量化择时和小波分析

量化择时下，有很多关于对大盘涨跌的预测的研究。对大盘的日涨跌的预测，这实际上是一个分类的问题。
本文使用了7个指标(最近M日最高价，最近M日最低价，当日成交额在最近M日成交额中占比，过去M日涨跌幅，最近M日均价，MACD，RSI)，来对大盘的日涨跌进行预测。
针对建模的输入，使用了小波分析的处理方式，来对各个金融时间序列做降噪处理，再作为输入进入分类模型。
从结果来看，小波降噪的结果不理想，降噪前后的预测结果没有显著差异。不过基于预测结果的择时策略效果还可以。
参考：量化择时系列二-水至清则鱼自现——小波分析与支持向量机择时研究，平安证券公司

In [245]:

```
import pywt   #小波分析库
import pandas as pd
import talib as ta
import numpy as np
from sklearn import preprocessing
from sklearn import svm
from WindPy import *
import matplotlib.pyplot as plt
plt.style.use('ggplot')
w.start()
```

```
COPYRIGHT (C) 2017 Wind Information Co., Ltd. ALL RIGHTS RESERVED.
IN NO CIRCUMSTANCE SHALL WIND BE RESPONSIBLE FOR ANY DAMAGES OR LOSSES
CAUSED BY USING WIND QUANT API FOR PYTHON.
```

一、小波分析浅谈

小波分析是数字信号处理领域中的一种重要方法。其主要作用是平滑和过滤，将序列中过于频率的噪声变动过滤掉。小波分析常用于金融时间序列的处理，因为金融时间序列往往是非平稳的，且存在很多噪声。
小波分析属时频分析的一种，主要依据下面三步来实现(Python开源库pywt是专门做小波分析的库)：

1. Decomposition——将信号序列分解为N层小波
2. Denoise——对于分解得到的小波，选择阈值并进行去噪处理(要注意的是不是对每个小波都要处理，一般选择几个进行处理)
3. Reconstruction——将去噪后的小波，和未处理的小波，重构得到处理后的信号

In [79]:

```
print('查看小波函数族：',pywt.families())
print('查看sym小波族下的所有函数：',pywt.wavelist(family='sym',kind='all'))
```

```
查看小波函数族： ['haar', 'db', 'sym', 'coif', 'bior', 'rbio', 'dmey', 'gaus', 'mexh', 'morl', 'cgau', 'shan', 'fbsp', 'cmor']
查看sym小波族下的所有函数： ['sym2', 'sym3', 'sym4', 'sym5', 'sym6', 'sym7', 'sym8', 'sym9', 'sym10', 'sym11', 'sym12', 'sym13', 'sym14', 'sym15', 'sym16', 'sym17', 'sym18', 'sym19', 'sym20']
```

下面演示了对上证综指时间序列的小波分解、降噪和重构

In [218]:

```
szzz = w.wsd("000001.SH", "close", "2007-01-01", "2017-12-13", "")
szzz_df = pd.DataFrame(szzz.Data[0],columns=['上证综指'])
szzz_df.index = [i.strftime('%Y-%m-%d') for i in szzz.Times]
szzz_df.head()
```

	上证综指
2007-01-04	2715.719
2007-01-05	2641.334
2007-01-08	2707.199
2007-01-09	2807.804
2007-01-10	2825.576

