKUSDT', 'XRPUSDT', 'BCHABCUSDT'
                    , 'LTCUSDT', 'EOSUSDT', 'VETUSDT', 'TRXUSDT', 'NEOUSDT', 'ONTUSDT', 'QTUMUSDT'
                    , 'XLMUSDT', 'ATOMUSDT', 'BATUSDT', 'ADAUSDT', 'FETUSDT', 'BUSDUSDT', 'IOSTUSDT'
                    , 'ETCUSDT', 'XTZUSDT', 'ALGOUSDT', 'KAVAUSDT', 'CHZUSDT', 'IOTAUSDT', 'MATICUSDT']

for simbolo in binance_simbolos:
    data = data_binance(data_inicio, simbolo, intervalo) #Extraindo a base de dados
    base = data
    base = base.reset_index() #Resetando os índices, para que começem no 0
```

Figura 2: Comentários completos encontram-se no código

3 Funções

Todos os indicadores técnicos utilizados no programa são funções que dependem dos dados de preço extraídos da base de dados da binance. Logo, todos os indicadores podem ser escritos como funções em python. Abaixo segue um exemplo de um indicador escrito como uma função em python, seguindo padrão de logo abaixo ao definirmos a função ter alguns sites que explicam muito bem o funcionamento do indicador. O código foi feito da maneira mais explicativa possível, sendo que em todas as linhas escritas do código tem algum comentário ao lado explicando o funcionamento da determinada linha. As funções retornam apenas a fórmula final do indicador

```
def RSI(periodoRSI, Close):

    '''
     Relative Strength Index(RSI)

    https://www.investopedia.com/terms/r/rsi.asp
    https://en.wikipedia.org/wiki/Relative_strength_index
    https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/RSI
    '''

    delta = Close.diff() #Diferença de preço entre o Close atual(Close[i]) e o Close anterior(Close[i - 1]) da coluna Close

    up_days = delta.copy() #Definimos aqui os intervalos de ganhos, criando uma cópia superficial de delta
    up_days[delta <= 0] = 0.0 #Na coluna up_days, se o valor for negativo ou 0, então será convertido em 0

    down_days = abs(delta.copy()) #Definimos aqui os intervalos de perdas, usando o valor absoluto (abs()) e a cópia superfícial de delta
    down_days[delta > 0] = 0.0 #Caso o valor seja positivo, então será retornado 0

    RS_up = up_days.rolling(window = periodoRSI).mean() #Média móvel de ganhos em 14 dias
    RS_down = down_days.rolling(window = periodoRSI).mean() #Média móvel de perdas em 14 dias
    RSI = 100 - 100 / (1 + RS_up / RS_down) #Fórmula do RSI

    return RSI
```

Figura 3: Relative Strength Index(RSI)

Abaixo outra função, que além de retornar a fórmula final do indicador, retorna também a linha de sinal. Isto é comum nos indicadores técnicos, já que é possível uma melhor interpretação do indicador ao adicionar outra uma reta de sinal.

```python
def SMI(periodoSMI, periodoLinhaSinalSMI, High, Low, Close):

    '''
    (21)
     SMI

    https://www.tradingview.com/script/HLbqdCku-Stochastic-Momentum-Index-SMI/
    https://www.investopedia.com/ask/answers/021315/what-difference-between-stochastic-oscillator-stochastic-momentum-index.asp
    https://www.marketbeat.com/financial-terms/what-is-stochastic-momentum-index/
    '''

    hh = High.rolling(window = periodoSMI).max() #Maior valor de High no intervalo períodoSMI
    ll = Low.rolling(window = periodoSMI).min() #Menor valor de Low no intervalo períodoSMI

    C = (hh + ll) / 2

    h = Close - C

    HS1 = h.ewm(span = 3, adjust = False).mean() #EMA de período 3(EMA3) de h
    HS2 = HS1.ewm(span = 3, adjust = False).mean() #EMA3 de HS1

