


ai-自动 测试

me_robot

1小时前 #1

 克隆策略

基础参数配置

class conf:

start_date = '2006-01-01'

end_date='2017-07-19'

split_date 之前的数据用于训练, 之后的数据用作效果评估

split_date = '2011-01-01'

D.instruments: https://bigquant.com/docs/data_instruments.html

instruments = D.instruments(start_date, split_date)

机器学习目标标注函数

如下标注函数等价于 $\min(\max((\text{持有期间的收益} * 100), -20), 20) + 20$ (后面的M.fast_auto_labeler会做取整操作)# 说明: max/min这里将标注分数限定在区间 $[-20, 20]$, +20将分数变为非负数 (Stock Ranker要求标注分数非负整数)

label_expr = ['return * 100', 'where(label > {0}, {0}, where(label < -{0}, -{0}, label)) + {0}'.format(20)]

持有天数, 用于计算label_expr中的return值(收益)

hold_days = 5

特征 https://bigquant.com/docs/data_features.html, 你可以通过表达式构造任何特征

features = [

'close_5/close_0', # 5日收益

'close_10/close_0', # 10日收益

'close_20/close_0', # 20日收益

'avg_amount_0/avg_amount_5', # 当日/5日平均交易额

'avg_amount_5/avg_amount_20', # 5日/20日平均交易额

'rank_avg_amount_0/rank_avg_amount_5', # 当日/5日平均交易额排名

'rank_avg_amount_5/rank_avg_amount_10', # 5日/10日平均交易额排名

'rank_return_0', # 当日收益

```

        'rank_return_5', # 5日收益
        'rank_return_10', # 10日收益

        'rank_return_0/rank_return_5', # 当日/5日收益排名
        'rank_return_5/rank_return_10', # 5日/10日收益排名

        'pe_ttm_0 < 50',
        'st_status_0 >0',
        'list_days_0 >99',
    ]

# 给数据做标注：给每一行数据（样本）打分，一般分数越高表示越好
m1 = M.fast_auto_labeler.v6(
    instruments=conf.instruments, start_date=conf.start_date, end_date=conf.end_date,
    split_date=conf.split_date,
    label_expr=conf.label_expr, hold_days=conf.hold_days,
    benchmark='000300.SHA', sell_at='open', buy_at='open')
# 计算特征数据
m2 = M.general_feature_extractor.v5(
    instruments=conf.instruments, start_date=conf.start_date, end_date=conf.end_date,
    split_date=conf.split_date,
    features=conf.features)
# 数据预处理：缺失数据处理，数据规范化，T.get_stock_ranker_default_transforms为
# StockRanker模型做数据预处理
m3 = M.transform.v2(
    data=m2.data, transforms=T.get_stock_ranker_default_transforms(),
    drop_null=True, astype='int32', except_columns=['date', 'instrument'],
    clip_lower=0, clip_upper=200000000)
# 合并标注和特征数据
m4 = M.join.v2(data1=m1.data, data2=m3.data, on=['date', 'instrument'],
    sort=True)
# StockRanker机器学习训练
m5 = M.stock_ranker_train.v3(training_ds=m4.data, features=conf.features)

# 每一个节点都可以点击展开
m5.plot_model()

## 量化回测 https://bigquant.com/docs/module\_trade.html
# 回测引擎：准备数据，只执行一次
def prepare(context):
    # context.start_date / end_date，回测的时候，为trader传入参数；在实盘运行的时候，由系统替换为实盘日期

```

```

n1 = M.general_feature_extractor.v5(
    instruments=D.instruments(),
    start_date=context.start_date, end_date=context.end_date,
    model_id=context.options['model_id'])
n2 = M.transform.v2(
    data=n1.data, transforms=T.get_stock_ranker_default_transforms()
,
    drop_null=True, astype='int32', except_columns=['date', 'instrument'],
    clip_lower=0, clip_upper=200000000)
n3 = M.stock_ranker_predict.v2(model_id=context.options['model_id'],
data=n2.data)
context.instruments = n3.instruments
context.options['predictions'] = n3.predictions

