高频策略研究报告——基于orderbook指标的高频收益率预测

一、orderbook数据撮合

详细代码见脚本`get_orderbook.py`以及`readme`，订单簿输出结果见文件`res/xxxxxx_20200110.txt`

撮合的逻辑：

每个订单信息都储存在OrderedDict里面（买卖分开存），实现O(1)复杂度的存取、删除；

下单信号到达时，向OrderedDict新增元素；

成交信号到达时，向OrderedDict删除元素（both 买卖）；

撤单信号到达时，向OrderedDict删除元素；

在指定时间点撮合时，先对OrderedDict依照价格优先级进行排序，并取前10档的买卖价格汇总成List，复杂度`O(nlogn)`。

其实在最开始有考虑使用顺序存储的方式动态维护一个有序的oederbook，但代价是每个下单信息都要花`O(logn)`进行二分查找并插入`O(n)`，总体复杂度`O(nlogn+n^2)`；但其实我们只是在指定时间截面上关注order的顺序，因而当输出orderbook的频率 f 远小于订单量 n 时，可以只用OrderedDict维护每个tick的orderbook，当需要输出orderbook再排序，复杂度`O(f*nlogn)`

下图是处理结果：000069_20200110

```
order book at: 20200110093000030
sell list:
(78800, 1370000)
(78700, 6470000)
(78600, 2900000)
(78500, 6900000)
(78400, 7400000)
(78300, 10130000)
(78200, 9760000)
(78100, 610000)
(78000, 10441300)
(77900, 12820000)
------------------------------------------------------------------
buy list:
(77800, 11630000)
(77700, 24920000)
(77600, 150000)
(77500, 2370000)
(77400, 29370000)
(77300, 9550000)
(77200, 2940000)
(77100, 5200000)
(77000, 2760000)
(76900, 12910000)
```

效率较为理想：

```
100%|██████████| 48214/48214 [00:02<00:00, 17889.36it/s]
100%|██████████| 13644/13644 [00:00<00:00, 17803.72it/s]
100%|██████████| 101253/101253 [00:05<00:00, 17789.69it/s]
100%|██████████| 73411/73411 [00:04<00:00, 18090.98it/s]
100%|██████████| 30819/30819 [00:01<00:00, 17932.70it/s]
100%|██████████| 9683/9683 [00:00<00:00, 18019.07it/s]
```

二、指标选择

除了常见的spread、depth等指标，重点介绍本研究引入的买卖失衡类指标：

①订单不平衡（Order Imbalance）：

订单不平衡指标反映了市场供需关系和价格买卖压力，侧面反映了市场结构、市场偏好和投资者结构等问题。目前关于订单不平衡是构建基于买卖压力失衡策略的重要指标，对其的研究普遍有如下观点：

1. 正负体现了供需关系，指标小于零意味着供给大于需求；
2. 聚变意味着市场供需出现较大缺口，股票的换手率加大；
3. 均值回复，指标的波动性较大，即订单流不平衡的方向改变较快、较频繁，随着统计时间间隔的跨度增大，订单流不平衡的波动性加大；
4. 存在波动性"聚集"现象。
5. 自相关性/长记忆性，"聚集"，相邻的符号相同，造成的原因可能是1.流动性交易者"分拆订单"的交易行为，避免信息释放速度较快和减小冲击成本；2.信息的连续性；反应了订单流不平衡的收敛速度。

根据Shen（2015）构建的交易量订单流不平衡（Volume Order Imbalance）指标（下称VOI）的方法，订单流不平衡测量了在研究的特定时间内，最优买卖价格上委托量的量之差，反映了投资行为在最优买卖价格上的供需情况，其定义为：

$$OI_t = \delta V_t^B - \delta V_t^A$$

$$\delta V_t^B = \begin{cases} 0, & P_t^B < P_{t-1}^B \\ V_t^B - V_{t-1}^B, & P_t^B = P_{t-1}^B \\ V_t^B, & P_t^B > P_{t-1}^B \end{cases}$$

$$\delta V_t^A = \begin{cases} 0, & P_t^A > P_{t-1}^A \\ V_t^A - V_{t-1}^A, & P_t^A = P_{t-1}^A \\ V_t^A, & P_t^A < P_{t-1}^A \end{cases}$$

VtB和VtA分别是在t时刻的买入和卖出交易量，PtB和PtA分别是最优买卖报价。δVtB表示买入订单的委托增量。

假设当前的买入报价低于上一期的买入价格，意味着投资者撤单或者订单在的价格上成交了，因此设定δVtB=0；

假设当前的买入价格和上一期相同，用委托量的增量作为δVtB；假设当前的买入报价大于上一期买入报价，意味着投资者愿意在更高的价格上买入，价格存在上行的趋势。

同理δVtA表示卖出订单的委托量增量。将买入订单委托量和卖出订单委托量增量之差作为订单流不平衡指标。

②深度不平衡（Depth/Length imbalance）

$$QR_j = \frac{Q_j^{bid} - Q_j^{ask}}{Q_j^{bid} + Q_j^{ask}}, j = 1,\ldots,10$$

其中Qjask表示第j档的委托卖量，Qjbid表示第j档的委托买量

③宽度不平衡（height Imbalance）

$$HR_j = \frac{(P_j^{bid} - P_{j-1}^{bid}) - (P_j^{ask} - P_{j-1}^{ask})}{(P_j^{bid} - P_{j-1}^{bid}) + (P_j^{ask} - P_{j-1}^{ask})}, \quad j = 2,\ldots,10$$

其中Pjask表示第j档的委托卖价，其中Pjbid表示第j档的委托买价。

④买卖压力指标（Press）：

根据天风证券的《买卖压力失衡-利用高频数据拓展盘口数据》，根据挂单价格与中间价格的距离赋予不同档位的委托订单不同的权重，以此对成交量进行加权并构建买卖压力指标，定义挂单i的权重为

$$w_i = \frac{M/(p_i - M)}{\sum_i^n M/(p_i - M)}$$

其中M为中间报价，因此结合各档位的委托成交量，可得各个股票的买卖方压力分别为：

$$press_{buy} = \sum_i^n vol_i \frac{M/(M - p_i)}{\sum_i^n M/(M - p_i)}$$

$$press_{sell} = \sum_i^n vol_i \frac{M/(p_i - M)}{\sum_i^n M/(p_i - M)}$$

使用买卖压力的对数比作为订单不平衡指标：

$$P = log(press_{buy}) - log(press_{sell})$$

⑤价差和价格变量：

加权价格（Weighted Price）

$$WP^{n_1 - n_2} = \frac{\sum_{j=n_1}^{n_2} (Q_j^{ask} P_j^{ask} + Q_j^{bid} P_j^{bid})}{\sum_{j=n_1}^{n_2} (Q_j^{ask} + Q_j^{bid})}$$