    DHL1 = (hh - ll).ewm(span = 3).mean() #EMA3 de (hh - ll)
    DHL2 = DHL1.ewm(span = 3).mean() #EMA3 de DHL1
    DHL2 = DHL2 / 2

    SMI = 100 * HS2 / DHL2 #Fórmula final do Stochastic Momentum Index

    linha_sinal_SMI = SMI.ewm(span = periodoLinhaSinalSMI).mean() #Fórmula final da linha de sinal do Stochastic Momentum Index

    return SMI, linha_sinal_SMI
```

Figura 4: Stochastic Momentum Index(SMI)

Além disto, a maioria dos indicadores dependem de SMA's(médias móveis) e EMA's(médias móveis exponenciais). E com a linguagem python é facil definir tais médias, usando simples comandos como o *.rolling()* e *.ewm()*. Abaixo a média movel exponencial do preço de fechamento(Close) de 9 períodos:

```python
Close.ewm(span = 9).mean()
```

Figura 5: EMA de período 9 do Close

Abaixo a aplicação da média móvel exponencial sobre outra média móvel exponencial

```python
m = Close.diff() #Coluna Close - Close.shift(1)(Close de 1 índice anterior)
EMA25_m = m.rolling(window = periodoTSImaior).mean() #SMA de periodoTSImaior de m
EMA13_EMA25_m = EMA25_m.rolling(window = periodoTSImenor).mean() #SMA de periodoTSImenor do EMA25_m
EMA25_abs_m = (abs(m)).rolling(window = periodoTSImaior).mean() #SMA de periodoTSImaior do módulo de m
EMA13_EMA25_abs_m = EMA25_abs_m.rolling(window = periodoTSImenor).mean() #SMA de periodoTSImenor do EMA25_absm
```

Figura 6: Parte do indicador True strength index(TSI)

4 Indicações

Todos os indicadores técnicos utilizados no programa dão recomendação de compra e venda(exceto um que dá apenas recomendações de compra). A característica em comum entre a maioria deles é a indicação de sobre-compra e sobre-venda. Por exemplo, o indicador *RSI* nos mostra que acima de 80% e abaixo de 20% temos sinal de que o mercado está sobre-comprado e sobre-vendido respectivamente.

```
'''
()
 WT(WaveTrend Oscillator)
'''

WT = wt(10, 21, 0.015, Preco)

for i in range(1,len(base)):
    if WT[i - 1] > 60 and WT[i] < 60:
        Comprar_Vender[i] += -1
    elif WT[i - 1] < -60 and WT[i] > -60:
        Comprar_Vender[i] += 1

'''
MFI
'''

MFI = mfi(14, base, Preco)

for i in range(1,len(base)):
    if MFI[i - 1] > 80 and MFI[i] < 80:
        Comprar_Vender[i] += -1
    elif MFI[i - 1] < 20 and MFI[i] > 20:
        Comprar_Vender[i] += 1

'''
UO
'''

Ult_Osc = uo(7, 14, 28, 4, 2, 1, Low, High, Preco, Close)

for i in range(1,len(base)):
    if Ult_Osc[i - 1] > 70 and Ult_Osc[i] < 70:
        Comprar_Vender[i] += -1
    elif Ult_Osc[i - 1] < 30 and Ult_Osc[i] > 30:
        Comprar_Vender[i] += 1
```

Figura 7: Estes tipos de indicadores são fáceis de automatizar

Exemplo de um indicador que não segue esta característica, porém igualmente fácil para automatizar

```
'''
 CM_Williams_Vix_Fix
'''

WVF, WVF_linha_superior, WVF_alcancemaior = cm_williams_vix_fix(22, 20, 2, 50, 0.85, 1.01, Close)

#Caso WVF >= (WVF_linha_superior ou WVF_alcancemaior) temos sobrevenda(compra)

for i in range(0,len(base)):
    if WVF[i] >= WVF_linha_superior[i] or WVF[i] >= WVF_alcancemaior[i]:
        Comprar_Vender[i] += 1
```

Figura 8: Indicador criado por um usuário do site Tradingview

Foi criado uma coluna formato *Series* chamada *Comprar_Vender* com o mesmo tamanho da base(*len(base)*). Desta forma, quando a indicação é de compra na posição *i*, então soma-se 1(você pode escolher outro valor) e quando a indicação é de venda subtrai-se 1. Fazendo-se isto com todos os indicadores automatizados, temos então uma média de todos os indicadores. Tal coluna foi transformada em percentagem dividindo as linhas pelo número total de indicadores.

