Reporte Laboratorio 8

Gabriel Octavio Lozano Pinzón

1 de mayo de 2020

En este reporte damos los resultados de modificaciones al algoritmo para trading que se encuentra en `https://github.com/gjhernandezp/AT/blob/master/Sample%20Mean%20Reversion%20Example.py` , para poder comparar nuestras modificaciones con la linea base elegimos un tiempo definido, en nuestro caso de enero de 2011 a enero de 2012 (no se escogen los ultimos seis meses por que han sido una anormalidad muy grande en el comportamiento de los stocks) y corremos el algoritmo. En nuestro caso esta sera la linea base, nuestro objetivo es entender y poder mejorar el algoritmo moviendo sus parametros. Para este algoritmo nos concentraremos en los parametros de `RETURNS_LOOKBACK_DAYS`, `MAX_POSITION_CONCENTRATION`, planeación(scheduling) y valores de altos y bajos.

Nuestra linea base tiene como `RETURNS_LOOKBACK_DAYS` un valor de 5 , `MAX_POSITION_CONCENTRATION` tiene un valor de 0,001, el algoritmo se corre cada semana despues de 1 hora y media de abrirse el mercado, los filtros para los percentiles altos y bajos son de 10 % ambos. Con esto obtenemos los resultados de la Figura 1.

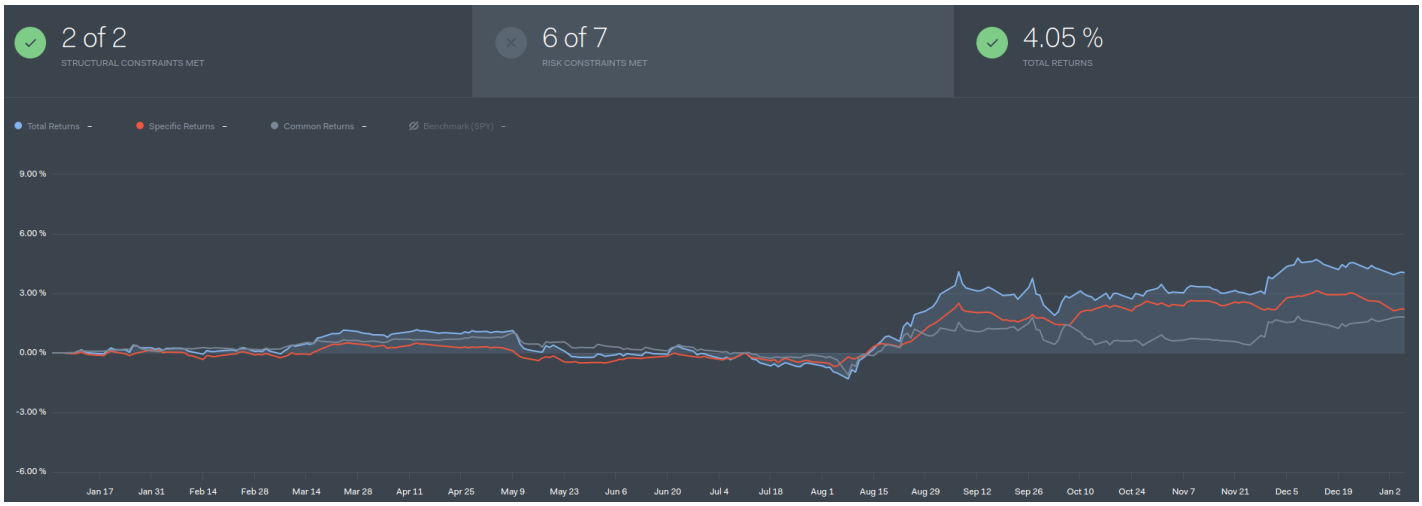

Figura 1: Resultados de la linea base del algoritmo

En el siguiente experimento hacemos que el algoritmo empiece desde el miercoles en lugar de una vez a la semana lo hacemos todos los dias, y hacemos que el algoritmo solo pueda ver 2 dias hacia atras. Con esto obtenemos la Figura 2 que indica que actualizar tan rapidamente los stocks con tan poca información no es buena idea.