中间价格（Mid Price）

$$MID = \frac{P_1^{ask} + P_1^{bid}}{2}$$

绝对价差（Spread）

$$Spread = P^{ask} - P^{bid}$$

相对价差（Relative Spread）

$$RelativeSpread = \frac{(P^{ask} - P^{bid})}{(P^{ask} + P^{bid})/2}$$

其中Qjask表示第j档的委托卖量，Qjbid表示第j档的委托买量，Pjask表示第j档的委托卖价，其中Pjbid表示第j档的委托买价

三、模型概述

LightGBM是微软开发的boosting集成模型，和XGBoost一样是对GBDT的优化和高效实现，原理有一些相似之处，但它很多方面比XGBoost有着更为优秀的表现。

XGBoost通过对loss二阶泰勒展开，并优化了树模型的生成方法，相比传统GBDT具有更强的拟合能力，而且可以进行更加灵活的目标函数定义以及正则化处理，但处理大规模数据仍然存在搜索空间巨大，在节点分裂过程中仍需要遍历数据集。预排序过程的空间复杂度过高，不仅需要存储特征值，还需要存储特征对应样本的梯度统计值的索引，相当于消耗了两倍的内存。

LightGBM 为了解决这些问题提出了以下几点解决方案：单边梯度抽样算法；直方图算法；互斥特征捆绑算法；基于最大深度的 Leaf-wise 的垂直生长算法；类别特征最优分割；特征并行和数据并行；缓存优化。

本研究用到的工具包LightGBM 可以应用多种树的boosting方式，其中包含gbdt与dart等。在实验当中最终选择dart方式，相比传统不带drop的gbdt相比泛化能力更强，具体可参照Dropouts meet Multiple Additive Regression Trees，简单来说就是利用了深度神经网络中dropout设置的技巧，随机丢弃生成的决策树，然后再从剩下的决策树集中迭代优化提升树，算法原理如下：

Algorithm 1 The DART algorithm

Let N be the total number of trees to be added to the ensemble
$S_1 \leftarrow \{x, -L'_x(0)\}$
T_1 be a tree trained on the dataset S_1
$M \leftarrow \{T_1\}$
for $t = 2, \ldots, N$ **do**
  $D \leftarrow$ the subset of M such that $T \in M$ is in D with probability p_{drop}
  if $D = \emptyset$ **then** $D \leftarrow$ a random element from M
  end if
  $\hat{M} \leftarrow M \setminus D$
  $S_t \leftarrow \left\{x, -L'_x\left(\hat{M}(x)\right)\right\}$
  T_t be a tree trained on the dataset S_t
  $M \leftarrow M \cup \left\{\frac{T_t}{|D|+1}\right\}$
  for $T \in D$ **do**
    Multiply T in M by a factor of $\frac{|D|}{|D|+1}$
  end for
end for
Output M

LightGBM 本质上还是在GBDT框架下的优化算法，因而在下文中统一用**GBDT**代指这里提到的**LightGBM-dart**

四、实验

4.1 个股1s频率收益率预测

基于以上分析，构建基本订单簿的指标库。在本研究中，一共构建了两种指标库，一种是包含10档量价基本信息的基础指标库（如价差，买卖的量价以及对数差），另一种是包含10档量价基本信息，以及若干合成指标的指标库（参考：二、指标选择）。首先对基本指标库进行研究。

4.1.1 指标相关性研究

基本指标库共包含86个指标：

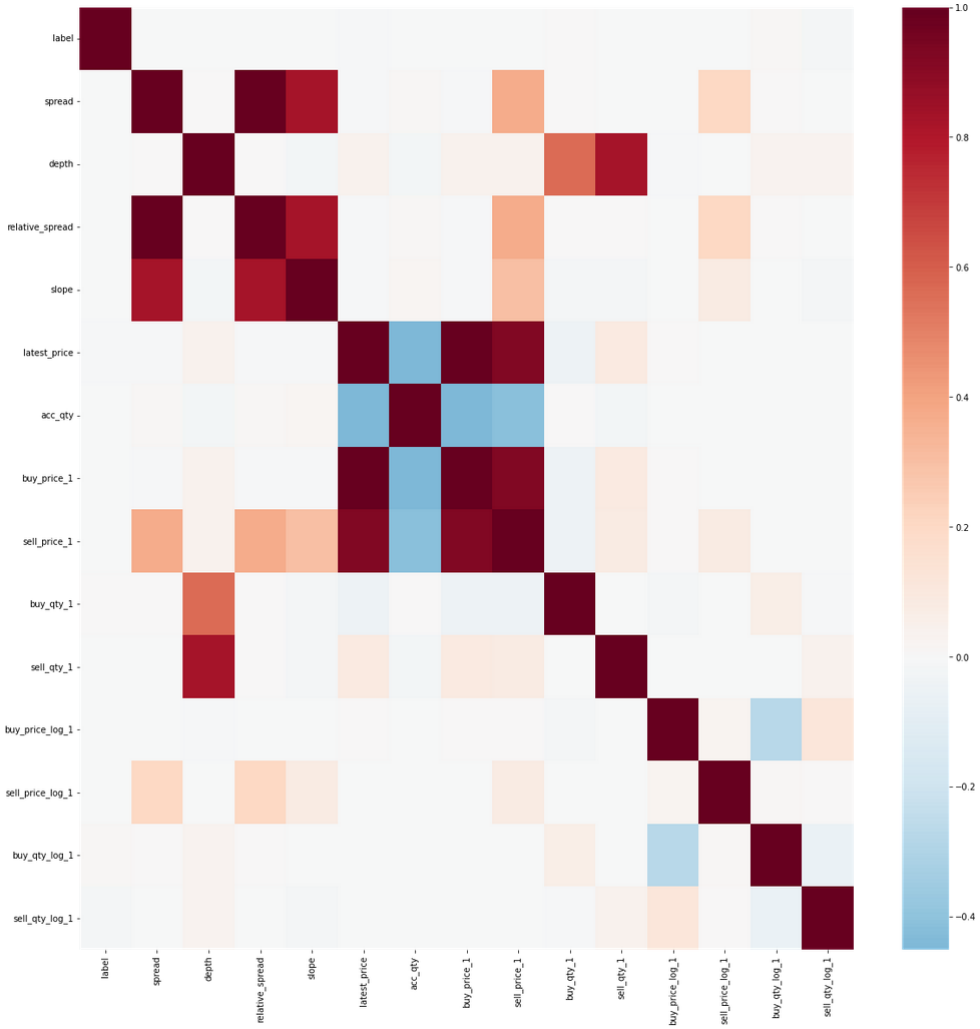
指标名称	指标变量
价差	spread
深度	depth
相对价差	relative_spread
斜率	slope
最新价	latest_price
累计收益率	acc_qty
i档买价	buy_price_i （i=1…10）
i档卖价	sell_price_i（i=1…10）
i档买量	buy_qty_i（i=1…10）
i档卖量	sell_qty_i（i=1…10）
i档买价对数差	buy_price_log_i（i=1…10）
i档卖价对数差	sell_price_log_i（i=1…10）
i档买量对数差	buy_qty_log_i（i=1…10）
i档卖量对数差	sell_qty_log_i（i=1…10）