```
#Transformando Comprar_Vender em porcentagem:

for i in range(0,len(base)):
    if Comprar_Vender[i] > 0:
        Comprar_Vender[i] = Comprar_Vender[i] / 24
    elif Comprar_Vender[i] < 0:
        Comprar_Vender[i] = Comprar_Vender[i] / 23
```

Comprar_Vender - Series

Index	0
77	-0.130435
78	-0.173913
79	-0.173913
80	-0.26087
81	-0.434783
82	-0.217391
83	-0.173913
84	-0.0434783
85	-0.0434783
86	0
87	-0.0434783
88	-0.0434783
89	0
90	0.0416667

Formato | Redimensionar | Cor de fundo | Min/max de coluna | Salvar e Fechar | Fechar

Figura 9: Um indicador dá apenas indicação de compra

5 Candlestick

Como já temos nossas indicações de compra e venda, basta então plotar um gráfico de candlestick junto aos dados da coluna *Comprar_Vender*. Escolhemos então o gráfico histograma, formado por barras verdes e vermelhas, junto à uma faixa preta, na qual indicava o valor de 30%, caso alguma barra verde ou vermelha ultrapassasse a faixa.

Figura 10: Gráfico candlestick junto ao histograma de compra e venda

Na figura acima, na legenda à esquerda temos o par da criptomoeda que está sendo plotada no gráfico e a escala também à esquerda representa o preço do par. Na legenda à direita temos o intervalo de tempo entre um candle e outro.

Figura 11: Outro par de moeda com intervalo de tempo diferente

```
candlesticks = zip(Date1,Open[inicio_grafico:final_grafico].reset_index(drop = True)
,Close[inicio_grafico:final_grafico].reset_index(drop = True)
,High[inicio_grafico:final_grafico].reset_index(drop = True)
,Low[inicio_grafico:final_grafico].reset_index(drop = True)
,Comprar_Vender) #Definindo o gráfico de Candlestick(https://pt.wikipedia.org/wiki/Candlestic

Comprar_Vender[0] = 2.5 #Para diminuir tamanho das barras de indicações
#Abaixo uma reta que servirá como suporte de compra e venda. Está configurada
#para aparecer no ponto 0.1, que representa 10% de Comprar_Vender. Será
#representado pela cor preta(linha 1270)

fig = plt.figure(figsize = (9,20)) #Tamanho da figura(9 de altura e 20 de largura)
ax = fig.add_subplot(1,1,1)

ax.set_ylabel(simbolo, size=20) #20 é o tamanho do texto(simbolo) à esquerda do gráfico
candlestick(ax, candlesticks,width = 1,colorup = 'g', colordown = 'r')

#Indicações de compra e venda mais compacta:

pad = 0.25
yl = ax.get_ylim()
ax.set_ylim(yl[0]-(yl[1]-yl[0])*pad,yl[1])

ax2 = ax.twinx() #Criando eixo das indicações de compra e venda(Comprar_Vender)

ax2.set_position(matplotlib.transforms.Bbox([[0.125,0.1],[0.9,0.32]])) #([['',''],['',''']]) P

pos = Comprar_Vender > 0 #Barra positiva quando Comprar_Vender indica compra
neg = Comprar_Vender < 0 #Barra negativa quando Comprar_Vender indica venda

ax2.bar(Date1, reta_sinal, color = 'black', width = 1, align = 'center') #Barras pretas que r
ax2.bar(Date1[pos], Comprar_Vender[pos], color = 'green',width = 1, align = 'center') #Barras
ax2.bar(Date1[neg], Comprar_Vender[neg] * (-1), color = 'red', width = 1,align = 'center') #B