```

回测引擎：初始化函数，只执行一次

```
def initialize(context):
```

```
    # 加载预测数据
```

```
    context.ranker_prediction = context.options['predictions'].read_df()
```

```
    # 系统已经设置了默认的交易手续费和滑点，要修改手续费可使用如下函数
```

```
    context.set_commission(PerOrder(buy_cost=0.0003, sell_cost=0.0013, margin_cost=5))
```

```
    # 预测数据，通过options传入进来，使用 read_df 函数，加载到内存 (DataFrame)
```

```
    # 设置买入的股票数量，这里买入预测股票列表排名靠前的5只
```

```
    stock_count = 5
```

```
    # 每只的股票的权重，如下的权重分配会使得靠前的股票分配多一点的资金，[0.339160,
0.213986, 0.169580, ...]
```

```
    context.stock_weights = T.norm([1 / math.log(i + 2) for i in range(0
, stock_count)])
```

```
    # 设置每只股票占用的最大资金比例
```

```
    context.max_cash_per_instrument = 0.2
```

回测引擎：每日数据处理函数，每天执行一次

```
def handle_data(context, data):
```

```
    # 按日期过滤得到今日的预测数据
```

```
    ranker_prediction = context.ranker_prediction[
```

```
        context.ranker_prediction.date == data.current_dt.strftime('%Y-%m-%d')]

```

```
    # 1. 资金分配
```

```
    # 平均持仓时间是hold_days，每日都将买入股票，每日预期使用 1/hold_days 的资金
```

```
    # 实际操作中，会存在一定的买入误差，所以在前hold_days天，等量使用资金；之后，尽量使用剩余资金（这里设置最多用等量的1.5倍）

```

```

    is_staging = context.trading_day_index < context.options['hold_days']
] # 是否在建仓期间 (前 hold_days 天)
    cash_avg = context.portfolio.portfolio_value / context.options['hold_days']
    cash_for_buy = min(context.portfolio.cash, (1 if is_staging else 1.5) * cash_avg)
    cash_for_sell = cash_avg - (context.portfolio.cash - cash_for_buy)
    positions = {e.symbol: p.amount * p.last_sale_price
                  for e, p in context.perf_tracker.position_tracker.positions.items()}

```

2. 生成卖出订单: hold_days天之后才开始卖出; 对持仓的股票, 按StockRanker预测的排序末位淘汰

```

    if not is_staging and cash_for_sell > 0:
        equities = {e.symbol: e for e, p in context.perf_tracker.position_tracker.positions.items()}
        instruments = list(reversed(list(ranker_prediction.instrument[ranker_prediction.instrument.apply(
            lambda x: x in equities and not context.has_unfinished_sell_order(equities[x]))])))
        # print('rank order for sell %s' % instruments)
        for instrument in instruments:
            context.order_target(context.symbol(instrument), 0)
            cash_for_sell -= positions[instrument]
            if cash_for_sell <= 0:
                break

