AULA01

March 1, 2021

[1]:
```
import pandas as pd
import numpy as np
import talib as ta
import pytz
import datetime
from datetime import datetime

import MetaTrader5 as mt5
import time
# display data on the MetaTrader 5 package
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")
```

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.34
```

1 Login em contas:

[2]:
```
# establish connection to the MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

 #display data on MetaTrader 5 version
print(mt5.version())
# connect to the trade account without specifying a password and a server
account=17221085
authorized=mt5.login(account)  # the terminal database password is applied if␣
 ↪connection data is set to be remembered
if authorized:
    print("connected to account #{}".format(account))
else:
    print("failed to connect at account #{}, error code: {}".format(account,␣
 ↪mt5.last_error()))
```

```
 # now connect to another trading account specifying the password
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # display trading account data 'as is'
    print(mt5.account_info())
    # display trading account data in the form of a list
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("failed to connect at account #{}, error code: {}".format(account,␣
 ↪mt5.last_error()))

 # shut down connection to the MetaTrader 5 terminal
mt5.shutdown()
```

```
(500, 2755, '15 Jan 2021')
failed to connect at account #17221085, error code: (-7, 'Unsupported
authorization mode, OTP or certificate password needed')
failed to connect at account #25115284, error code: (-2, 'Terminal: Invalid
params')
```

[2]: True

2 Função copy_rates_from

[3]:
```
# establish connection to MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# set time zone to UTC
timezone = pytz.timezone("America/Sao_Paulo")
# create 'datetime' object in UTC time zone to avoid the implementation of a␣
 ↪local time zone offset
utc_from = datetime(2010, 1, 10, tzinfo=timezone)
# get 10 EURUSD H4 bars starting from 01.10.2020 in UTC time zone
rates = mt5.copy_rates_from("IBOV", mt5.TIMEFRAME_D1, utc_from, 10)

# shut down connection to the MetaTrader 5 terminal
mt5.shutdown()

# create DataFrame out of the obtained data
rates_frame = pd.DataFrame(rates)
# convert time in seconds into the datetime format
```

```
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')
```

[4]:
```
rates_frame
```

[4]:

	time	open	high	low	close	tick_volume	spread	real_volume
0	2009-12-22	65940.0	67418.0	65940.0	67418.0	208233	0	3791291008
1	2009-12-23	67418.0	67810.0	66943.0	67589.0	164479	0	2774712872
2	2009-12-28	67591.0	68277.0	67591.0	67902.0	106648	0	2060447007
3	2009-12-29	67902.0	68309.0	67902.0	68296.0	105108	0	1833906944
4	2009-12-30	68277.0	68588.0	67749.0	68588.0	152632	0	3302830600
5	2010-01-04	68587.0	70081.0	68587.0	70045.0	206093	0	4090042029
6	2010-01-05	70046.0	70595.0	69928.0	70240.0	248563	0	5292203902
7	2010-01-06	70237.0	70937.0	70016.0	70729.0	246685	0	5389765539
8	2010-01-07	70723.0	70723.0	70045.0	70451.0	232050	0	4450266724
9	2010-01-08	70455.0	70766.0	70158.0	70263.0	211826	0	4378980631

3 Função copy_rates_from_pos

[5]:
```
# establish connection to MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

 # get 10 GBPUSD D1 bars from the current day
rates = mt5.copy_rates_from_pos("IBOV", mt5.TIMEFRAME_D1, 0, 100)

# shut down connection to the MetaTrader 5 terminal
mt5.shutdown()

 # create DataFrame out of the obtained data
rates_frame = pd.DataFrame(rates)
# convert time in seconds into the datetime format
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')
```

[6]:
```
rates_frame
```

[6]:

	time	open	high	low	close	tick_volume	spread	real_volume
0	2020-09-29	94665.0	95505.0	93408.0	93580.0	1784765	0	18685951927
1	2020-09-30	93586.0	95340.0	93584.0	94603.0	1806453	0	19760375609
2	2020-10-01	94604.0	95486.0	93599.0	95479.0	1804742	0	19366495804
3	2020-10-02	95475.0	95996.0	93897.0	94016.0	6591936	0	75701770626
4	2020-10-05	94020.0	96414.0	93984.0	96089.0	1664555	0	17900800784
..	…	…	…	…	…	…	…	…
95	2021-02-22	118386.0	118386.0	111650.0	112668.0	4678543	0	54328114749
96	2021-02-23	112683.0	115380.0	112667.0	115228.0	3113533	0	39507487046
97	2021-02-24	115250.0	116208.0	114668.0	115668.0	2310833	0	29205286181
98	2021-02-25	115668.0	116506.0	111764.0	112256.0	2549031	0	31619847848
99	2021-02-26	112260.0	113466.0	109827.0	110035.0	2746240	0	38458235363

[100 rows x 8 columns]

4 Função copy_rates_range

[7]:
```
# establish connection to MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