指标预处理：

1. 数据去极值: 中位数去极值
2. 标准化：z-score

协方差矩阵：

不同类型指标相关系数矩阵



除了买卖一档价格和最新价之间的相关性较强，整体来看不同类型的变量彼此之间独立性较强，因而可以认为指标库有较好的性质，可以考虑不进行特征的降维处理（实际在实验过程中也有尝试用PCA进行降维处理，降至8、16、32维后的特征，无论用哪种模型处理，表现都不如原始特征，但可以利用PCA的正交化方法对特征进行预处理，关于PCA的引入在后面会提到）

4.1.2 baseline

首先对样本构建**线性模型**作为Baseline

样本设置：

个股：000069

频率：1s，构建1s的next return作为label

时间：训练集 20200110-20200310，验证集 20200311-20200316

划分比例 9:1

普通最小二乘法指标结果:

	mae	mse	r2
train	3.9482	30.4500	0.16830
test	4.6845	54.7598	0.01952

考虑到线性回归结果在训练集和验证集r方差距很大，有严重的过拟合问题，应该考虑使用正则化方法；另一方面，并不能完全排除变量之间可能存在多重共线性问题，因而改用岭回归做近一步研究

岭回归结果（经grid search最优罚系数为`linear_model.Ridge(alpha=0.5)`）：

	mae	mse	r2
train	3.9385	30.4930	0.16713
test	4.5798	54.7331	0.06857

使用岭回归后的结果有了很大改善，测试集r方有了明显增加。但是关于模型的解释能力，如评价各个指标的重要度，岭回归的解释能力有所不足。

对于特征重要性岭回归的解释方法是拟合出一个LinearRegression模型，并检索*coeff_*属性，该属性包含为每个输入变量（特征）找到的系数。这些系数可以为粗略特征重要性评分提供依据。该模型假设输入变量具有相同的比例或者在拟合模型之前已被按比例缩放。但由于岭回归本身存在L2正则项，倾向于筛选稀疏的特征，因而不同特征的系数差异相差悬殊，大部分特征的重要度趋于0，因而这种模型的解释能力存在局限。

一方面出于更好的模型解释能力，另一方面出于模型本身的泛化能力，本研究在后面使用GBDT的拟合方法，使得各个变量的重要性以及相互作用关系有了更好的解释，另外通过PCA预处理特征，提升了模型的拟合能力。

4.1.3 GBDT预测及分析

样本设置：

个股：000069

频率：1s

时间：训练集 20200110-20200310（36天 510737 样本），验证集 20200311-20200316 （4天 49703 样本）

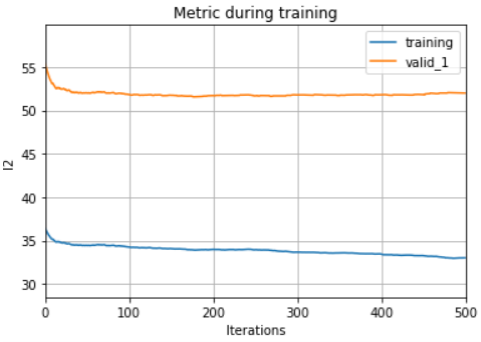
划分比例 9:1

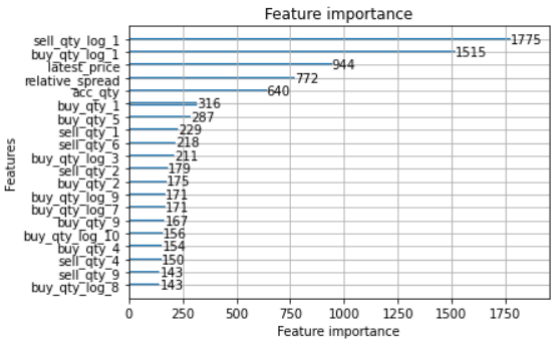
训练设置：

```
params = {
    'task': 'train',
    'boosting_type': 'dart',  # 设置提升类型 dart带dropout的gbdt
    'objective': 'regression', # 目标函数
    'metric': {'l2'},  # 评估函数
    'num_leaves': 31,   # 叶子节点数
    'min_data_in_leaf': 200, # 叶子节点最小样本数
    'max_depth':5,
    'learning_rate':  0.05,  # 学习速率
    'feature_fraction': 0.9, # 建树的特征选择比例
    'bagging_fraction': 0.8, # 建树的样本采样比例
    "random_seed":2,
    'num_boost_round':500 # 迭代（BOOST）次数
}
```

训练结果：

训练集与测试集在500次迭代中的L2_loss曲线，以及特征重要性：





在500次迭代后训练集误差持续下降，但验证集误差已无法进一步下降，某种程度上模型的预测能力已达极限，在多次试验后选取0.05的学习率，并评价其训练结果

指标结果:

	mae	mse	r2
train	2.9727	33.0227	0.15463
test	4.5749	51.9946	0.09803

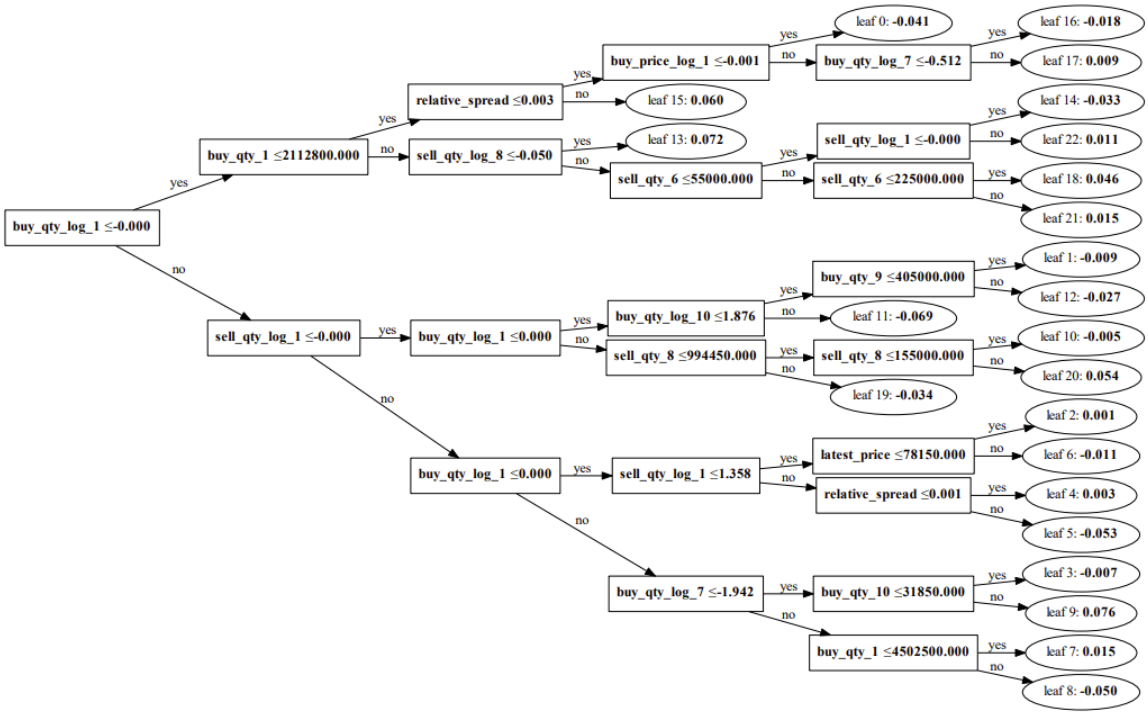
指标结果（与线性回归对比）：

test metrics	mae	mse	r2
OLS	4.6845	54.7598	0.01952
Ridge	4.5798	54.7331	0.06857
GBDT	**4.5749**	**51.9946**	**0.09803**

利用基本指标库，GBDT相较于线性模型提升明显。

模型可视化：

通过构建树模型，GBDT能直观显示出各个指标在预测当中的相互关系以及重要性，可以根据需要进一步对已有指标进行筛选或改进。



对于基础指标库，可能是因为特征并不全面，因此GBDT这类模型可以学到更复杂的pattern进行推断，测试集表现比线性模型改善较多，具有更强的泛化能力。

4.2 个股1s频率收益率预测（加入更多指标）

以上的分析都是基于基本订单簿指标，以及一些简单的合成指标，预测结果比较一般，因而考虑更多合成指标（参考：二、指标选择），以000069的1s订单簿指标为例进行实验

指标使用：

指标名称	指标变量
价差	spread
深度	depth
相对价差	relative_spread
斜率	slope
最新价	latest_price
累计收益率	acc_qty
交易量订单流不平衡	VOI
中间价	MID
买卖压力	PRESS
2-4档加权价格	WP_2_4
5-10档加权价格	WP_5_10
i档买价	buy_price_i （i=1...10）
i档卖价	sell_price_i（i=1...10）
i档买量	buy_qty_i（i=1...10）
i档卖量	sell_qty_i（i=1...10）
i档买价对数差	buy_price_log_i（i=1...10）
i档卖价对数差	sell_price_log_i（i=1...10）
i档买量对数差	buy_qty_log_i（i=1...10）
i档卖量对数差	sell_qty_log_i（i=1...10）
深度不平衡	QR_i（i=1...10）
宽度不平衡	HR_i（i=2...10）

样本设置：

个股：000069

频率：1s

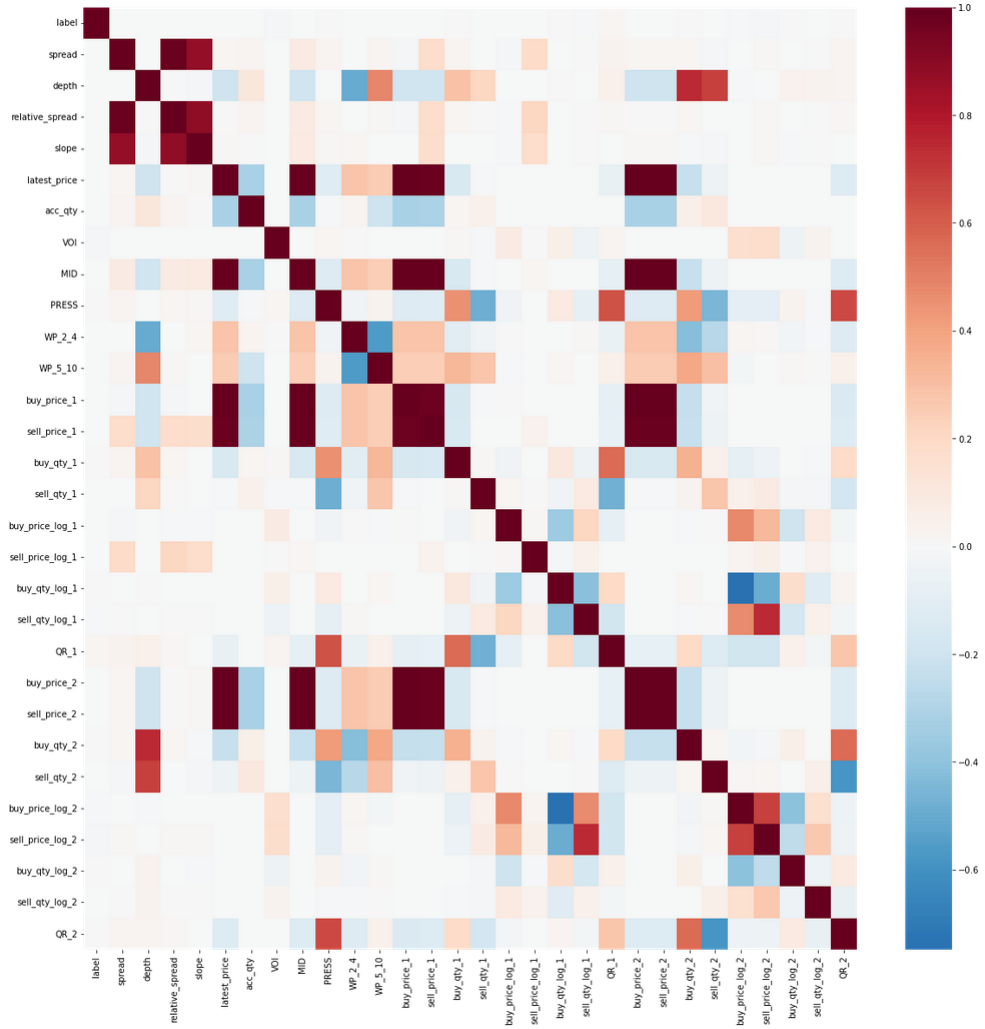
时间：训练集 20200110-20200310（36天 510737 样本），验证集 20200311-20200316 （4天 49703 样本）