Para el siguiente experimento consideramos que el algoritmo se ejecute cada semana con información de 6 dias y que además aumente el `MAX_POSITION_CONCENTRATION` a 0,05 , los resultados que se indican en la Figura 3 indican que nuestras perdidas y nuestras ganancias son más grandes a las de la linea base, es un algoritmo más .ªgresivo".

Ahora deberíamos pensar en que el algoritmo tuvo poca información en el primer experimento. Por tanto vamos a rebalancear teniendo en cuenta que se actualiza todos los dias pero con más información de por medio, es decir en vez de dos días le vamos a dar 10 y dejar el resto de variables como en el anterior experimento. Los resultados se pueden ver en la Figura 4, muestran que no solo se minimizan las perdidas tambien se maximizan las ganancias.

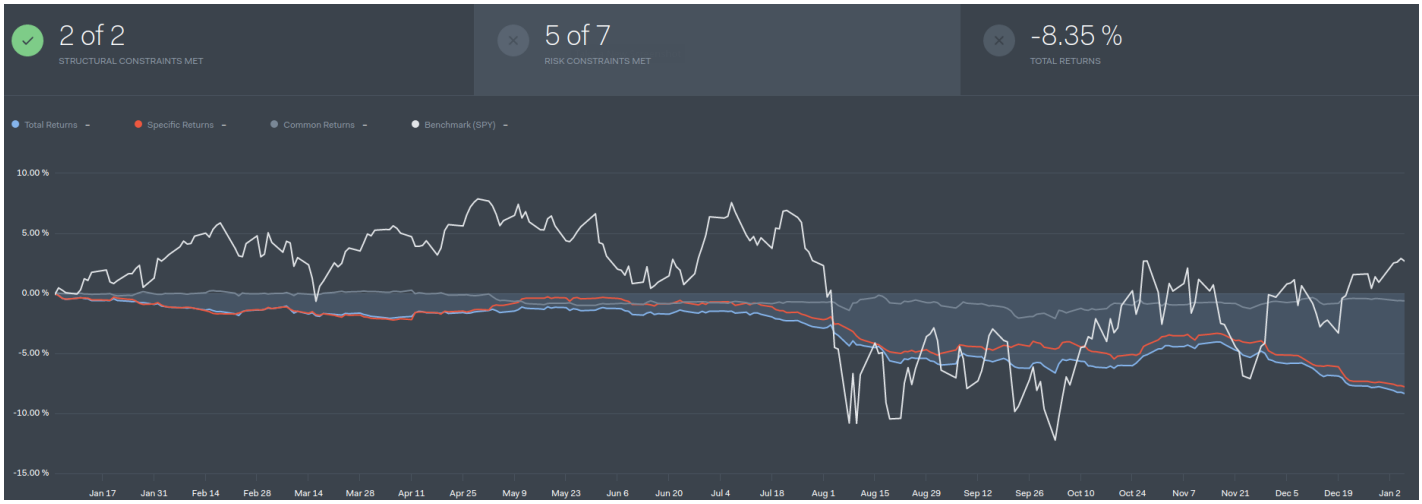

Figura 2: Resultados del primer experimento

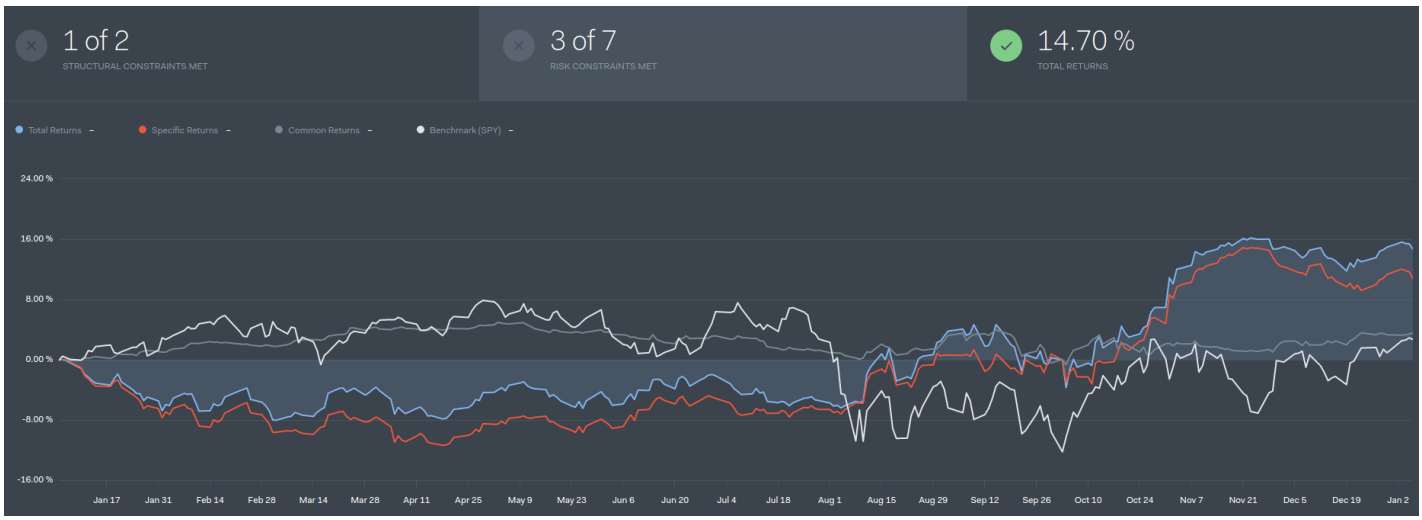

Figura 3: Resultados del segundo experimento

Para este experimento modificamos la forma en la que se toman los percentiles de altos y bajos para ir largo y corto respectivamente. Los bajos los dejamos en 13 % y los altos los dejamos en 7 %. El experimento que se muestra en la Figura 5 evidencia que el cambio no es significativo y no hace gran diferencia comparado a los anteriores.

Para el último experimento consideramos que debemos cambiar el universo así que en vez de tomar todo el universo de las acciones tomamos únicamente las 1000 con mejor revenue". De esta forma tenemos el quinto experimento que nos muestra que el algoritmo es muy similar SPY y que de hecho se pierde dinero.

El código con el cual se termina y que da un resultado de 54 % de ganancia es :

```
import quantopian.algorithm as algo
import quantopian.optimize as opt
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data.builtin import USEquityPricing
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.filters import QTradableStocksUS

MAX_GROSS_EXPOSURE = 1.0
MAX_POSITION_CONCENTRATION = 0.05
RETURNS_LOOKBACK_DAYS = 10


