基于价值策略的选股模型

本文实现了一个基于价值策略的选股模型，参考了文章《价值选股策略——基于机器学习算法》【1】，根据文章中的思路，完成了选股部分的代码（文章中因为利益相关的原因并没有公开），并对文章中使用模型的训练方法进行了改进。

本文采用 BigQuant（https://bigquant.com/）作为编写代码平台，该平台基于 Jupyter 进行了包装，内置了很多金融数据的访问接口，可以方便地获取并处理股票数据，编写模型，并进行回测、实测。

一、 公允价值

2017 年 8 月 J.P.Morgan 发表了文章《Value Strategies based on Machine Learning》，提出了一种通过大量股票特征来预测公司市净率的统计建模方法。该方法使用的策略是：通过预测公司市净率，然后与实际市净率进行比较来得到“公允价值”，以便发现哪些被“错误定价”的股票，识别出哪些股票被高估，哪些股票被低估，然后开发出买入低估股票，卖出高估股票的策略。

“公允价值”的计算为预测市净率数据和实际市净率之差，具体计算公式为：

$$S_{j,t} = \frac{E\left[PB_{j,t+1}\right] - PB_{j,t}}{\sigma_{j,t}}$$

选择市净率来衡量股票价值的主要原因有：

1. 市净率是公司估值的一个常用指标，并被实践证明是企业价值很好的一个度量。
2. 市净率来自于财务指标，而财务指标通常很长一段时间才发生变化，因此比较稳定，即信噪比较高，使得机器学习、深度学习策略的建模更有效。

二、 预测市净率

我们想依据股票在某一年的财务特征，来预测下一年的市净率，从而得到股票的公允价值。财务特征由股票在一年内的财务指标计算得到，通过使用 XGBoost（eXtreme Gradient Boosting）模型，来回归股票下一年的市净率。

1. 特征选择

本文使用了【1】中介绍到的 22 个股票特征：

每股收益	销售净利润	每股净资产	资产周转率	总资产报酬率	净资产报酬率	总利润/总资产	现金流/总资产
fs_eps_0	fs_net_profit_margin_ttm_0	fs_bps_0	asset_turnover	fs_roa_0	fs_roe_0	gross_profit_to_asset	cash_flow_to_assets

是否持有股利政策	前期市盈率	营业利润/价格	前期营业利润/价格	现金流/价格	销售收入/权益总额	息税前利润/权益总额	总市值
positive_earnings	pe_forward	sales_yield_forward	sales_yield_forward	cash_flow_yield	sales_to_ev	ebitda_to_ev	market_cap_0

系统因子	1 月动量	12 月动量	价格波动率	平均成交额	每股收益同化增长率
beta_szzs_90_0	month_1_mom	month_12_mom	volatity	mean_amount	fs_eps_yoy_0

提取完的数据如图所示，data 和 instrument 代表了股票代码和日期，其中 pb_lf 代表了 date 时的市净率，label 代表了该股票一年后的市净率。鉴于国内 A 股公司最迟在次年 4 月最后一天发布上一年财务报表，所以我们将 date 设为 5 月的第一个交易日。

	A	B	C	D	E	F	G	H	I	J	K	L	M
		pb_lf	label	instrument	date	fs_eps_0	fs_net_pro	fs_bps_0	asset_turn	fs_roa_0	fs_roe_0	gross_prof	cash_flow_
0		-0.77227	-0.06083	000001.SZA	2010/5/4	2.953093		1.190389			1.077986		
1		-1.03011	-0.05662	000002.SZA	2010/5/4	0.096522	13.9303	-0.08088	-0.66645	0.010049	0.185064	-0.01402	0.352279
2		2.650589	0.132181	000004.SZA	2010/5/4	-0.60717	9.5008	-1.08599	-0.70981	-0.00789	-0.43214	-0.08478	-0.35458
3		0.407392	-0.00665	000005.SZA	2010/5/4	-0.65594	-136.503	-1.07439	-1.24976	-0.06268	-0.60287	-0.35603	0.28393
4		-1.0342	-0.05191	000006.SZA	2010/5/4	-0.07766	16.0079	0.157709	-0.81148	-0.00509	-0.04623	-0.11514	0.230511
5		2.650589	0.094465	000008.SZA	2010/5/4	-0.60578	3.7551	-0.99081	-1.06246	-0.05421	-0.42199	-0.25792	0.293082
6		0.172744	0.021483	000009.SZA	2010/5/4	-0.07836	12.822	-0.60434	-0.60852	0.067809	0.357677	0.148905	-0.6712
7		0.717605	-0.01498	000011.SZA	2010/5/4	-0.56676	10.1964	-0.94804	-0.768	-0.04055	-0.31976	-0.212	-0.27631
8		-0.05264	-0.00278	000012.SZA	2010/5/4	1.071937	19.7414	0.293026	-0.25489	0.143758	0.689013	0.520641	0.154592
9		-0.36792	-0.03297	000014.SZA	2010/5/4	1.764481	15.7945	-0.40492	-0.05656	0.244707	2.385158	1.042957	0.293267
10		-1.16391	-0.06498	000016.SZA	2010/5/4	-0.39397	1.0099	-0.1877	0.800054	-0.02208	-0.22246	-0.16413	0.303575