```

3. 生成买入订单: 按StockRanker预测的排序, 买入前面的stock_count只股票

```

    buy_cash_weights = context.stock_weights
    buy_instruments = list(ranker_prediction.instrument[:len(buy_cash_weights)])
    max_cash_per_instrument = context.portfolio.portfolio_value * context.max_cash_per_instrument
    for i, instrument in enumerate(buy_instruments):
        cash = cash_for_buy * buy_cash_weights[i]
        if cash > max_cash_per_instrument - positions.get(instrument, 0):
            # 确保股票持仓量不会超过每次股票最大的占用资金量
            cash = max_cash_per_instrument - positions.get(instrument, 0)
        if cash > 0:
            context.order_value(context.symbol(instrument), cash)

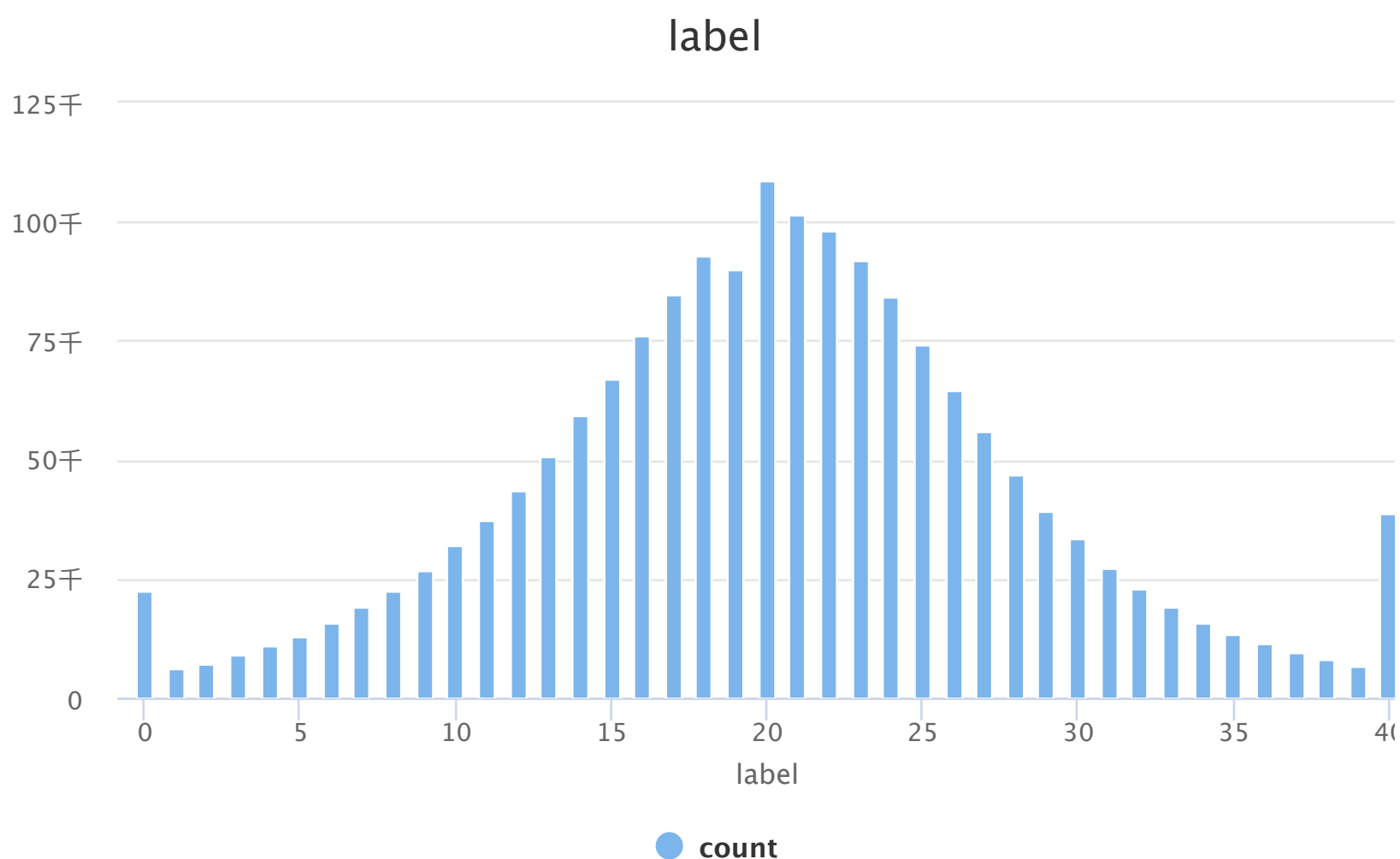
```

```
# 调用交易引擎
m6 = M.trade.v1(
    instruments=None,
    start_date=conf.split_date,
    end_date=conf.end_date,
    prepare=prepare,
    initialize=initialize,
    handle_data=handle_data,
    order_price_field_buy='open',      # 表示 开盘 时买入
    order_price_field_sell='close',    # 表示 收盘 前卖出
    capital_base=1000000,              # 初始资金
    benchmark='000300.SHA',           # 比较基准, 不影响回测结果
    # 通过 options 参数传递预测数据和参数给回测引擎
    options={'hold_days': conf.hold_days, 'model_id': m5.model_id}
)
```