划分比例 9:1

特征工程：

1. 数据去极值 中位数去极值
2. 标准化 z-score
3. PCA：分别在模型有无PCA的方法下进行对比

相关系数矩阵：



4.2.1 网格搜索

可以使用SKLearn中的超参数调优方法来进行模型调优，给出候选参数列表字典，通过`GridSearchCV`进行交叉验证实验评估，选出LightGBM在候选参数中最优的超参数

```
estimator = lgb.LGBMRegressor(**params)
param_grid = {
    'num_leaves': [31,63,127],
    'min_data_in_leaf': [200,500,2000,5000],
    'max_depth':[5,6,10],
}
gbm = GridSearchCV(estimator, param_grid)
gbm.fit(X_train, y_train,
        eval_set=[(X_test, y_test)],
        eval_metric='l2')

print('best params :')
print(gbm.best_params_)
```

依据上述所得最优参数进行模型训练，最优参数设置见下

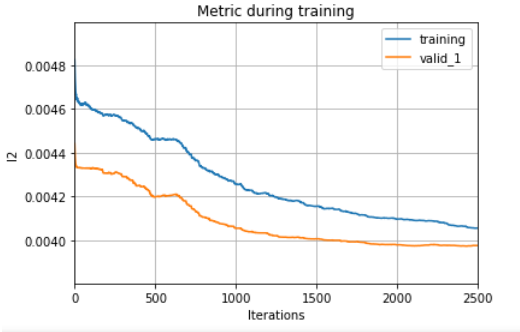
4.2.2 训练过程以及结果分析

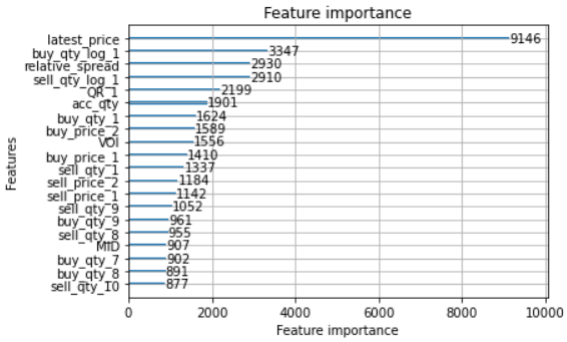
训练设置：

```
params = {
    'task': 'train',
    'boosting_type': 'dart',  # 设置提升类型 dart带dropout的gbdt
    'objective': 'regression', # 目标函数
    'metric': {'l2'},  # 评估函数
    'num_leaves': 63,   # 叶子节点数
    'min_data_in_leaf': 2000, # 叶子节点最小样本数
    'max_depth':6,
    'learning_rate':  0.1,  # 学习速率
    'feature_fraction': 0.9, # 建树的特征选择比例
    'bagging_fraction': 0.8, # 建树的样本采样比例
    "random_seed":2,
    'num_boost_round':2500 # 迭代（BOOST）次数
}
```

训练结果：

训练集与测试集在2500次迭代中的L2_loss曲线，以及特征重要性（without PCA）：





指标结果：

GBDT without PCA	mae	mse	r2
train	4.1052	40.5521	0.16702
test	4.0472	39.7515	0.11154

相对于基本指标库，r2提升明显。另外在所有指标当中，latest_price，relative_spread，以及各种一档量价的对数差指标有较重权重，对模型解释能力贡献较大。

对特征进行PCA处理后再进行训练：

经过实验发现用了带PCA处理的数据更容易收敛，因而酌情调小学习率和迭代次数，其他参数设置如下：

```
params = {
    'task': 'train',
    'boosting_type': 'dart',  # 设置提升类型
    'objective': 'regression', # 目标函数
    'metric': {'l2'},  # 评估函数
    'num_leaves': 63,   # 叶子节点数
```

```
    'min_data_in_leaf': 2000,
    'max_depth':6,
    'learning_rate': 0.01,
    'verbose': 10,
    "random_seed":1,
    'lambda_l2':0.6,
    'num_boost_round':700
}
```

GBDT with PCA	mae	mse	r2
train	4.2828	39.8011	0.18244
test	4.2653	38.1074	0.14828

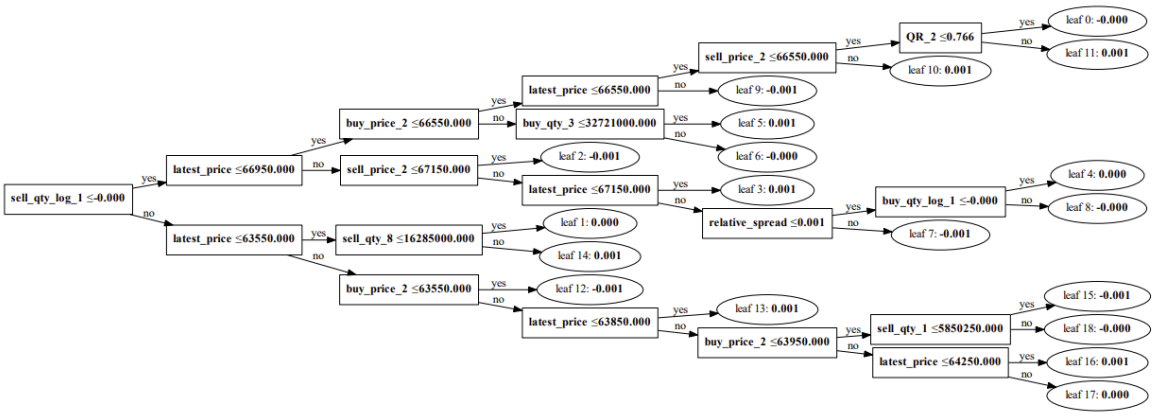
与线性模型进行对比：

test metrics	mae	mse	r2
OLS	4.4556	38.2601	0.14487
Ridge (alpha=0.5)	4.4409	38.2432	0.14524
GBDT without PCA	4.0472	39.7515	**0.11154**
GBDT with PCA	4.2653	38.1074	**0.14828**

由上述结果可知，**GBDT相对于线性模型有更好的表征能力，但这也要求需要对输入特征进行预处理（如PCA正交化）**，正交化之后的泛化性能有很大提高。对于增加特征后的指标库，由于特征所含信息更多，GBDT与线性模型表现接近，但因为经过PCA处理后，GBDT泛化性能相对线性模型会有进一步提升。