 # set time zone to UTC
timezone = pytz.timezone("America/Sao_Paulo")
# create 'datetime' objects in UTC time zone to avoid the implementation of a␣
 ↪local time zone offset
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# get bars from USDJPY M5 within the interval of 2020.01.10 00:00 - 2020.01.11␣
 ↪13:00 in UTC time zone
rates = mt5.copy_rates_range("IBOV", mt5.TIMEFRAME_M5, utc_from, utc_to)
```

```
# shut down connection to the MetaTrader 5 terminal
mt5.shutdown()


# create DataFrame out of the obtained data
rates_frame = pd.DataFrame(rates)
# convert time in seconds into the 'datetime' format
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')
```

[8]:
```
rates_frame
```

[8]:
```
                  time      open      high       low     close  tick_volume  \
0  2020-01-10 10:00:00  115948.0  116308.0  115948.0  116230.0         5096
1  2020-01-10 10:05:00  116202.0  116266.0  116112.0  116112.0         7918
2  2020-01-10 10:10:00  116051.0  116159.0  116025.0  116159.0         6660
3  2020-01-10 10:15:00  116180.0  116233.0  116032.0  116183.0         7258
4  2020-01-10 10:20:00  116189.0  116390.0  116189.0  116303.0         6967
..                 …         …         …         …         …            …
91 2020-01-10 17:35:00  115087.0  115143.0  115081.0  115086.0        24025
92 2020-01-10 17:40:00  115021.0  115072.0  114952.0  115047.0        34159
93 2020-01-10 17:45:00  115068.0  115225.0  115068.0  115182.0        36680
94 2020-01-10 17:50:00  115076.0  115258.0  115076.0  115258.0        22556
95 2020-01-10 17:55:00  115280.0  115503.0  115280.0  115503.0        13405

    spread  real_volume
0        0     74244471
1        0    122773370
2        0    128125632
3        0    125516525
4        0    107651078
..     …           …
91       0    255371000
92       0    298933880
93       0    326985470
94       0    298665110
95       0   1498749356

[96 rows x 8 columns]
```

5 Função Order_Send Compra

[21]:
```
# establish connection to the MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()
```

```
 #prepare the buy request structure
symbol = "VALE3"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

 #if the symbol is unavailable in MarketWatch, add it
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()


lot = 100
SL= 30
SP= 100
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - price*(SL/100),
    "tp": price + price*(SP/100),
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# send a trading request
result = mt5.order_send(request)
```

[23]:
```
result
```

6 Função Order_Send Venda

[24]:
```
# establish connection to the MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

 #prepare the buy request structure
symbol = "VALE3"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

 #if the symbol is unavailable in MarketWatch, add it
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) failed, exit",symbol)
        mt5.shutdown()
        quit()


lot = 100
SL= 30
SP= 100
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "price": price,
    'tp': price - price*(SP/100),
    'sl': price + price*(SL/100),
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# send a trading request
result = mt5.order_send(request)
```

[25]:
```
result
```

7 Gerando Indicadores com a biblioteca TALIB

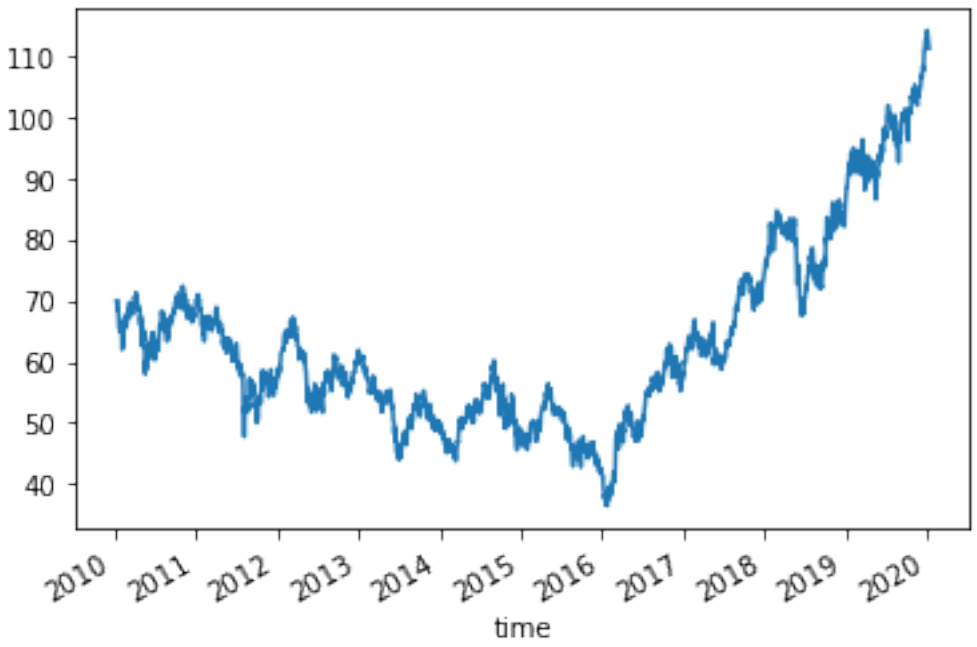
Documentação da biblioteca: https://mrjbq7.github.io/ta-lib/doc_index.html

[32]:
```
# establish connection to MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