对上证综指时间序列进行小波分解，小波函数选择为sym8，分解层个数为4个，即一个主要趋势，和四个细节成分

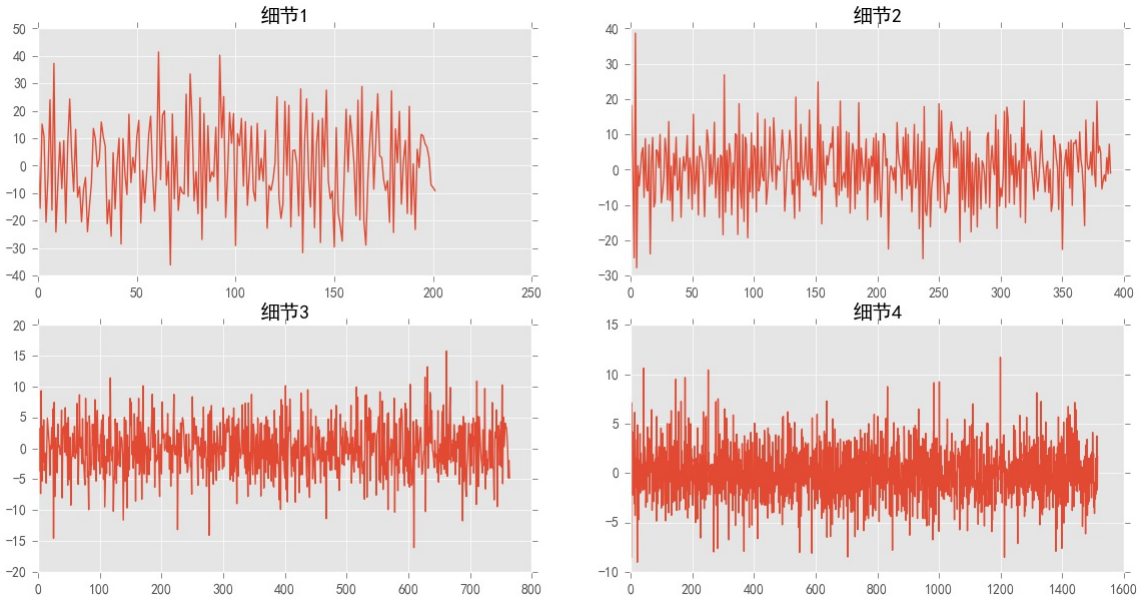
In [220]:

```
res_4 = pywt.wavedec(np.array(szzz_df['上证综指']),wavelet='sym8',mode='symmetric',level=4)
#四个参数分别 对应 输入的信号 ，小波名称， 信号扩展模式， 分解阶次
res_4_df = pd.DataFrame([res_4[0],res_4[1],res_4[2],res_4[3],res_4[4]],index=['主趋势','细节1','细节2','细节3','细节4']).T
res_4_df.head()
```

	主趋势	细节1	细节2	细节3	细节4
0	10941.557222	158.459710	13.776685	89.779976	98.423240
1	10992.098545	-87.425353	81.652149	41.928543	-38.744595
2	10950.256328	72.655379	-22.578860	42.045525	30.348337
3	10927.301781	37.801811	-112.971343	42.749231	-55.988051
4	10986.043605	-75.893808	178.609129	-93.941567	68.192835

In [222]:

```
fig = plt.figure(figsize=(14,7))       #设置图形大小
for j in [1,2,3,4]:
    plt.subplot(2,2,j)
    plt.plot(range(len(res1[j])),res1[j])
    plt.title('细节'+str(j))
plt.show()
```



从上图可以看到，相对而言，细节3和4更像是高频的噪声信号，故对这两个细节进行阈值降噪处理。这里涉及到阈值选择的问题。
通常来说，阈值选择有：1、无偏风险估计阈值(rigrsure) 2、固定阈值 3、启发式阈值 4、极大极小阈值 这些方法，在这里选择rigsure方法来确定阈值
这里不展开细节，关于信号去噪可以参考https://wenku.baidu.com/view/63d62a818762caaedd33d463.html (https://wenku.baidu.com/view/63d62a818762caaedd33d463.html)

In [211]:

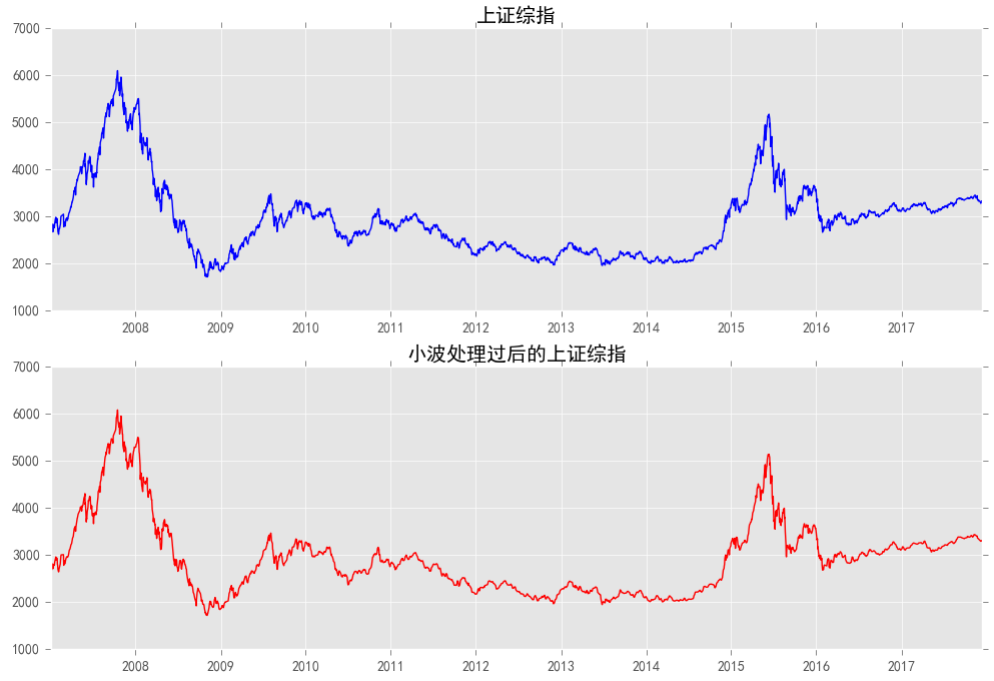
```
def threshold_cal(signal):
    '''rigrsure方法计算阈值'''
    signal = abs(signal)
    signal.sort()
    signal = signal**2
    list_risk_j = []; N = len(signal)
    for j in range(N):
        if j == 0:
            risk_j = 1 + signal[N-1]
        else:
            risk_j = (N-2*j + (N-j)*(signal[N-j]) + sum(signal[:j]))/N
        list_risk_j.append(risk_j)
    k = np.array(list_risk_j).argmin()
    threshold = np.sqrt(signal[k])
    return threshold
```

In [225]:

```
for j in [3,4]:     #对于高频的信号进行阈值处理
    signal = np.array(res_4[j])
    threshold = threshold_cal(signal)       #固定阈值方法 np.sqrt(2*np.log(len(signal)))
    res_4[j] = pywt.threshold(signal, threshold, 'soft')
rec_szzz = pywt.waverec(res_4, 'sym8')
```

In [233]:

```
szzz_df['小波滤波后的上证综指'] = rec_szzz
fig = plt.figure(figsize=(12,8))       #设置图形大小
plt.subplot(2,1,1)
plt.plot(szzz.Times,szzz_df['上证综指'],color='blue')
plt.title('上证综指')
plt.subplot(2,1,2)
plt.plot(szzz.Times,szzz_df['小波滤波后的上证综指'],color='red')
plt.title('小波处理过后的上证综指')
plt.show()
```



可以看到小波处理后的序列，相对平滑一些，可能可以避免原序列中的噪声对模型预测带来的负面影响

二、上证综指日涨跌预测

考虑利用7个指标(最近M日最高价，最近M日最低价，当日成交额在最近M日成交额中占比，过去M日涨跌幅，最近M日均价，MACD，RSI)，来对上证综指的日涨跌进行预测。 本例中选择的分类器为SVM(核函数默认)。要说明的是，对于上述这些特征进行了标准化处理。M选择为5。查看了不同的模型训练窗宽下对上证综指日涨跌的预测效果，汇总在res_svm_pred中。