ax2.set_xlim(Date1[0],Date1[final_grafico - inicio_grafico - 1]) #Período de início e fim do

yticks = ax2.get_yticks()
ax2.set_yticks(yticks[::3])

ax2.yaxis.set_label_position("right") #Texto à direita do gráfico
ax2.set_ylabel(intervalo, size=20) #20 é o tamanho do texto('Comprar ou Vender') à direita do
```

Figura 12: Parte do código da criação do gráfico de candlestick

6 Conclusões

Devemos ter muita cautela ao utilizar este programa. Não adianta deixar o indicador rodando por 1 semana e depois voltar para o computador para ver os lucros. Existem momentos em que o programa funciona bem e momentos que não funcionam. No período atual que estou escrevendo isto, 24 de setembro de 2019, o Bitcoin e as altcoins (em relação ao BTC) estão em momento de baixa. Deixar o capital em stablecoins(moedas que simulam alguma outra moeda fiduciária, como o dólar, por exemplo) neste momento está sendo a opção que melhor diminuem as perdas. Provavelmente após este período de baixa, que deve ocorrer após fechamento de contratos futuros da CME, seja uma boa opção para ligar o programa afim de lucrarmos no mercado.

Por sorte, foi criando um programa paralelo que verifica os ganhos e percas caso utilizássemos o robô trade por um determinado período de tempo do passado. O programa supõe que você utiliza uma quantidade "x"quando a indicação é de compra e venda esta mesma quantidade apenas quando o indicador indica venda. Calculasse então o somatório dos lucros entre uma indicação e outra, sempre supondo que você usa a mesma quantidade "x"quando compra. Além disto o programa de teste analisa vários intervalos.

```
lista_data = ['24 Oct 2018', '24 Dec 2018'
              , '24 Feb 2019', '24 Apr 2019'
              , '24 Jun 2019', '24 Aug 2019'
              , '24 Oct 2019']

#Lista de intervalos que serão utilizados para o teste do programa:

lista_intervalo = ['4h', '8h', '12h', '1d']

#Lista de símbolos da Binance que serão utilizados para o teste do programa:

binance_simbolos_BTC = ['RVNBTC', 'XLMBTC', 'NEOBTC', 'LINKBTC', 'XRPBTC', 'MATICBTC'
                    , 'TRXBTC', 'EOSBTC', 'XMRBTC', 'LTCBTC', 'CHZBTC', 'FETBTC'
                    , 'ADABTC', 'ALGOBTC', 'ATOMBTC', 'BATBTC', 'ONTBTC', 'XTZBTC'
                    , 'QTUMBTC', 'KAVABTC', 'IOTABTC', 'TNTBTC', 'BQXBTC', 'VIBBTC'
                    , 'DLTBTC', 'KMDBTC', 'EVXBTC', 'ZECBTC', 'OMGBTC', 'ZILBTC'
                    , 'DATABTC', 'BCHABCBTC', 'MCOBTC', 'ONEBTC', 'STXBTC', 'KEYBTC']

binance_simbolos_USDT = ['BTCUSDT', 'ADAUSDT', 'LTCUSDT', 'LINKUSDT', 'BCHABCUSDT', 'XRPUSDT'
                         , 'TRXUSDT', 'BATUSDT', 'EOSUSDT', 'BNBUSDT', 'VETUSDT', 'MATICUSDT'
                         , 'NEOUSDT', 'ETCUSDT', 'ATOMUSDT', 'FETUSDT', 'ONTUSDT', 'XLMUSDT']

grafico = []

tempolocal = time.localtime(time.time()) #Mostra a data de hoje(dia, mês, ano, horario, ...)

