# 使用模块导入

import numpy as np

import pandas as pd

import datetime

import math

import time

from datetime import date

import re

'''

银行行业增强策略-银行翻倍策略

'''

# 参数设置：

name = '银行翻倍策略，择时'

# 对比基准设置'801780.SL''000300.SH' '000905.SH'

num1 = '000905.SH'

# 全仓(1)或者半仓(0.5)

num4 = 1

# 选股条件

num2 = '中证500股票 上市天数>200'

# 目标选股数量

num3 = 5

#长期空仓天数

num5 = 250

#市场空仓对应的PB分位数阈值

num6 = 0.8

def init(context):

set\_benchmark(num1)

get\_iwencai(num2)

context.signal = True

context.b = True

g.output\_stocklist = {}

g.day = 0

g.step = 5

def before\_trading(context):

#完成选股

if g.day/g.step==0:

g.stock = GrahamStockFilter(context)

for s in g.stock:

g.output\_stocklist[s] = num4 / len(g.stock)

set\_stock\_picks(g.output\_stocklist)

g.output\_stocklist.clear()

def handle\_bar(context,bar\_dict):

if g.day%g.step==0:

patt=r'择时'

pattern = re.compile(patt)

result = pattern.findall(name)

if result:

period, context.signal = time\_select(context)

if context.signal == 'kongcang' and context.b is True:

context.b = False

context.date = get\_datetime() + datetime.timedelta(days=period)

for stock in list(context.portfolio.stock\_account.positions.keys()):

order\_target\_value(stock, 0)

log.info('空仓信号在' + get\_datetime().strftime('%Y-%m-%d') + '触发，空仓' + str(period) + '天')

if context.b is False and get\_datetime() > context.date:

context.b = True

if context.signal == 'chaodi':

context.b = True

context.signal = True

if context.signal is True and context.b is True:

#buylist = GrahamStockFilter(context)

for hold in list(context.portfolio.stock\_account.positions.keys()):

if hold not in list(g.stock):

order\_target\_value(hold, 0)

for stock in g.stock:

order\_target\_percent(stock, num4 / len(g.stock))

else:

#buylist = GrahamStockFilter(context)

for hold in list(context.portfolio.stock\_account.positions):

if hold not in list(g.stock):

order\_target\_value(hold, 0)

for stock in g.stock:

order\_target\_percent(stock, num4 / len(g.stock))

def after\_trading(context):

g.day += 1

def GrahamStockFilter(context,overflow=0):

df1 = Data(context)

buylist = Yhfb(context,df1)

log.info(get\_datetime().strftime('%Y-%m-%d') + '选股为:' + str(buylist)[:])

return buylist

# 数据获取

def Data(context):

yesterday = get\_last\_datetime().strftime('%Y%m%d')

today = get\_datetime().strftime('%Y%m%d')

stock\_list = context.iwencai\_securities

#stock\_list = get\_industry\_stocks('T1901')

value = get\_price(stock\_list, None, today, '1d', ['is\_paused', 'is\_st','open','high\_limit','low\_limit'], True, None, 1, is\_panel=1)

stock\_list = [stock for stock in stock\_list if value['is\_paused'].ix[0,stock] == 0]

stock\_list = [stock for stock in stock\_list if value['is\_st'].ix[0,stock] == 0]

stock\_list = [stock for stock in stock\_list if value['open'].ix[0,stock] != value['high\_limit'].ix[0,stock] and value['open'].ix[0,stock]!= value['low\_limit'].ix[0,stock]]

log.info('股票池数量：%d' %len(stock\_list))

#获取财务数据

q = query(valuation.symbol, income.basic\_eps, valuation.pb,valuation.pe, valuation.market\_cap,

income\_one\_season.profit\_from\_operations, valuation.capitalization, profit\_one\_season.roe,balance.total\_liabilities,

growth\_one\_season.opt\_profit\_growth\_ratio).filter(valuation.symbol.in\_(stock\_list)).order\_by(

valuation.symbol)

df1 = get\_fundamentals(q, date=yesterday)

df1 = df1.set\_index(df1['valuation\_symbol'])

df1 = df1[df1['profit\_one\_season\_roe'] > 0]

df1['close'] = list([float("nan")] \* (len(df1.index)))

true\_list = list(df1.index)

close\_p = history(true\_list, ['close','turnover'], 1, '1d', skip\_paused=False, fq='pre', is\_panel=1)

for stock in true\_list:

df1.loc[stock, 'close'] = close\_p['close'][stock].values[0]

df1.loc[stock, 'volume'] = close\_p['turnover'][stock].values[0]/10000

return df1

def Yhfb(context,df1):

df1 = df1[df1['growth\_one\_season\_opt\_profit\_growth\_ratio']>0]

#pb小于3的股票

df1 = df1[df1['valuation\_pb']<3]

#成交额筛选

df1 = df1[df1['volume']>5000]

df1['double\_time'] = df1.apply(

lambda row: round(math.log(2.0 \* row['valuation\_pb']/(1+row['growth\_one\_season\_opt\_profit\_growth\_ratio']/100), (1.0 + row['profit\_one\_season\_roe'] / 100)), 2), axis=1)

df\_double = df1.sort\_values('double\_time', ascending=True)

df1 = df\_double[:num3]

buylist = list(df1['valuation\_symbol'].values)

return buylist

def time\_select(context):

yesterday = get\_last\_datetime().strftime('%Y-%m-%d')

qt = query(valuation.symbol, valuation.pb).filter()

df = get\_fundamentals(qt, date=yesterday)

# 计算昨天市场所有股票PB值的分位数

factor\_quantiles = df.dropna().quantile([num6])

PB = factor\_quantiles.iloc[0].values

'''

# 计算昨天市场上跌停的股票占比

stock\_list = list(get\_all\_securities('stock', yesterday).index)

pct\_all = history(stock\_list, ['quote\_rate'], 5, '1d',

skip\_paused=False, fq=None, is\_panel=1)

pct\_list = []

for i in range(5):

values = list(pct\_all.iloc[0,i,])

pct = (len([x for x in values if x <= -9.5])) / len(stock\_list)

pct\_list.append(pct)

pct = max(pct\_list)

log.info('上周最大跌停股票占比: %.2f%%' %(pct\*100))

'''

# 择时空仓条件设置

if PB >= 10:

log.info('空仓',PB)

return num5, 'kongcang'

#elif pct > 0.1:

#return num6, False

#elif PB <= 5:

#return 0, 'chaodi'

else:

return 0, True