```

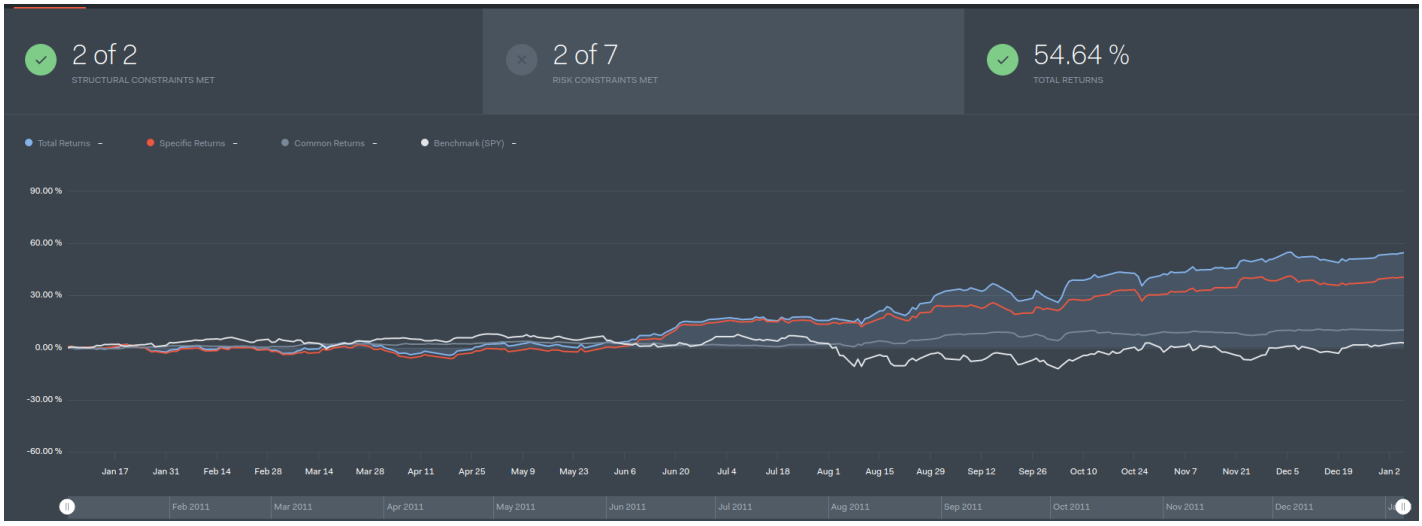

Figura 4: Resultados del experimento 3

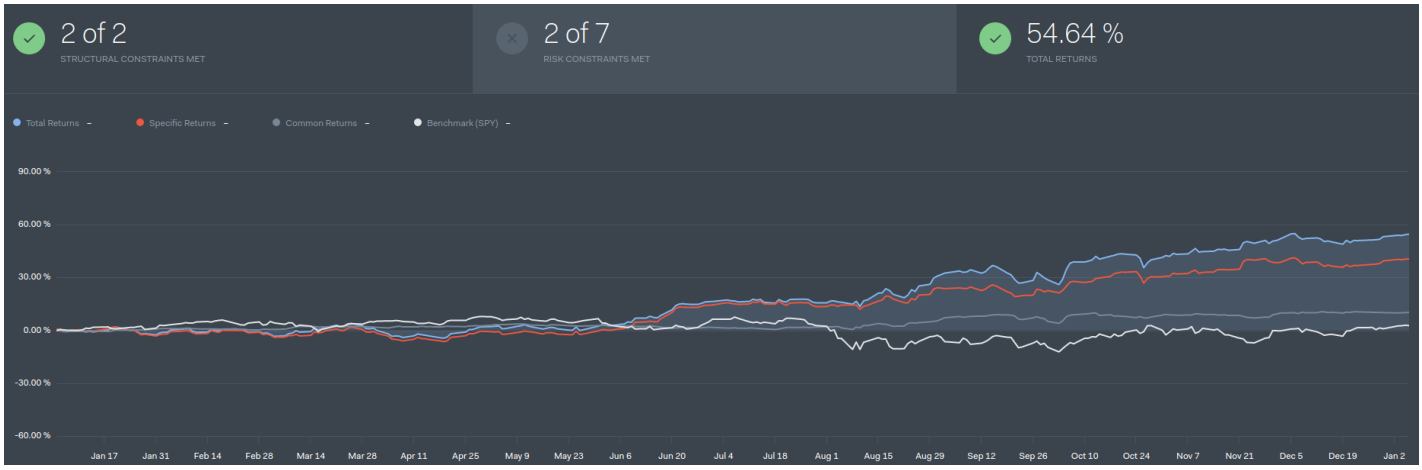

Figura 5: Resultados cuarto experimento

```
def initialize(context):
    algo.schedule_function(
        rebalance,
        #algo.date_rules.week_start(days_offset = 3),
        algo.date_rules.every_day(),
        algo.time_rules.market_open(hours=1, minutes=30)
    )
    algo.attach_pipeline(make_pipeline(context), 'mean_reversion_example')

def make_pipeline(context):
    universe = QTradableStocksUS()
    recent_returns = Returns(
        window_length=RETURNS_LOOKBACK_DAYS,
        mask=universe
    )
    recent_returns_zscore = recent_returns.zscore()
    low_returns = recent_returns_zscore.percentile_between(0,10)
    high_returns = recent_returns_zscore.percentile_between(90,100)
    securities_to_trade = (low_returns | high_returns)
    pipe = Pipeline(
        columns={
            'recent_returns_zscore': recent_returns_zscore
        },
        screen=securities_to_trade
```