In [236]:

```
class model:
    '''对数据进行建模预测'''
    def __init__(self, data, M, window, whether_wave_process=False):
        self.data = data
        self.M = M
        self.window = window
        self.whether_wave_process = whether_wave_process

    def wave_process(self):
        '''对数据进行小波处理(可选)'''
        data = self.data
        for col in ['HIGH','LOW','AMT','PCT_CHG','CLOSE','MACD','RSI']:
            res1 = pywt.wavedec(np.array(data[col]),wavelet='sym8',mode='symmetric',level=4)
            for j in [3,4]:     #对于高频的信号进行阈值处理
                signal = np.array(res1[j])
                threshold = threshold_cal(signal)       #固定阈值方法 np.sqrt(2*np.log(len(signal)))
                res1[j] = pywt.threshold(signal, threshold, 'soft')
            data[col] = pywt.waverec(res1, 'sym8')[:len(data)]   # 为什么小波重构后的序列 和 原序列的长度 存在差异呢？？？？
        self.to_use = data
        self.whether_wave_process = True

    def preprocess(self):
        '''生成相应的特征'''
        if self.whether_wave_process == False:
            data = self.data
        else:
            data = self.to_use
        data['近M日最高价'] = np.NaN; data['近M日最低价'] = np.NaN
        data['成交额占比'] = np.NaN; data['近M日涨跌幅'] = np.NaN; data['近M日均价'] = np.NaN
        for j in range(len(data)-self.M):
            index = data.index[(j+self.M-1)]
            data.loc[index,'近M日最高价'] = max(np.array(data[j:(j+self.M)]['HIGH']))
            data.loc[index,'近M日最低价'] = min(np.array(data[j:(j+self.M)]['LOW']))
            data.loc[index,'成交额占比'] = data.loc[index,'AMT']/sum(np.array(data[j:(j+self.M)]['AMT']))
            data.loc[index,'近M日涨跌幅'] = np.prod(np.array(data[j:(j+self.M)]['PCT_CHG'])/100+1) - 1
            data.loc[index,'近M日均价'] = np.mean(np.array(data[j:(j+self.M)]['CLOSE']))
        self.to_use = data[['近M日最高价','近M日最低价','成交额占比','近M日涨跌幅','近M日均价','MACD','RSI','Y']]
        self.to_use = self.to_use.dropna()

    def standardization(self):
        '''对所有特征进行标准化处理'''
        data = preprocessing.scale(np.array(self.to_use[self.to_use.columns[:-1]]))
        data = pd.DataFrame(data, index=self.to_use.index, columns=self.to_use.columns[:-1])
        data['Y'] = self.to_use['Y']
        self.to_use = data

    def svm(self):
        '''利用SVM模型进行建模预测'''
        self.to_use['预测'] = np.NaN
        for j in range(len(self.to_use)-self.window):
            train_data = self.to_use[j:(j+self.window)]
            classifier = svm.SVC(gamma=0.001)
            classifier.fit(train_data[train_data.columns[:-2]].values, np.array(train_data['Y']))
            index = self.to_use.index[j+self.window]
            predict_base = np.array(self.to_use.loc[index,self.to_use.columns[:-2]]).reshape(1,7)
            prediction = classifier.predict(predict_base)
            self.to_use.loc[index, '预测'] = prediction
        self.to_use = self.to_use.dropna()
```

In [243]:

```
wsd_data = w.wsd("000300.SH", "high,low,amt,pct_chg,close", "2005-01-01", "2017-06-30")
data_df = pd.DataFrame(wsd_data.Data, index=wsd_data.Fields, columns=wsd_data.Times).T
data_df['MACD'] = ta.MACD(np.array(data_df['CLOSE']),fastperiod=6,slowperiod=12,signalperiod=9)[0]
data_df['RSI'] = ta.RSI(np.array(data_df['CLOSE']),timeperiod=14)
data_df['Y'] = 0
data_df['Y'][data_df['PCT_CHG']>0] = 1
data_df = data_df.dropna()
data_df.head()
```