lucro_total = []

count = 0

for intervalo in lista_intervalo:
    for simbolo in binance_simbolos_BTC:
        for data_inicio in lista_data:
            count += 1
            data = data_binance(data_inicio, simbolo, intervalo) #Extraindo a base de dados
            base = data[0:60] #Fazendo análise bimestral
            base = base.reset_index() #Resetando os índices, para que começem no 0
```

Figura 13: Programa teste analisa vários parâmetros

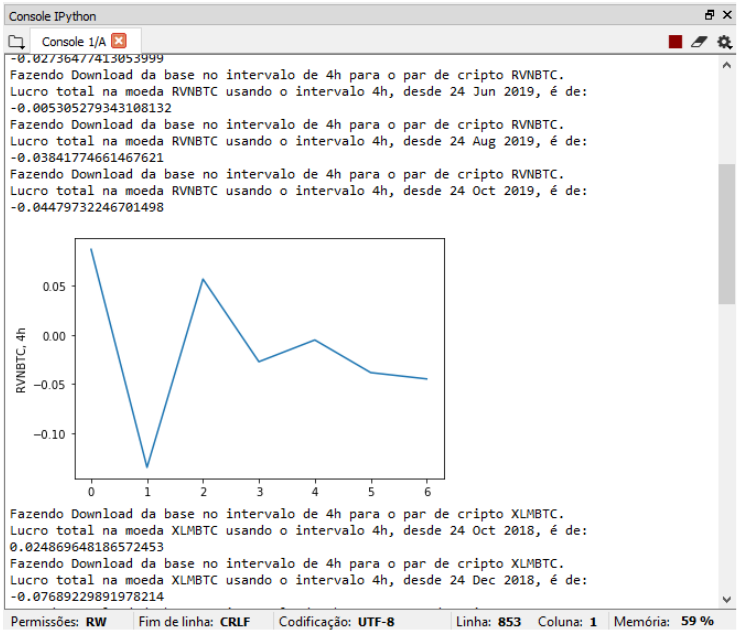

Figura 14: Programa de teste em execução

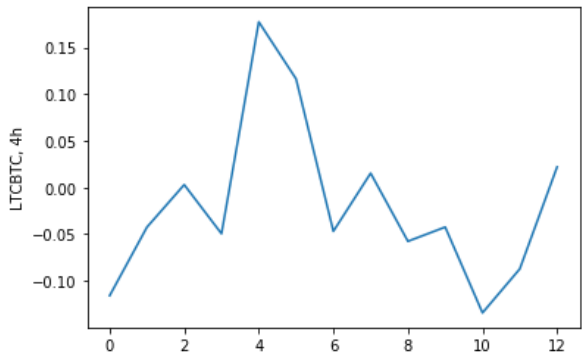

Figura 15: Cada valor do eixo x representa 1 mês

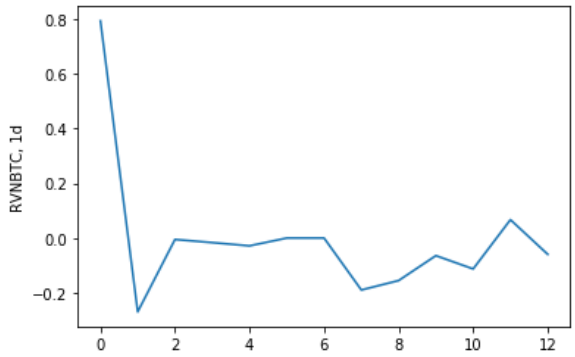

Figura 16: Teste utilizando intervalo de 1d, mostrando lucro elevado no início, porém entrando em prejuízo posteriormente

Caso utilizarmos gráficos com intervalos grandes como *4h*, *12h* ou *1d*, o programa principal irá demorar os respectivos tempos de intervalo para atualizar as indicações novamente. Mas como sabemos, este mercado é muito volátil, e afim de não perdermos boas oportunidades que podem aparecer entre os intervalos, foi criado outro programa que em questão de segundos ele atualiza os preços dos ativos que já foram comprados pelo programa principal. Caso o valor ultrapasse 15%(cada um é livre para mudar este valor do lucro) do valor que havia sido comprado, o programa automaticamente vende a moeda.