 # set time zone to UTC
timezone = pytz.timezone("America/Sao_Paulo")
# create 'datetime' objects in UTC time zone to avoid the implementation of a␣
 ↪local time zone offset
utc_from = datetime(2010, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# get bars from USDJPY M5 within the interval of 2020.01.10 00:00 - 2020.01.11␣
 ↪13:00 in UTC time zone
rates = mt5.copy_rates_range("BOVA11", mt5.TIMEFRAME_D1, utc_from, utc_to)

# shut down connection to the MetaTrader 5 terminal
mt5.shutdown()



# create DataFrame out of the obtained data
rates_frame = pd.DataFrame(rates)
# convert time in seconds into the 'datetime' format
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')
```

[33]:
```
rates_frame.index=rates_frame['time']
```

[34]:
```
rates_frame['close'].plot()
```

[34]:
```
<AxesSubplot:xlabel='time'>
```



[36]:
```
rates_frame['upperband'], rates_frame['middleband'], rates_frame['lowerband'] =␣
↪ta.BBANDS(rates_frame['close']
                                                                              ␣
↪          , timeperiod=5, nbdevup=2, nbdevdn=2, matype=0)
```

[39]:
```
rates_frame
```

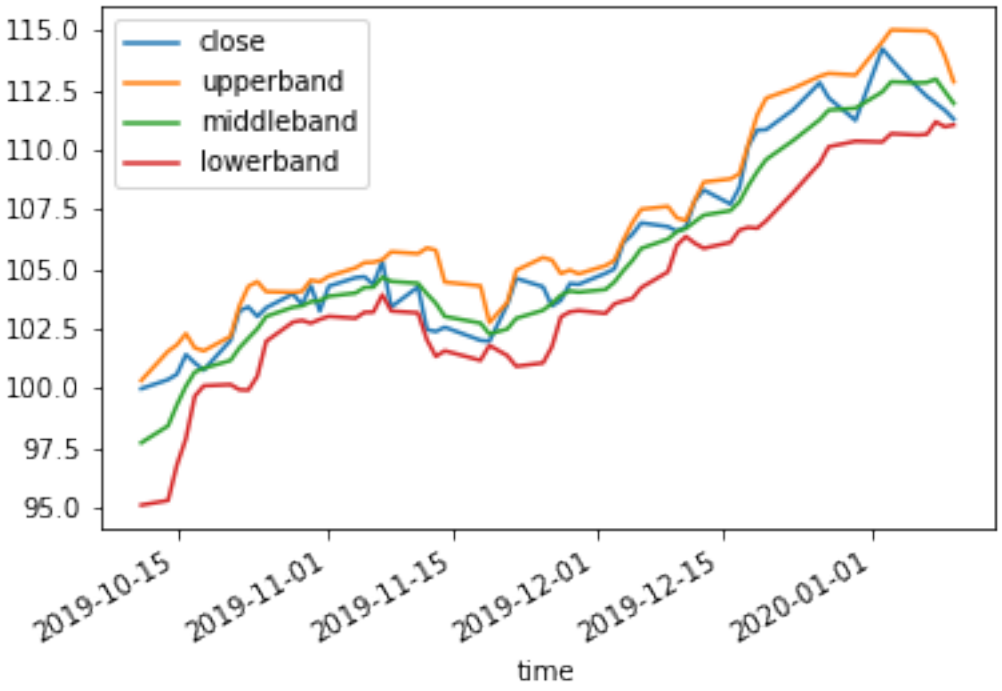
[39]:

time	time	open	high	low	close	tick_volume	spread	real_volume	upperband	middleband	lowerband
time											
2010-01-11	2010-01-11	70.25	70.47	69.63	70.00	90	1	180200	NaN	NaN	NaN
2010-01-12	2010-01-12	69.65	69.72	68.71	69.71	220	1	233800	NaN	NaN	NaN
2010-01-13	2010-01-13	69.67	70.01	68.98	70.01	67	1				
2010-01-14	2010-01-14	70.01	70.01	69.14	69.51	130	1				
2010-01-15	2010-01-15	68.81	69.20	68.18	68.51	187	1				
…	…	…	…	…	…	…	…				
2020-01-06	2020-01-06	113.00	113.45	112.02	112.59	49161	1				
2020-01-07	2020-01-07	112.90	112.90	111.59	112.24	82363	1				
2020-01-08	2020-01-08	112.65	113.10	111.40	111.95	46982	1				
2020-01-09	2020-01-09	111.95	112.55	111.18	111.66	23764	1				
2020-01-10	2020-01-10	112.40	112.51	110.70	111.28	21554	1				