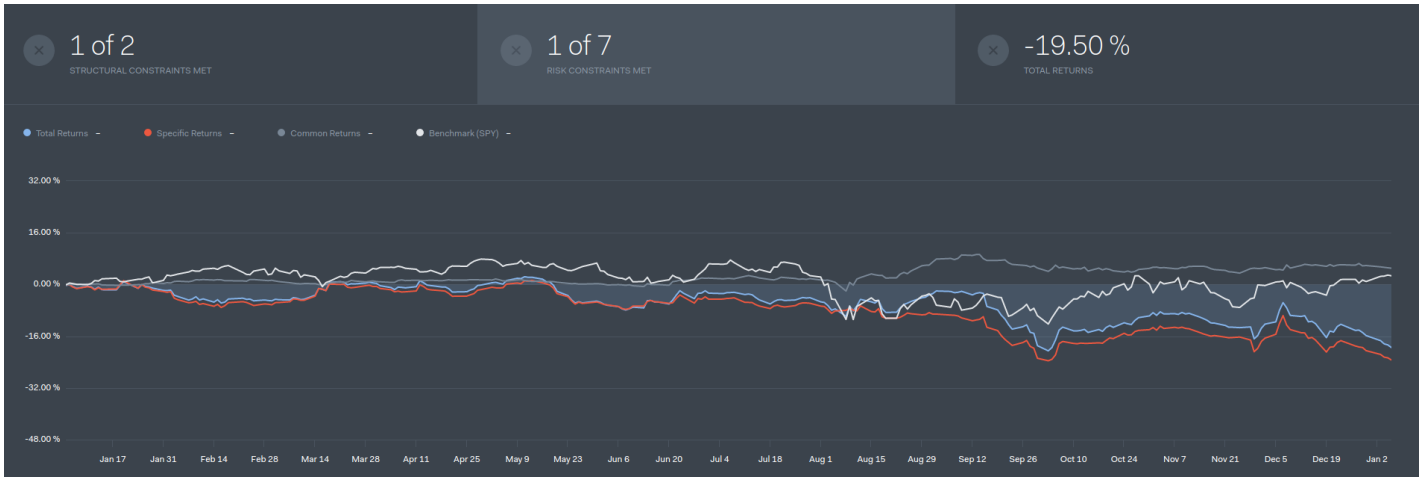

Figura 6: Resultados del quinto experimento

```
    )
    return pipe

def before_trading_start(context, data):
    context.output = algo.pipeline_output('mean_reversion_example')
    context.recent_returns_zscore = context.output['recent_returns_zscore']

def rebalance(context, data):
    objective = opt.MaximizeAlpha(-context.recent_returns_zscore)
    max_gross_exposure = opt.MaxGrossExposure(MAX_GROSS_EXPOSURE)
    max_position_concentration = opt.PositionConcentration.with_equal_bounds(
        -MAX_POSITION_CONCENTRATION,
        MAX_POSITION_CONCENTRATION
    )
    dollar_neutral = opt.DollarNeutral()
    constraints = [
        max_gross_exposure,
        max_position_concentration,
        dollar_neutral,
    ]
    algo.order_optimal_portfolio(objective, constraints)
```

Por último para analizar las acciones vendidas tomamos BKI de la cual se compraron 1500USD el primero de enero de 2011. En la Figura 7 se puede notar que la acción crece durante el periodo que le sigue a la compra usando yahoo Finance para obtener la información. Además contamos con la información provista para RDN con la cual se vendió 1700 USD y se puede notar como disminuyeron de precio.

Figura 7: Comportamiento de una acción que se compro usando en el algoritmo

Figura 8: Comportamiento de una acción que se vendió usando en el algoritmo