from quantopian.algorithm import (attach\_pipeline,pipeline\_output,order\_optimal\_portfolio,)

import quantopian.optimize as opt

from quantopian.pipeline import Pipeline

from quantopian.pipeline.data.psychsignal import stocktwits

from quantopian.pipeline.factors import SimpleMovingAverage

from quantopian.pipeline.filters import QTradableStocksUS

from quantopian.pipeline.experimental import risk\_loading\_pipeline

from quantopian.pipeline.data.builtin import USEquityPricing

def initialize(context):

# Restricciones &&

context.max\_lever = 0.98

context.max\_posTam = 0.025

# Volumen de Negocios

context.max\_Volum = 0.85

#context.assets = [sid(24), sid(5061), sid(8554)]

context.assets = [sid(37514)]

# adjunta la informacion de pipelines

attach\_pipeline(make\_pipeline(),'data\_pipe')

attach\_pipeline(risk\_loading\_pipeline(),'risk\_pipe')

# llama la funcion rebalance cada dia, 15 minutos despues que el mercado abre

schedule\_function(

rebalance,

date\_rules.week\_start(),

time\_rules.market\_open(minutes=15),

)

def handle\_data(context, data):

VolumHistory = data.history(context.assets, fields="volume", bar\_count=20, frequency="1d")

return VolumHistory

def before\_trading\_start(context, data):

# Obtiene la salida del pipeline y la guarda en context

context.output = pipeline\_output('data\_pipe')

context.risk\_factor\_betas = pipeline\_output('risk\_pipe')

# Pipeline definition

def make\_pipeline():

sentiment\_score = SimpleMovingAverage(

inputs=[stocktwits.bull\_minus\_bear],

window\_length=5,

mask=QTradableStocksUS()

)

close\_price = USEquityPricing.close.latest

return Pipeline(

columns={

'close\_price':close\_price,

'sentiment\_score': sentiment\_score,

},

screen=sentiment\_score.notnull()

)

def rebalance(context, data):

objective = opt.MaximizeAlpha(

context.output.sentiment\_score

)

constrain\_posTam = opt.PositionConcentration.with\_equal\_bounds(-1.5\*context.max\_posTam,context.max\_posTam)

dollar\_net = opt.DollarNeutral()

max\_lever = opt.MaxGrossExposure(context.max\_lever)

max\_Volum = opt.MaxTurnover(context.max\_Volum)

factor\_risk\_constraints = opt.experimental.RiskModelExposure(

context.risk\_factor\_betas,

version=opt.Newest

)

order\_optimal\_portfolio(

objective=objective,

constraints=[

max\_lever,

dollar\_net,

constrain\_posTam,

max\_Volum,

factor\_risk\_constraints,

]

)