```
2010-01-13        89200          NaN          NaN          NaN
2010-01-14       176000          NaN          NaN          NaN
2010-01-15       286200    70.651768       69.548    68.444232
…                   …          …            …            …
2020-01-06      6771940   114.993111      112.806   110.618889
2020-01-07      6096900   114.991488      112.820   110.648512
2020-01-08      6469320   114.757169      112.964   111.170831
2020-01-09      5500010   113.934009      112.448   110.961991
2020-01-10      6103030   112.850210      111.944   111.037790

[2474 rows x 11 columns]
```

[38]:
```
rates_frame[['close','upperband','middleband','lowerband']].tail(60).plot()
```

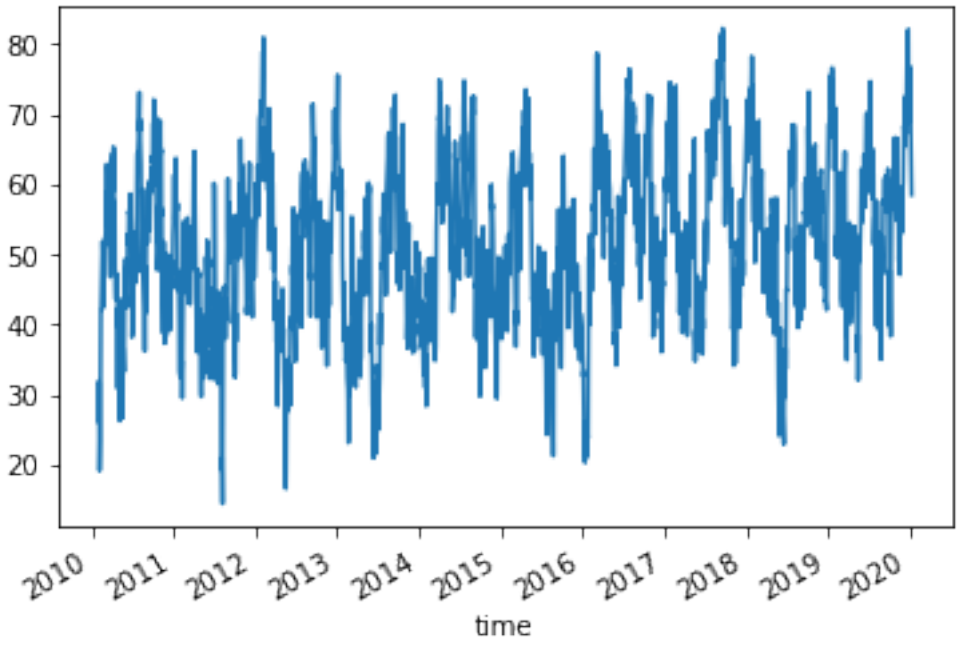
[38]: <AxesSubplot:xlabel='time'>



[40]:
```
rates_frame['RSI']= ta.RSI(rates_frame['close'], timeperiod=14)
```

[41]:
```
rates_frame['RSI'].plot()
```

[41]: <AxesSubplot:xlabel='time'>



[44]:
```
rates_frame[['close','RSI']].tail()
```

[44]:

time	close	RSI
2020-01-06	112.59	66.308855
2020-01-07	112.24	64.265530
2020-01-08	111.95	62.545729
2020-01-09	111.66	60.793692
2020-01-10	111.28	58.481958