2. XGBoost 模型

XGBoost（eXtreme Gradient Boosting）的本质即多个回归树的集成，它的核心思想是：建立 K 个回归树，使得树群的预测值尽量接近真实值（准确率）而且有尽量大的泛化能力。

其中，回归树是一个应用于回归问题的决策树模型，一个回归树对应着输入空间（即特征空间）的一个划分以及在划分单元上的输出值。分类树中，我们采用信息论中的方法，通过计算选择最佳划分点。而在回归树中，采用的是启发式的方法。假如我们有n个特征，每个特征有$s_i(i \in (1,n))$个取值，那我们遍历所有特征，尝试该特征所有取值，对空间进行划分，直到取到特征j的取值s，使得损失函数最小，这样就得到了一个划分点。

建立 XGBoost 模型的代码如下：

```
# xgboost 构建对象
dtrain = xgb.DMatrix(X_train.values,label=y_train.values) # 这个地方如果是 X_train 其实不影响结果
dtest = xgb.DMatrix(X_test.values,label=y_test.values)
# 设置参数，参数的格式用 map 的形式存储
param = {'max_depth': 3,                # 树的最大深度
        'eta': 0.1,                     # 一个防止过拟合的参数，默认 0.3
```

```
    'n_estimators':100,              # Number of boosted trees to fit
    'silent': 1,                     # 打印信息的繁简指标，1 表示简， 0 表示繁
    'objective': 'reg:linear'} # 使用的模型，分类的数目
num_round = 50 # 迭代的次数
```

3. XGBoost 模型训练的优化

文章【1】中的 XGBoost 训练代码存在问题：即没有将训练数据打乱，训练数据按日期顺序和股票号顺序排列，导致训练过程容易造成过拟合。如下所示：

```
[0] eval-rmse:1.40968   train-rmse:0.79868
[1] eval-rmse:1.40015   train-rmse:0.746451
[2] eval-rmse:1.39564   train-rmse:0.698628
[3] eval-rmse:1.3953    train-rmse:0.65477
[4] eval-rmse:1.3984    train-rmse:0.614491
[5] eval-rmse:1.40428   train-rmse:0.577441
[6] eval-rmse:1.41239   train-rmse:0.54331
[7] eval-rmse:1.42226   train-rmse:0.511824
[8] eval-rmse:1.43348   train-rmse:0.482735
[9] eval-rmse:1.44571   train-rmse:0.455814
[10]    eval-rmse:1.45867   train-rmse:0.430869
...
[97]    eval-rmse:1.49248   train-rmse:0.066496
[98]    eval-rmse:1.49228   train-rmse:0.06525
[99]    eval-rmse:1.49228   train-rmse:0.065089
```

虽然训练集的 loss 值在下降，但是验证集上的 loss 值几乎没有变化，这是很明显的过拟合现象。

我使用了如下代码将训练数据打乱，**让不同年份的股票数据参杂在一起**，使得训练过程避免过拟合：

```
data = data.sample(frac=1)
```

加入了该行代码之后，训练过程会变得平稳，验证集上的损失值显著降低，如下所示：

```
[0] eval-rmse:0.544987  train-rmse:0.967636
[1] eval-rmse:0.504367  train-rmse:0.928216
[2] eval-rmse:0.47058   train-rmse:0.886903
[3] eval-rmse:0.439664  train-rmse:0.855488
[4] eval-rmse:0.414679  train-rmse:0.820882
[5] eval-rmse:0.391104  train-rmse:0.795777
[6] eval-rmse:0.371715  train-rmse:0.773692
[7] eval-rmse:0.356588  train-rmse:0.745805
```

```
[8] eval-rmse:0.342439  train-rmse:0.727368
...
[47]    eval-rmse:0.276735  train-rmse:0.316814
[48]    eval-rmse:0.276747  train-rmse:0.312425
[49]    eval-rmse:0.276965  train-rmse:0.30456
```