接下来的实验是分别在不同个股上进行相同的训练和验证，即分别在有无PCA的情况下测试GBDT和相关指标的预测效果。

模型可视化：

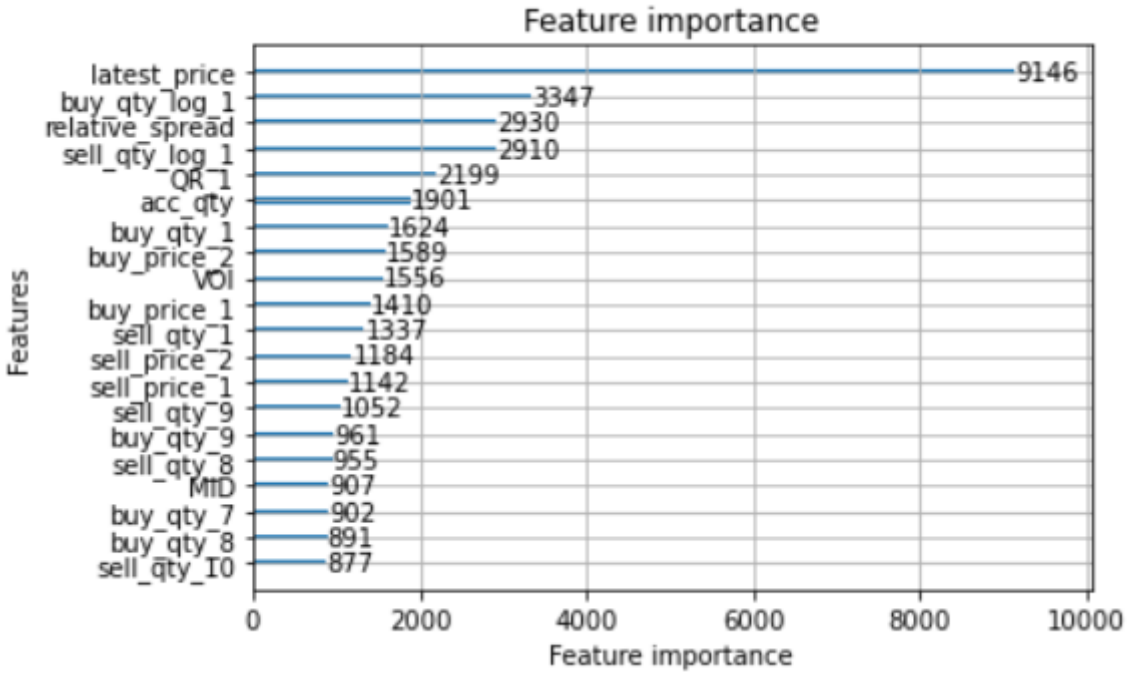


4.3 全部个股1s频率预测结果

实验设置：同上，所有个股的数据集划分比例均为9:1，采用改进后的指标库并进行标准化和PCA等预处理

000069

特征重要性：

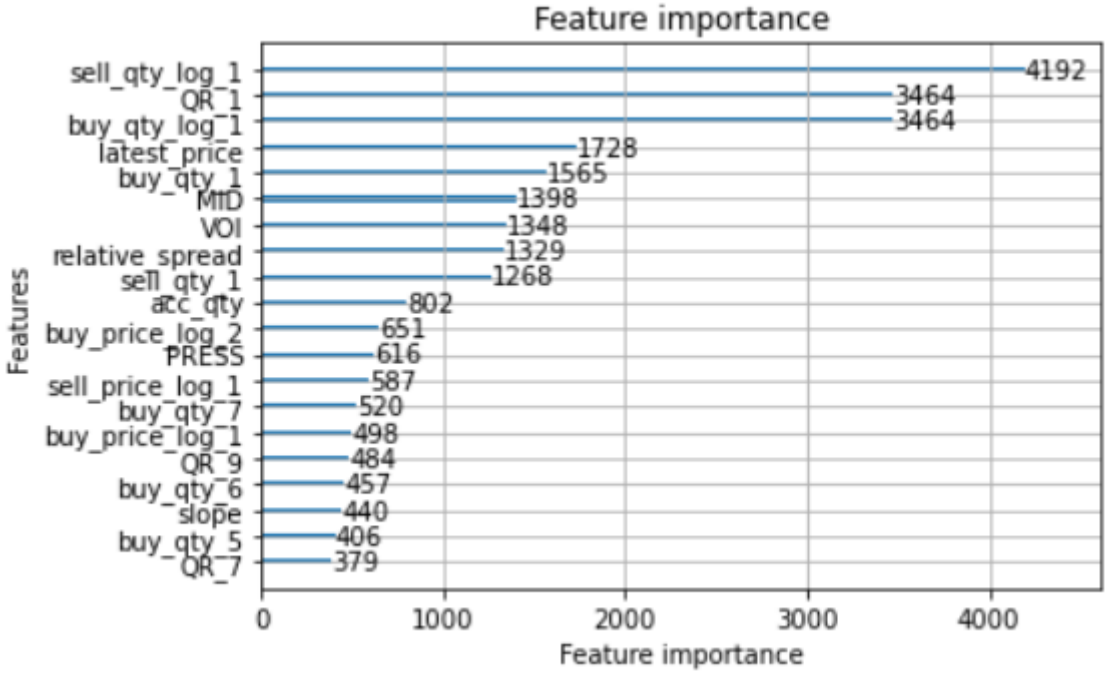


指标结果：

	mae	mse	r2
GBDT without PCA	4.0472	39.7515	0.11154
GBDT with PCA	4.2653	38.1074	**0.14828**

000566

特征重要性：

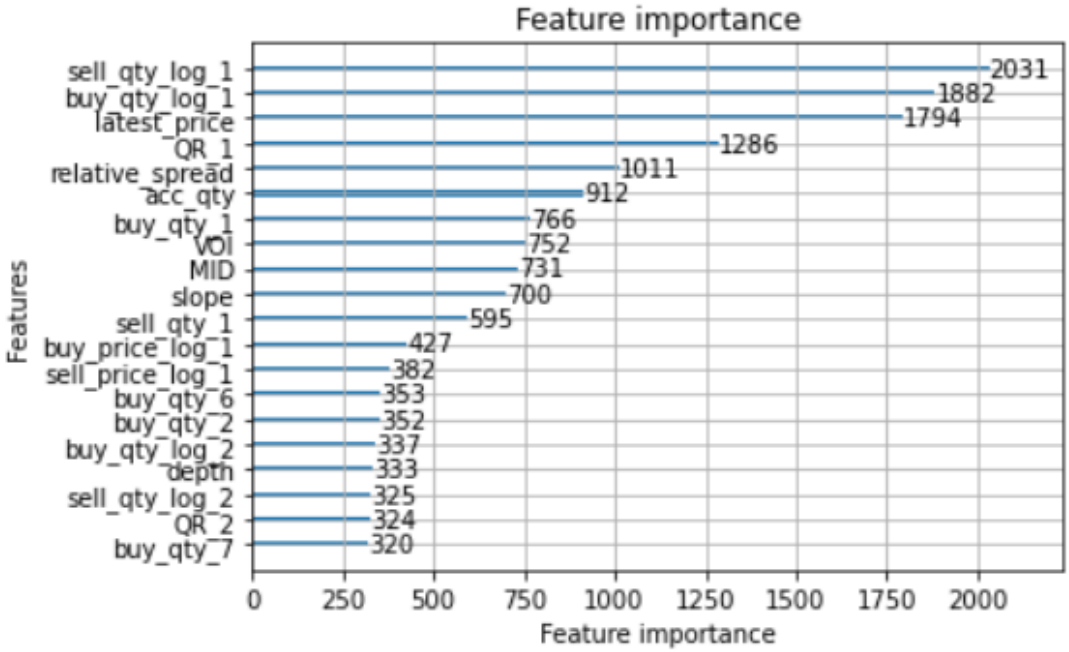


指标结果：

	mae	mse	r2
GBDT without PCA	3.8283	33.6860	0.16392
GBDT with PCA	3.8931	33.3578	**0.17206**