[2017-07-21 10:14:52.353625] WARNING: bigquant: 此模块版本 M.fast_auto_labeler.v5 已不再维护, 并可能在未来被删除: 请更新到 fast_auto_labeler 最新版本

[2017-07-21 10:14:52.355375] INFO: bigquant: fast_auto_labeler.v5 start ..

[2017-07-21 10:14:52.359177] INFO: bigquant: hit cache

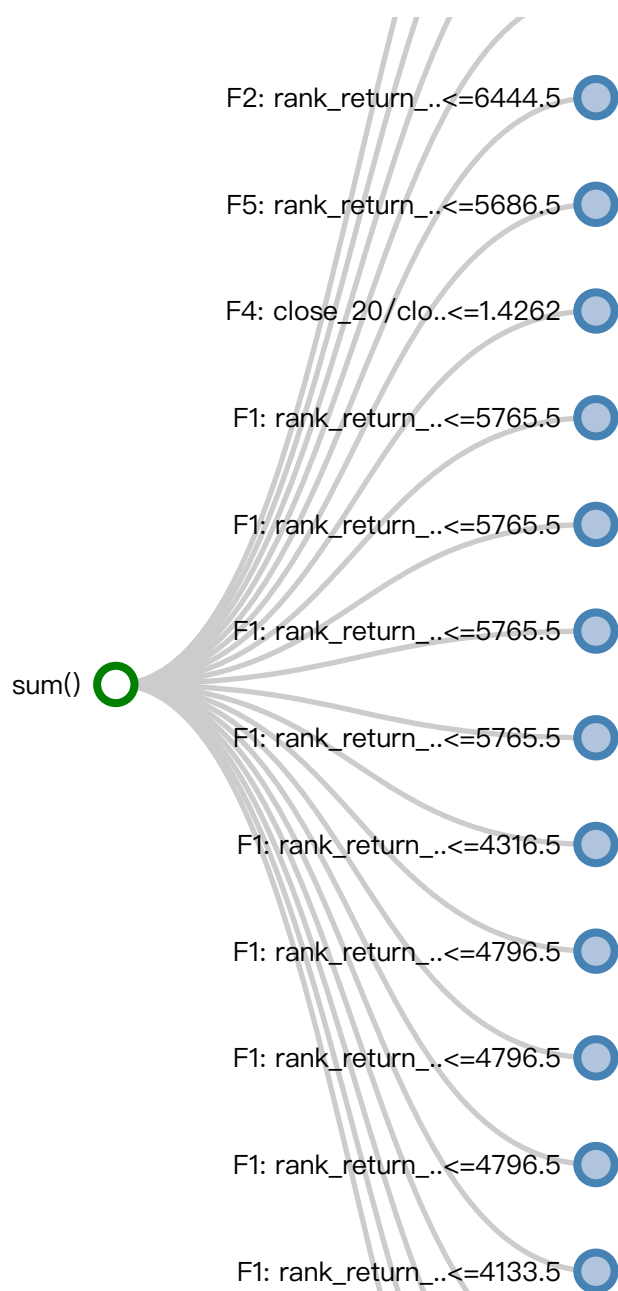


BigQuant.co

[2017-07-21 10:14:52.378027] INFO: bigquant: fast_auto_labeler.v5 end [0.022615s].