由于训练数据较少，所以建议多次运行训练代码，取验证集上损失最低的那次作为最终的模型参数。

三、 选股过程

文章【1】的作者与机构有一些合作，只放出了数据和算法，我根据文章中的思路大致实现了选股代码。

首先根据 XGBoost 模型的预测市净率，计算每只股票的公允价值：

```
out_of_sample_data['diff_value'] = out_of_sample_data['predict_pb_lf'] - out_of_sample_data['pb_lf']
```

然后，在每个交易日，根据公允价值的大小进行降序排序，买入前 100 支股票：

```
selected_data = selected_data.groupby('date').apply(lambda df: df.sort_values('diff_value', ascending=False)[:stock_num])
```

资金分配采用平均买入的方式：

```
# 等量分配资金买入股票
weight = 1.0 / len(instruments_to_buy)
for instrument in instruments_to_buy:
    if data.can_trade(context.symbol(instrument)):
        context.order_target_percent(context.symbol(instrument), weight)
```

使用平台提供的接口进行回测：

```
m = M.trade.v3(
    instruments=instrument,
    start_date=start_date,
    end_date=end_date,
    initialize=initialize,
    handle_data=handle_data,
    # 买入订单以开盘价成交
    order_price_field_buy='open',
    # 卖出订单以开盘价成交
    order_price_field_sell='open',
    capital_base=capital_base,
    benchmark=benchmark,
    options={'selected_data': selected_data, 'rebalance_period': rebalance_period},
```

```
    m_cached=False
)
```

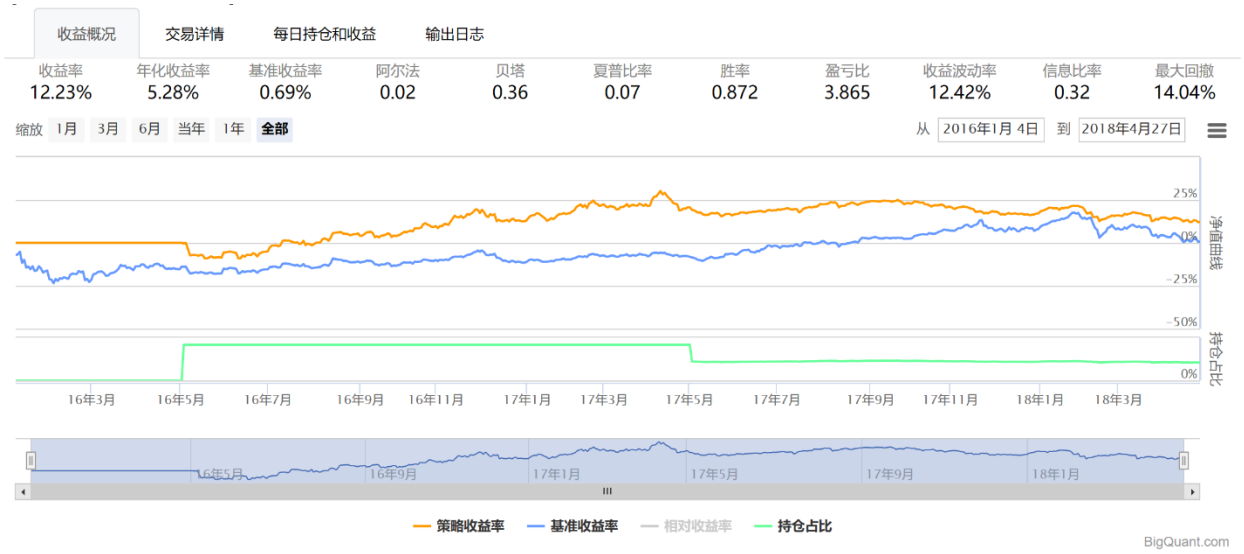
本文使用 2010-2015 年的股票数据作为训练数据，在 2016 年 1 月-2018 年 5 月进行股票的回测。在进行回测时，调仓周期是一年，在年报数据公布后的第一个交易日进行调仓。