```
binance_simbolos = ['LINKBTC', 'CHZBTC', 'QTUMBTC', 'ZILBTC', 'MATICBTC'
                    , 'KAVABTC', 'TNTBTC', 'VIBBTC', 'OMGBTC', 'MCOBTC']

vendida = [] #Moedas vendidas

while True:
    for simbolo in binance_simbolos:
        data = data_binance(data_inicio, simbolo, intervalo) #Extraindo a base de dados
        base = data
        base = base.reset_index() #Resetando os índices, para que comeÇem no 0

        #Extraindo os dados da base em forma de colunas:

        Close = cloSe(base)

        if os.path.exists('%s.txt' %simbolo) == True:
            ler_arquivo = open('%s.txt' %simbolo, 'r').read().split()
            if Close[len(base) - 1] >= 1.15 * (int(ler_arquivo[0]) / 0.01) ** (-1):
                ordem_venda = binance_client.create_order( #Função que executa ordem de venda na binance
                                        symbol = simbolo, #Símbolo da moeda
                                        side = Client.SIDE_SELL, #Ordem de venda
                                        type = Client.ORDER_TYPE_MARKET, #Vender usando ordem de preço de mercado
                                        quantity = ler_arquivo[0]) #Vendendo a mesma quantidade que compramos

                print('{} VENDIDA!'.format(simbolo)) #Mostrar print de que a moeda foi vendida
                print('Hora {}, dia {}, mês {}, ano {}'.format(tempolocal[3], tempolocal[2], tempolocal[1], tempolocal[0]))
                os.remove('%s.txt' %simbolo) #Remove o arquivo que foi criado quando foi realizado uma compra

                vendida.append(simbolo) #Acumulando as moedas vendidas
```

Figura 17: Parte do código do terceiro programa

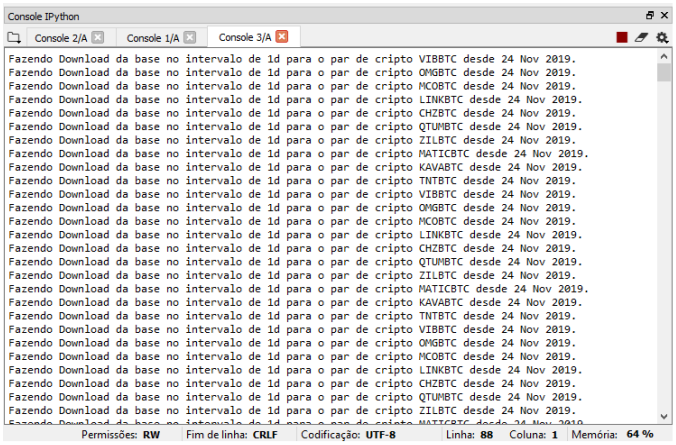

Figura 18: Programa em execução

[1] *https://www.investopedia.com/*

[2] *https://math.stackexchange.com/*

[3] *https://www.metatrader5.com/pt*

[4] *https://stackoverflow.com/*

[5] *http://daltonvieira.com/*

[6] *https://www.tradingview.com/*

[7] *https://mahifx.com/*

[8] *https://www.fmlabs.com/default.htm*

[9] *https://www.bussoladoinvestidor.com.br/*

[10] *https://www.nelogica.com.br/*

[11] *https://www.shanelynn.ie/*

[12] *https://pt.wikipedia.org/wiki/Wikip*

[13] *https://medium.com/*

[14] *https://www.alphavantage.co/*

[15] *https://commodity.com/*

[16] *https://www.analyticsvidhya.com*

[17] *https://www.tororadar.com.br/*

[18] *https://www.investimentonabolsa.com/*

[19] *https://stackexchange.com/*

[20] *https://www.marketvolume.com/*

[21] *https://machinelearningmastery.com/*

[22] *https://pandas.pydata.org/pandas-docs/stable/index.html*

[23] *https://www.daytrading.com/*

[24] *https://www.tradingtechnologies.com/*

[25] *https://ftmo.com/en/welcome/*