[2017-07-21 10:14:52.393046] INFO: bigquant: general_feature_extractor.v5 start ..

```
[2017-07-21 10:14:52.394930] INFO: bigquant: hit cache
[2017-07-21 10:14:52.395652] INFO: bigquant: general_feature_extractor.v
5 end [0.00261s].
[2017-07-21 10:14:52.406406] INFO: bigquant: transform.v2 start ..
[2017-07-21 10:14:52.407620] INFO: bigquant: hit cache
[2017-07-21 10:14:52.408286] INFO: bigquant: transform.v2 end [0.00188s]
.
[2017-07-21 10:14:52.420398] INFO: bigquant: join.v2 start ..
[2017-07-21 10:14:52.422900] INFO: bigquant: hit cache
[2017-07-21 10:14:52.424480] INFO: bigquant: join.v2 end [0.00412s].
[2017-07-21 10:14:52.474627] INFO: bigquant: stock_ranker_train.v3 start
..
[2017-07-21 10:15:00.285795] INFO: df2bin: prepare data: training ..
[2017-07-21 10:15:26.963041] INFO: stock_ranker_train: training 61795348
: 1745690 rows
[2017-07-21 10:17:49.550351] INFO: bigquant: stock_ranker_train.v3 end [
177.075762s].
```



```
[2017-07-21 10:17:49.615134] INFO: bigquant: backtest.v6 start ..
[2017-07-21 10:17:49.744583] INFO: bigquant: general_feature_extractor.v
5 start ..
```

```
[2017-07-21 10:18:05.100000] INFO: bigquant: backtest.v6 end [155.484866s]
```

[2017-07-21 10:18:05.199988] INFO: general_feature_extractor: year 2011, featurerows=511455

[2017-07-21 10:18:21.136626] INFO: general_feature_extractor: year 2012, featurerows=565675

[2017-07-21 10:18:29.797541] INFO: general_feature_extractor: year 2013, featurerows=564168

[2017-07-21 10:18:32.606851] INFO: general_feature_extractor: year 2014, featurerows=569948

[2017-07-21 10:18:35.490061] INFO: general_feature_extractor: year 2015, featurerows=569698

[2017-07-21 10:18:38.453846] INFO: general_feature_extractor: year 2016, featurerows=641546

[2017-07-21 10:18:40.437721] INFO: general_feature_extractor: year 2017, featurerows=388437

[2017-07-21 10:18:40.543923] INFO: general_feature_extractor: total feature rows: 3810927

[2017-07-21 10:18:40.550334] INFO: bigquant: general_feature_extractor.v5 end [50.80577s].

[2017-07-21 10:18:40.559148] INFO: bigquant: transform.v2 start ..

[2017-07-21 10:18:43.221939] INFO: transform: transformed /y_2011, 505694/511455

[2017-07-21 10:18:45.846855] INFO: transform: transformed /y_2012, 562381/565675

[2017-07-21 10:18:48.576841] INFO: transform: transformed /y_2013, 564139/564168

[2017-07-21 10:18:51.309058] INFO: transform: transformed /y_2014, 567630/569948

[2017-07-21 10:18:54.102325] INFO: transform: transformed /y_2015, 565355/569698

[2017-07-21 10:18:57.141434] INFO: transform: transformed /y_2016, 637125/641546

[2017-07-21 10:18:59.418691] INFO: transform: transformed /y_2017, 382981/388437

[2017-07-21 10:18:59.445927] INFO: transform: transformed rows: 3785305/3810927

[2017-07-21 10:18:59.461746] INFO: bigquant: transform.v2 end [18.902531s].

[2017-07-21 10:18:59.485756] INFO: bigquant: stock_ranker_predict.v2 start ..

[2017-07-21 10:19:02.191683] INFO: df2bin: prepare data: prediction ..

