MetaTrader5 と Python による自動売買

富山県立大学 情報基盤工学講座　　指導教員：奥原浩之
1515010　大谷和樹

1 はじめに

MetaTrader5（MT5）と Python で，様々なインジケータを利用して自動売買を行うシステムの開発を行っている．研究の背景や目的などについては，しっかりとした文にまとめることができていないため，今後まとめていく必要がある．

2 インジケータの計算

2.1 TA-Lib

インジケータの計算には Python でテクニカル指標が計算できる「TA-Lib」というライブラリを使用することにした．

```
# 単純移動平均(SMA: Simple Moving Average)
output = np.c_[output, talib.SMA(close)]
cols += ['SMA']
# 加重移動平均(WMA: Weighted Moving Average)
output = np.c_[output, talib.WMA(close)]
cols += ['WMA']
# 指数移動平均(EMA: Exponential Moving Average)
output = np.c_[output, talib.EMA(close)]
cols += ['EMA']
# ２重指数移動平均(DEMA: Double Exponential Moving Average)
output = np.c_[output, talib.DEMA(close)]
cols += ['DEMA']
# ３重指数移動平均(TEMA: Triple Exponential Moving Average)
output = np.c_[output, talib.T3(close)]
cols += ['TEMA']
# 三角移動平均(TMA: Triangular Moving Average)
output = np.c_[output, talib.TRIMA(close)]
cols += ['TMA']
# Kaufmanの適応型移動平均(KAMA: Kaufman Adaptive Moving Average)
output = np.c_[output, talib.KAMA(close)]
cols += ['KAMA']
# MESAの適応型移動平均(MAMA: MESA Adaptive Moving Average)
for arr in talib.MAMA(close):
    output = np.c_[output, arr]
cols += ['MAMA', 'FAMA']
# トレンドライン(Hilbert Transform - Instantaneous Trendline)
output = np.c_[output, talib.HT_TRENDLINE(close)]
cols += ['HT_TRENDLINE']
# ボリンジャー・バンド(Bollinger Bands)
for arr in talib.BBANDS(close):
    output = np.c_[output, arr]
cols += ['BBANDS_upperband', 'BBANDS_middleband', 'BBANDS_lowerband']
# MidPoint over period
output = np.c_[output, talib.MIDPOINT(close)]
cols += ['MIDPOINT']
```

図 1: TA-Lib でインジケータを計算するプログラム (一部)

上の図 1 からわかるように，TA-Lib を使うこと 1 行のコードでインジケータの計算を行うことができる．

3 進捗状況

3.1 プログラム

前回の発表の際に指摘された，「インジケータの計算をする際に使用するデータの値は一定の間隔ごとに取得しないといけない」，「データフレームに入れ続けることでデータが溢れてしまう」という 2 点を改良した．

まず，データの値をある一定の間隔でとってくる部分は，MT5 からとってきた tick データをどんどん取得していく中で，1 秒ごとにその時点で最新の tick データ resample をして ohlc の形で別のデータフレームに保存していくようにした．これにより，一定時間間隔の ohlc のデータフレームを作成することができた．図 2 は実際に 1 秒足で resample した結果である．秒足の部分に関しては指定できるので，任意の時間感覚で resample することができる．

次に，データフレームが溢れないようにする部分は，データフレームの中身がある程度の大きさになったら古いものから削除していくようにする．実際には，どんどん取得されてくる tick データをデータフレーム 1 に格納していき，その中から 1 秒ごとに resample して別のデータフレーム 2 に格納している．データフレーム 1 の中身をどんどん削除していけばデータフレーム 2 も対応して減っていく．実際に古いものから削除していくプログラムを動かした際の結果が図 3 である．

```
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
2020-10-01 11:57:40  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
2020-10-01 11:57:40  105.519  105.519  105.519  105.519
2020-10-01 11:57:41  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
2020-10-01 11:57:40  105.519  105.519  105.519  105.519
2020-10-01 11:57:41  105.519  105.519  105.519  105.519
2020-10-01 11:57:42  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
2020-10-01 11:57:40  105.519  105.519  105.519  105.519
2020-10-01 11:57:41  105.519  105.519  105.519  105.519
2020-10-01 11:57:42  105.519  105.519  105.519  105.519
2020-10-01 11:57:43  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 11:57:39  105.519  105.519  105.519  105.519
2020-10-01 11:57:40  105.519  105.519  105.519  105.519
2020-10-01 11:57:41  105.519  105.519  105.519  105.519
2020-10-01 11:57:42  105.519  105.519  105.519  105.519
2020-10-01 11:57:43  105.519  105.519  105.519  105.519
2020-10-01 11:57:44  105.519  105.519  105.519  105.519
```

図 2: 1 秒足の ohlc データの取得結果

```
                        open     high      low    close
time
2020-10-01 10:42:24  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 10:42:24  105.519  105.519  105.519  105.519
2020-10-01 10:42:25  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 10:42:24  105.519  105.519  105.519  105.519
2020-10-01 10:42:25  105.519  105.519  105.519  105.519
2020-10-01 10:42:26  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 10:42:25  105.519  105.519  105.519  105.519
2020-10-01 10:42:26  105.519  105.519  105.519  105.519
2020-10-01 10:42:27  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 10:42:25  105.519  105.519  105.519  105.519
2020-10-01 10:42:26  105.519  105.519  105.519  105.519
2020-10-01 10:42:27  105.519  105.519  105.519  105.519
2020-10-01 10:42:28  105.519  105.519  105.519  105.519
                        open     high      low    close
time
2020-10-01 10:42:26  105.519  105.519  105.519  105.519
2020-10-01 10:42:27  105.519  105.519  105.519  105.519
2020-10-01 10:42:28  105.519  105.519  105.519  105.519
2020-10-01 10:42:29  105.519  105.519  105.519  105.519
```

図 3: データフレームの中身を削除していく例

この 2 つを前回作成したゴールデンクロスのプログラムに組み込み実行してみたが，売買が行われなかった．これは本当に取引に値する動きがなかったのかプログラムに問題があったのか調査する．また，resample した後の 1 秒足のデータを描画する部分，インジケータの値の計算に使用する最適な値を見つける最適化を行うプログラムについても作成する必要がある．

3.2 オリジナリティの部分

実際に研究に組み込むことになるオリジナリティの部分は，先生と相談した結果，マルチンゲールやココモ式，ポートフォリオ選択等のオペレーションズ・リサーチのシステムを組み込んでいく方針にした．この中でどれを組み込むのかはそれぞれ調査した結果決定する．

4 おわりに

前回の指摘事項を修正したら新たな問題が出てきた．これに関しては修正を繰り返すしかないので相談しながら行っていく．中間発表まであまり時間がないためオリジナリティの部分も含め素早く対応していきたいと考えている．

参考文献

[1] https://qiita.com/ryoshi81/items/983c06e0bf859b280eba Accessed: July 30, 2020.

[2] https://note.nkmk.me/python-pandas-time-series-resample-asfreq/ Accessed: October 1, 2020.

[3] https://note.nkmk.me/python-pandas-drop/ Accessed: October 1, 2020.