	HIGH	LOW	AMT	PCT_CHG	CLOSE	MACD	RSI	Y
2005-01-31 00:00:00.005	965.785	953.142	3.863574e+09	-1.478525	954.879	-5.367997	38.972539	0
2005-02-01 00:00:00.005	965.477	952.741	4.275707e+09	0.112266	955.951	-6.755962	39.512661	1
2005-02-02 00:00:00.005	1006.932	956.701	1.020290e+10	5.331026	1006.913	-0.577613	58.373871	1
2005-02-03 00:00:00.005	1014.187	992.155	1.005731e+10	-1.360396	993.215	1.375614	53.541186	0
2005-02-04 00:00:00.005	1021.025	989.939	9.549871e+09	2.380451	1016.858	5.613427	59.737097	1

In [244]:

```
res_svm_pred = pd.DataFrame(columns=['M','训练窗宽','总预测次数','成功次数','成功概率'])
for j in range(15,100,5):
    model1 = model(data_df,5,j)
    model1.preprocess()
    model1.standardization()
    model1.svm()
    res_svm_pred.loc[len(res_svm_pred)] = 5,j,len(model1.to_use),sum(model1.to_use['预测'] == model1.to_use['Y']),sum(model1.to_use['预测'] == model1.to_use['Y'])/len(model1.to_use)
res_svm_pred
```

	M	训练窗宽	总预测次数	成功次数	成功概率
0	5.0	15.0	2995.0	1601.0	0.534558
1	5.0	20.0	2990.0	1649.0	0.551505
2	5.0	25.0	2985.0	1632.0	0.546734
3	5.0	30.0	2980.0	1658.0	0.556376
4	5.0	35.0	2975.0	1590.0	0.534454
5	5.0	40.0	2970.0	1631.0	0.549158
6	5.0	45.0	2965.0	1603.0	0.540641
7	5.0	50.0	2960.0	1638.0	0.553378
8	5.0	55.0	2955.0	1587.0	0.537056
9	5.0	60.0	2950.0	1609.0	0.545424
10	5.0	65.0	2945.0	1591.0	0.540238
11	5.0	70.0	2940.0	1620.0	0.551020
12	5.0	75.0	2935.0	1586.0	0.540375
13	5.0	80.0	2930.0	1599.0	0.545734
14	5.0	85.0	2925.0	1607.0	0.549402
15	5.0	90.0	2920.0	1612.0	0.552055
16	5.0	95.0	2915.0	1602.0	0.549571

三、上证综指日涨跌预测——小波降噪

与二中相比，这里增加了对输入序列的小波处理(分解、降噪和重构)。对上证综指日涨跌的预测效果，汇总在res_pwsvm_pred中。

In [241]:

```
wsd_data = w.wsd("000300.SH", "high,low,amt,pct_chg,close", "2005-01-01", "2017-06-30")
data_df = pd.DataFrame(wsd_data.Data, index=wsd_data.Fields, columns=wsd_data.Times).T
data_df['MACD'] = ta.MACD(np.array(data_df['CLOSE']),fastperiod=6,slowperiod=12,signalperiod=9)[0]
data_df['RSI'] = ta.RSI(np.array(data_df['CLOSE']),timeperiod=14)
data_df['Y'] = 0
data_df['Y'][data_df['PCT_CHG']>0] = 1
data_df = data_df.dropna()
data_df.head()
```

	HIGH	LOW	AMT	PCT_CHG	CLOSE	MACD	RSI	Y
2005-01-31 00:00:00.005	965.785	953.142	3.863574e+09	-1.478525	954.879	-5.367997	38.972539	0
2005-02-01 00:00:00.005	965.477	952.741	4.275707e+09	0.112266	955.951	-6.755962	39.512661	1
2005-02-02 00:00:00.005	1006.932	956.701	1.020290e+10	5.331026	1006.913	-0.577613	58.373871	1
2005-02-03 00:00:00.005	1014.187	992.155	1.005731e+10	-1.360396	993.215	1.375614	53.541186	0
2005-02-04 00:00:00.005	1021.025	989.939	9.549871e+09	2.380451	1016.858	5.613427	59.737097	1