[2017-07-21 10:19:58.513212] INFO: stock_ranker_predict: prediction: 3785305 rows

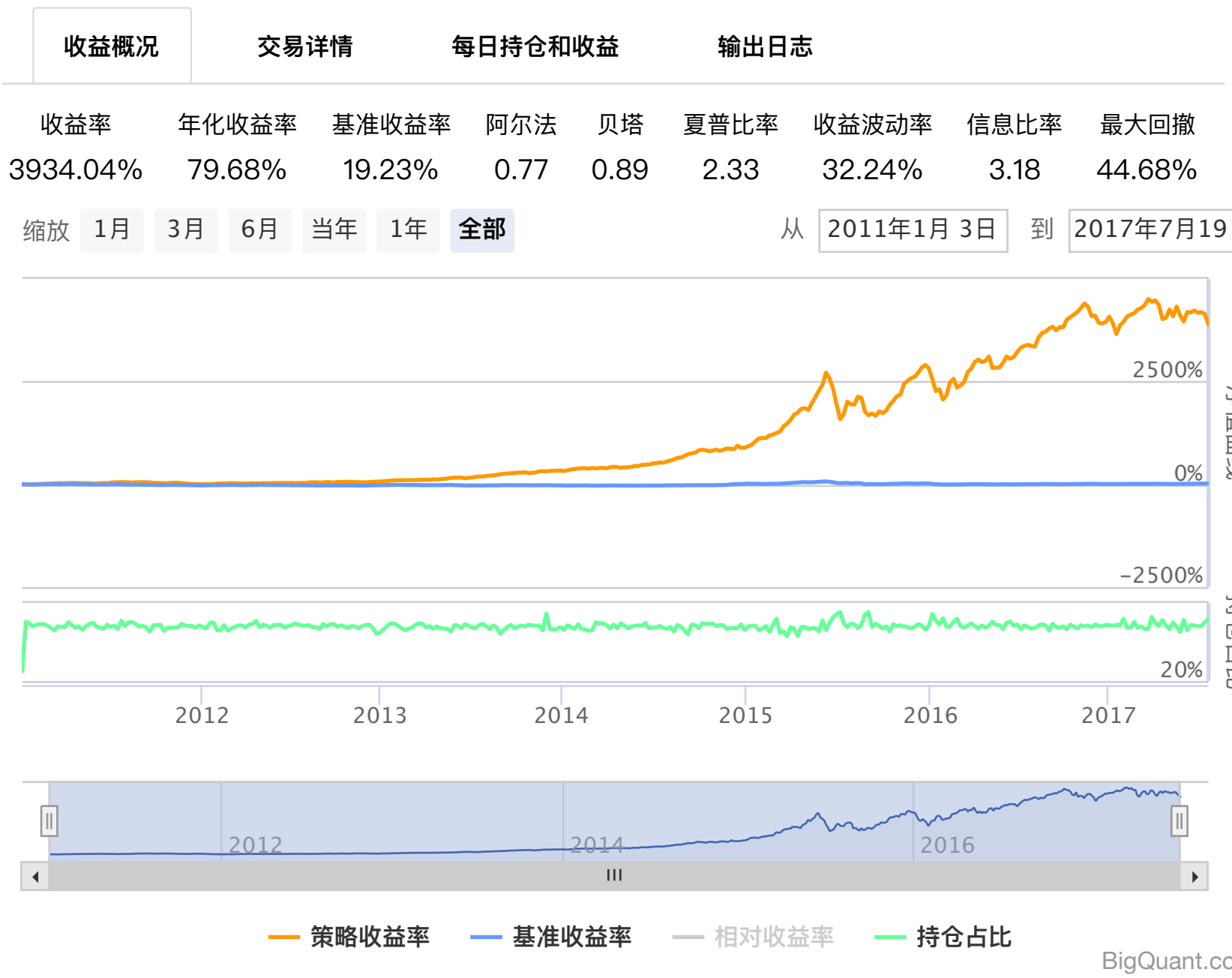
[2017-07-21 10:25:50.092518] INFO: bigquant: stock_ranker_predict.v2 end [410.606788s].

[2017-07-21 10:27:08.756082] INFO: Performance: Simulated 1590 trading days out of 1590.