000876

特征重要性：

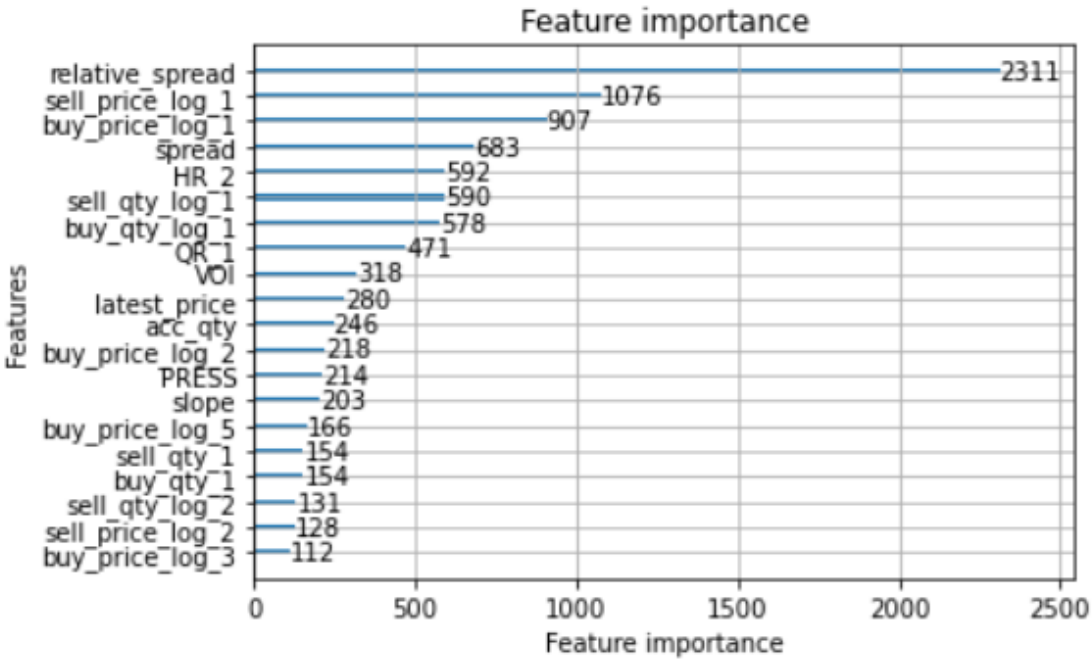


指标结果：

	mae	mse	r2
GBDT without PCA	3.2234	24.7998	0.21260
GBDT with PCA	3.2203	24.6631	**0.21694**

002304

特征重要性：

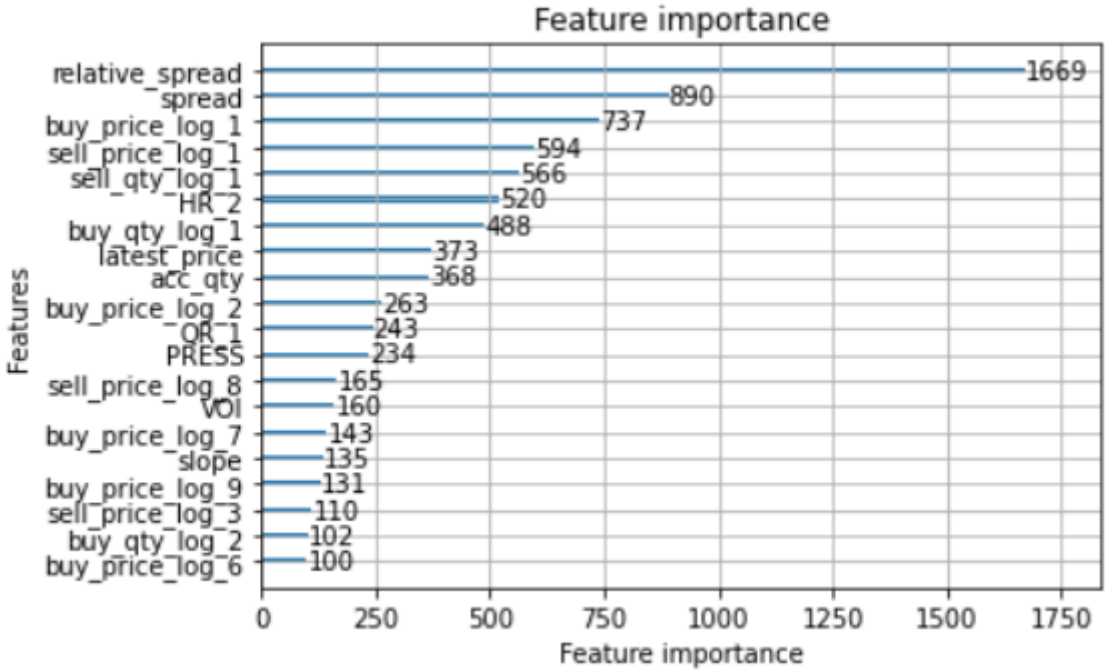


指标结果：

	mae	mse	r2
GBDT without PCA	0.9340	3.4014	0.18360
GBDT with PCA	0.9375	3.3970	**0.18466**