四、 实验与结果

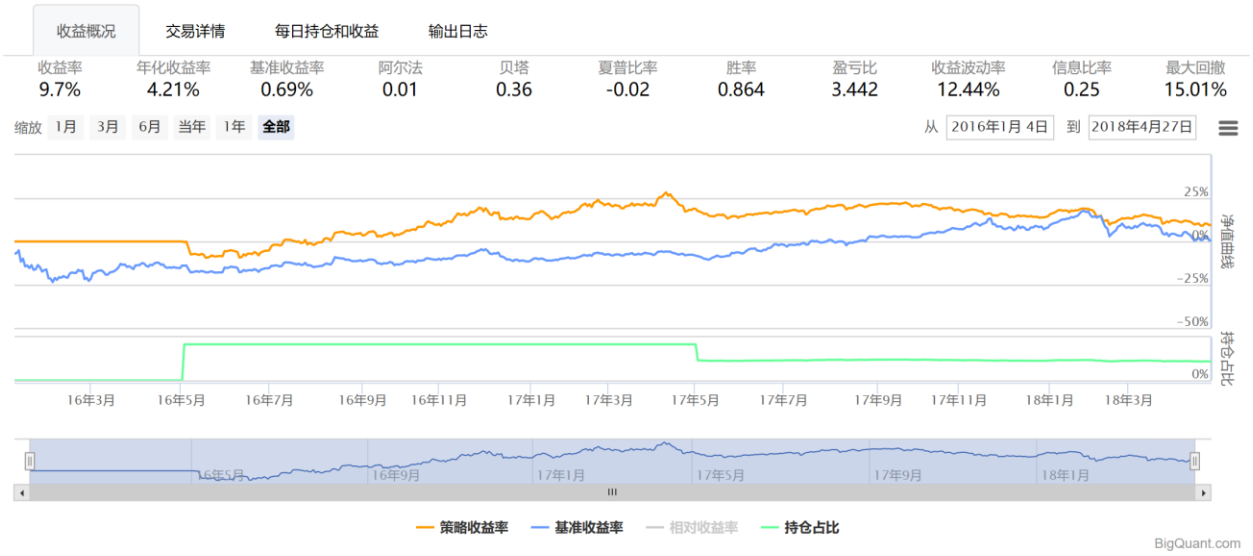
1. 模型回测结果

考虑到 XGBoost 是一种参数随机优化模型，所以我们运行多次模型得到实验结果（下图是 3 次实验的结果），从 2015 年 5 月到 2017 年底，本文的策略可以获得约 10%左右的年化收益率。到 2018 年 4 月底，收益率跌到 4%-5%左右，这可能跟 A 股的整体行情有关系。

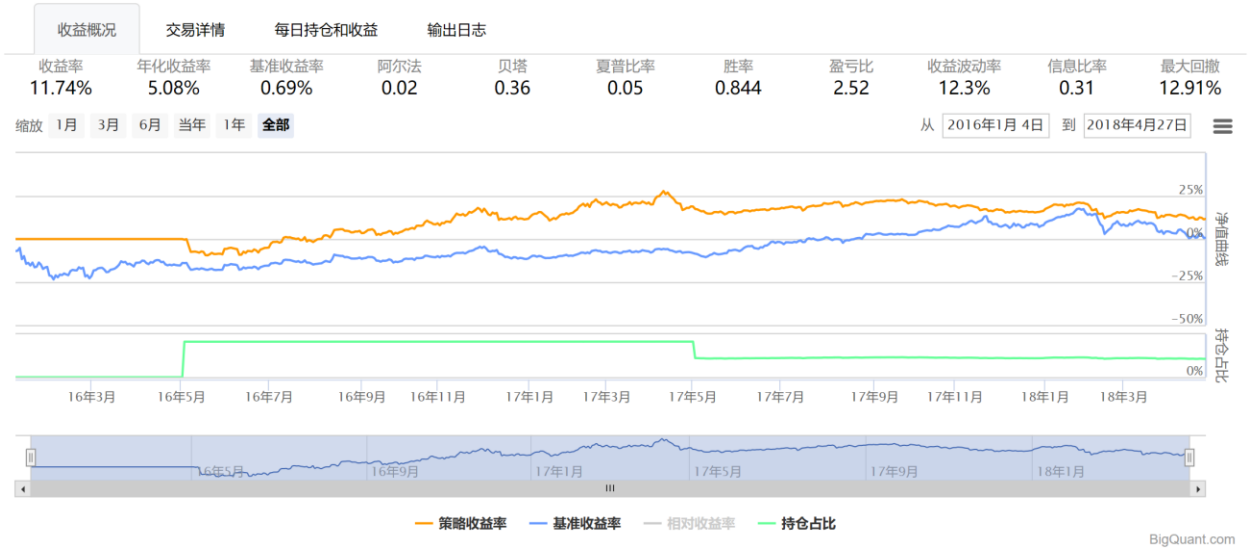
实验结果 1:



实验结果 2:



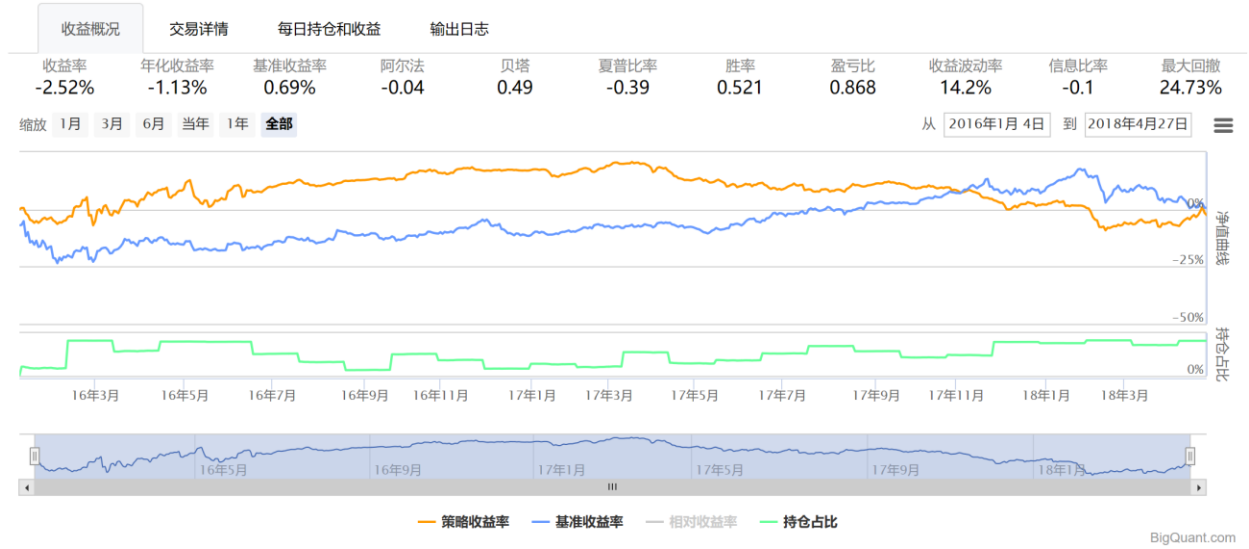
实验结果 3:



2. 模型对比

小市值策略是 A 股前些年较为流行的一种策略，也是一种基于价值的策略。在本实验中，调仓周期设为 20 天，每次买入市值最低的 30 支股票。

该策略在 2016 年-2018 年 5 月这段时间内的表现如图所示:



通过对比可以发现，本文基于机器学习的策略要优于传统的小市值策略，从 17 年末至今，本文的策略的收益率跌幅要明显小于小市值策略，所以机器学习模型从数据中学习的能力得到了体现。

五、 总结与未来改进

本文基于文章【1】的思路实现了一个完整的选股模型，并优化了其中 XGBoost 模型训练的训练过程。目前可改进的方面有： （1）选取更多有意义的特征，使得市净率的预测更加准确（2）根据公允价值的排序，在买入股票时分配不同的百分比，这样可能有利于收益的最大化。

（3）在加入更多的特征之后，考虑使用深度神经网络来作为回归模型，利用其强大的特征抽象能力。

六、 参考文献

【1】 价值选股策略——基于机器学习算法

【2】XGBoost 的维基百科

【3】http://xgboost.readthedocs.io/en/latest/tutorials/index.html

【4】分类回归树：https://blog.csdn.net/zhihua_oba/article/details/72230427

【5】回归树：https://blog.csdn.net/weixin_40604987/article/details/79296427