In [242]:

```
res_pwsvm_pred = pd.DataFrame(columns=['M','训练窗宽','总预测次数','成功次数','成功概率'])
for j in range(15,100,5):
    model2 = model(data_df,5,j)
    model2.wave_process()    #注意这里多了一步---对输入的序列进行了小波降噪处理
    model2.preprocess()
    model2.standardization()
    model2.svm()
    res_pwsvm_pred.loc[len(res_pwsvm_pred)] = 5,j,len(model2.to_use),sum(model2.to_use['预测'] == model2.to_use['Y']),sum(model2.to_use['预测'] == model2.to_use['Y'])/len(model2.to_use)
res_pwsvm_pred
```

	M	训练窗宽	总预测次数	成功次数	成功概率
0	5.0	15.0	2995.0	1601.0	0.534558
1	5.0	20.0	2990.0	1644.0	0.549833
2	5.0	25.0	2985.0	1632.0	0.546734
3	5.0	30.0	2980.0	1648.0	0.553020
4	5.0	35.0	2975.0	1590.0	0.534454
5	5.0	40.0	2970.0	1626.0	0.547475
6	5.0	45.0	2965.0	1603.0	0.540641
7	5.0	50.0	2960.0	1627.0	0.549662
8	5.0	55.0	2955.0	1587.0	0.537056
9	5.0	60.0	2950.0	1607.0	0.544746
10	5.0	65.0	2945.0	1590.0	0.539898
11	5.0	70.0	2940.0	1615.0	0.549320
12	5.0	75.0	2935.0	1585.0	0.540034
13	5.0	80.0	2930.0	1601.0	0.546416
14	5.0	85.0	2925.0	1607.0	0.549402
15	5.0	90.0	2920.0	1611.0	0.551712
16	5.0	95.0	2915.0	1600.0	0.548885

小波处理对于预测结果的提升效果不显著，因为二与三中的上证综指预测结果类似，没有显著差别。

四、基于涨跌预测的择时策略

基于上述预测结果中的最好预测结果，将其转换为一个择时策略。具体如下：若当天的指数头寸为空，而模型预测下一个交易日会上涨(预测=1)，这时在下一个交易日开盘的时候满仓买 入指数，如果模型预测下一个交易日仍然是上涨，则继续持有该头寸；当到了某日，模型预测下一天的市场是下跌(预测=0)，这时清空持有的指数头寸。在持仓指数时，资金在指数成分股之间平均分配。
在这里选择M=5，训练窗宽为30的模型的预测结果来构造择时策略。

In [256]:

```
wsd_data = w.wsd("000300.SH", "high,low,amt,pct_chg,close", "2005-01-01", "2017-06-30")
data_df = pd.DataFrame(wsd_data.Data, index=wsd_data.Fields, columns=wsd_data.Times).T
data_df['MACD'] = ta.MACD(np.array(data_df['CLOSE']),fastperiod=6,slowperiod=12,signalperiod=9)[0]
data_df['RSI'] = ta.RSI(np.array(data_df['CLOSE']),timeperiod=14)
data_df['Y'] = 0
data_df['Y'][data_df['PCT_CHG']>0] = 1
data_df = data_df.dropna()
model1 = model(data_df,5,30)
model1.wave_process()
model1.preprocess()
model1.standardization()
model1.svm()
res_wave_svm = model1.to_use
res_wave_svm.index = [i.strftime('%Y-%m-%d') for i in list(res_wave_svm.index)]
res_wave_svm.head()
```

	近M日最高价	近M日最低价	成交额占比	近M日涨跌幅	近M日均价	MACD	RSI	Y	预测
2005-03-29	-1.779193	-1.799097	-1.054481	-0.536255	-1.788967	-0.365502	-1.435556	0	0.0
2005-03-30	-1.784513	-1.801563	-0.219027	-0.555832	-1.791740	-0.347182	-1.510974	0	0.0
2005-03-31	-1.789504	-1.803784	0.949348	-0.576409	-1.793823	-0.325246	-1.469895	1	0.0
2005-04-01	-1.776679	-1.804754	1.779763	0.027368	-1.793320	-0.294151	-0.596442	1	0.0
2005-04-04	-1.776679	-1.804754	2.061541	0.098681	-1.792594	-0.253581	-0.865633	0	0.0