[2017-07-21 10:27:08.757557] INFO: Performance: first open: 2011-01-04 14:30:00+00:00

[2017-07-21 10:27:08.758653] INFO: Performance: last close: 2017-07-19 19:00:00+00:00

[注意] 有 643 笔卖出是在多天内完成的。当日卖出股票超过了当日股票交易的2.5%会出现这种情况。



[2017-07-21 10:27:15.487526] INFO: bigquant: backtest.v6 end [565.872326 s].

<div>📌 BigQuant官方交流群</div> <div>💡 交流</div>	1	5月5日
<div>ai-自动 测试</div> <div>💡 策略分享</div>	0	1小时前
<div>[量化学堂-金融市场]Barra风险结构管理模型</div> <div>💡 barra, alpha, 多因子</div>	0	2小时前
<div>产品动态——持续更新，为你提供更好的研究体验</div> <div>💡 产品动态, 产品功能</div>	0	2小时前
<div>上次访问</div>		
<div>[量化学堂-机器学习]AI量化策略的初步理解</div> <div>💡 stockranker, 机器学习, ai</div>	0	18小时前
<div>[量化学堂-机器学习]量化投资中的特征工程</div> <div>💡 量化交易, 机器学习, 特征工程</div>	0	18小时前
<div>[量化学堂-机器学习]机器学习有哪些常用算法</div> <div>💡 机器学习, ai</div>	0	18小时前
<div>[量化学堂-机器学习]什么是机器学习</div>	0	18小时前
<div>[量化学堂-新手专区]聊一聊策略生成器</div> <div>💡 策略生成器</div>	0	18小时前
<div>[量化学堂-新手专区]BigQuant回测机制</div> <div>💡 事件驱动, 回测</div>	0	18小时前
<div>[量化学堂-新手专区]BigQuant策略平台使用帮助</div> <div>💡 bigquant, 研究平台</div>	0	18小时前
<div>BigQuant平台学习第二篇：如何开发AI策略（上）</div> <div>💡 机器学习</div>	5	18小时前
<div>摩根大通报告12个亮点总结：金融领域的机器学习工具有哪些？</div> <div>💡 机器学习</div>	0	19小时前
<div>[量化学堂-金融市场]因子风险暴露</div> <div>💡 风险归因, 风险暴露</div>	0	20小时前

<div> <div>[量化学堂-Python编程]Numpy库</div> <div> <div></div> <div>numpy</div> </div> </div>	0	20小时前
<div> <div>[量化学堂-Python编程]数据类型之字典</div> <div> <div></div> <div>字典, dictionary</div> </div> </div>	0	20小时前
<div> <div>[功能需求] 用户上传自己的数据集</div> <div> <div></div> <div>功能需求</div> </div> </div>	3	21小时前
<div> <div>[量化学堂-数学知识]深入理解协整</div> <div> <div></div> <div>协整</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]初识协整</div> <div> <div></div> <div>协整</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]多元线性回归</div> <div> <div></div> <div>线性回归, 逐步回归</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]线性回归</div> <div> <div></div> <div>线性回归</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]相关关系</div> <div> <div></div> <div>相关系数, 相关性, 相关关系</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]数据集中趋势的度量：平均值</div> <div> <div></div> <div>集中趋势, 平均值</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]数据异常值处理</div> <div> <div></div> <div>数据异常值处理</div> </div> </div>	0	1天前
<div> <div>[量化学堂-数学知识]峰度和偏度</div> <div> <div></div> <div>策略分享, 峰度, 偏度, 正态性检验</div> </div> </div>	0	1天前
<div> <div>[量化学堂-Python编程]函数调用与定义</div> <div> <div></div> <div>函数</div> </div> </div>	0	1天前
<div> <div>[量化学堂-Python编程]条件与循环：if、while、for</div> <div> <div></div> <div>python, 条件与循环</div> </div> </div>	0	1天前
<div> <div>Keras 的units到底是怎么样算出来的。</div> <div></div> </div>	3	1天前
<div> <div>[量化学堂-Python编程]数据类型之元组、集合</div> <div> <div></div> <div>tuple, set, 集合</div> </div> </div>	0	1天前