In [265]:

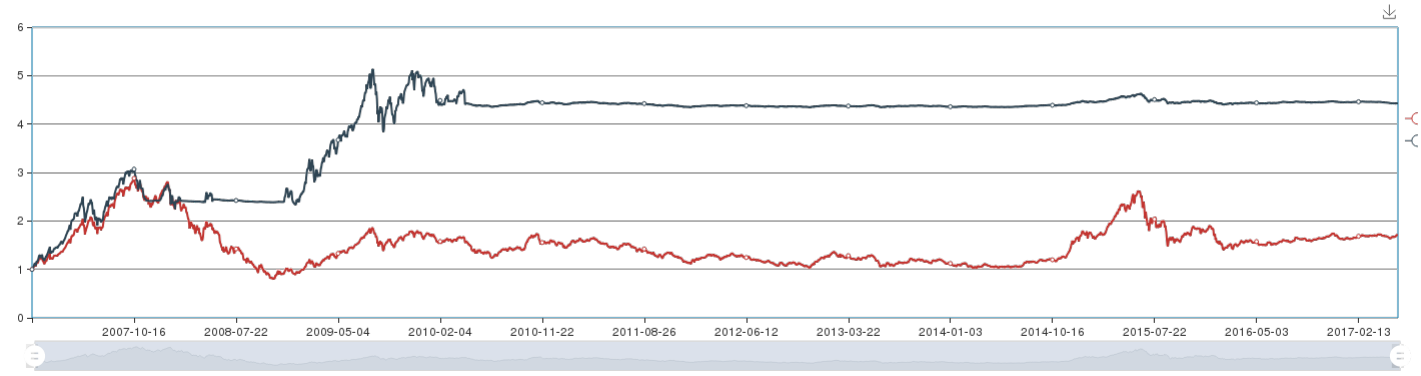
```
from WindAlgo import * #引入回测框架

def initialize(context):                  #定义初始化函数  initialize函数只在回测开始前调用一次之后不再调用
    context.capital = 1000000             #回测的初始资金
    context.securities = ['000001.SZ']       #回测标的
    context.start_date = "20070101"       #回测开始时间
    context.end_date = "20170601"         #回测结束时间
    context.period = 'd'                  #策略运行周期, 'd' 代表日, 'm'代表分钟
    context.benchmark = '000300.SH'       #上证50为基准
    context.res_wave_svm = res_wave_svm

def handle_data(bar_datetime, context, bar_data):                #定义策略函数

    bar_datetime_str = bar_datetime.strftime('%Y-%m-%d')
    position = wa.query_position()
    list_stock = w.wset("sectorconstituent", "date="+bar_datetime_str+";windcode=000300.SH").Data[1]    ## 选择上证50成分股 作为 股票池
    is_success,bar_data = wa.change_securities(list_stock)
    if is_success == True:
        if (context.res_wave_svm.loc[bar_datetime_str,'预测']==0)&(len(position)>200):  #预测为跌 且 持仓了指数
            res = wa.batch_order.sell_all(price='close', volume_check=False, no_quotation='skip')  #清仓
        if (context.res_wave_svm.loc[bar_datetime_str,'预测']==1)&(len(position)<20):   #预测为涨 且 未持仓指数
            res = wa.batch_order.change_to(list_stock,0.99999,price='open', volume_check=False, no_quotation='skip')  #满仓指数

wa = BackTest(init_func = initialize, handle_data_func=handle_data) #实例化回测对象
res = wa.run(show_progress=True)                                    #调用run()函数开始回测,show_progress可用于指定是否显示回测净值曲线图
nav_df=wa.summary('nav')                                            #获取回测结果
```



<img src="https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=674580860,3361601546&fm=27&gp=0.jpg
(https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=674580860,3361601546&fm=27&gp=0.jpg)" width=500 height=280 align=left />