002841

特征重要性：

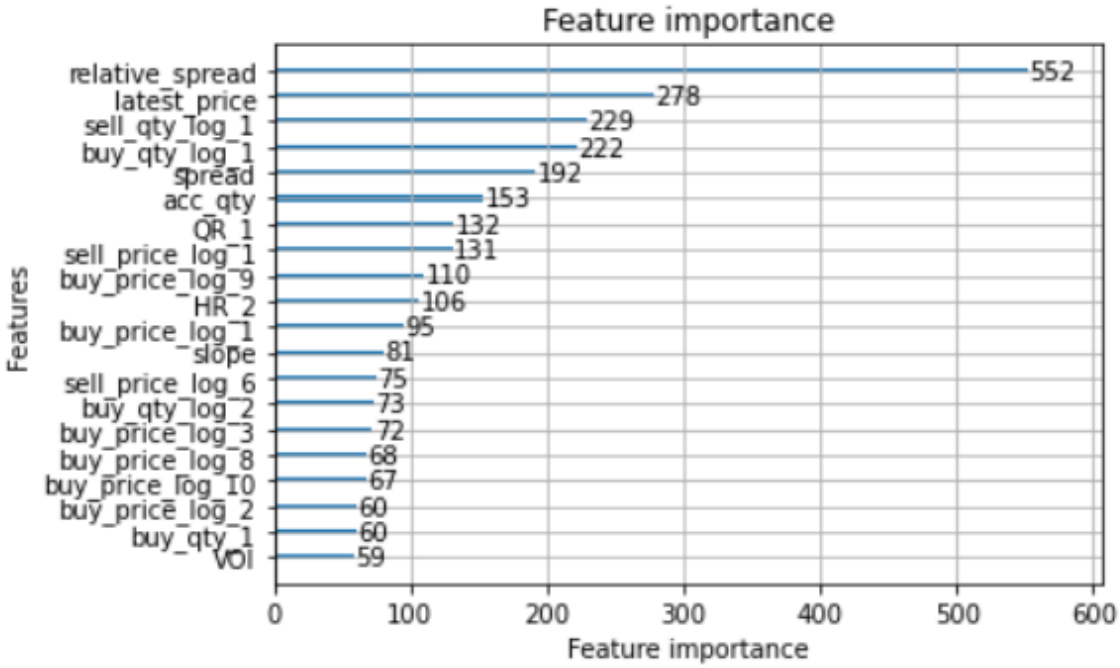


指标结果：

	mae	mse	r2
GBDT without PCA	2.1257	19.5241	0.08474
GBDT with PCA	2.1170	19.4947	**0.08612**

002918

特征重要性：



指标结果：

	mae	mse	r2
GBDT without PCA	3.0343	50.5431	0.06536
GBDT with PCA	3.0443	50.5342	**0.06553**

整体观察，通过PCA预处理后的指标库，利用GBDT模型有一定的的解释能力，对于不同个股的r2有所差异，但都大于0.05；

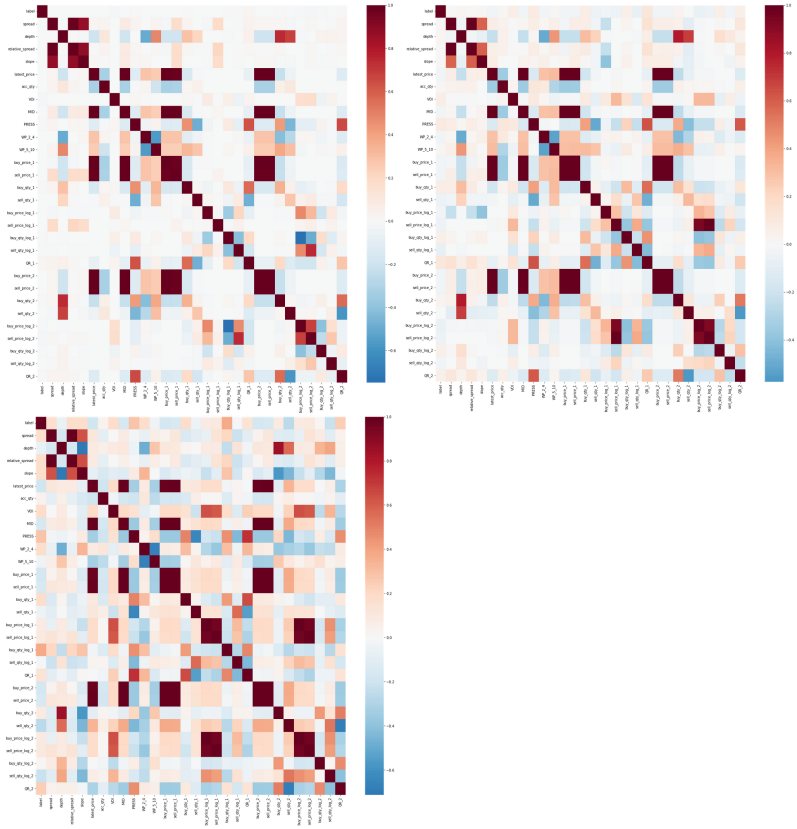
模型在部分个股上表现出色，但在部分个股上表现不佳，如002918和002841，这两只个股在处理数据的时候其实也发现了流动性可能不如其他股票那么好，下单密度相对来说低很多，虽然用了更多天的数据进行训练，但是可能稀疏的订单密度本身就缺失了很多信息，造成预测效果一般；

虽然在不同个股上预测能力有所不同，但是特征重要性有很多相似的地方，比如在所有指标当中，latest_price，relative_spread，以及各种一档量价的对数差指标有较重权重，对模型解释能力贡献较大，此外考虑本研究引入的**各种订单簿不平衡指标当中，QR指标效果较好，HR次之，两者都在多个个股特征当中重要性前列， VOI指标仅在部分个股当中重要，其他订单簿不平衡指标表现效果较为一般。**

4.4 个股不同频率收益率预测结果

指标相关性研究

当生成订单簿指标的频率分别为1s，1m，3m时，变量之间的相关系数矩阵变化如下



可见采样频率越低，损失的信息越大，各类型变量之间的相关性越强，而秒级别的变量之间相关系数普遍偏低

以000069股票为例，为别设定目标为预测1s，1m，3m的next return，结果如下：

模型统一使用GBDT，特征经过PCA预处理等操作

	mae	mse	r2
3m	19.0167	664.2065	0.02338
1m	8.6081	121.8759	0.04984
1s	3.9515	38.3388	**0.14311**

上表显示的是在不同时间尺度下的next return预测结果，由于长于三分钟的模型效果较差，这里仅展示三个样例。总体来说这里的逻辑是这样的：**由于所有的指标都是从相邻的两个时间截面上的订单簿提取的，时间跨度越长，实际订单簿与下一个订单簿之间发生的事件其实越多，仅从两个时间点的订单簿生成指标势必损失更多信息。**因此考虑高频收益率预测任务，来自订单簿的指标会有更好的预测表现。

五、高频信号回测与改进

信号回测基于实验4.2.2节的GBDT with PCA，针对个股000069做日内的信号评价，下述回测考虑的都是秒频率的换仓（实际考虑交易规则和流动性等因素限制，并且秒频换仓会带来巨大的交易成本，因而这样的回测并不实际，但是可以通过回测对信号的表现进行分析）

个股：000069

频率：1s

时间：训练集 20200110-20200310（36天 510737 样本），验证集 20200311-20200316 （4天 49703 样本）

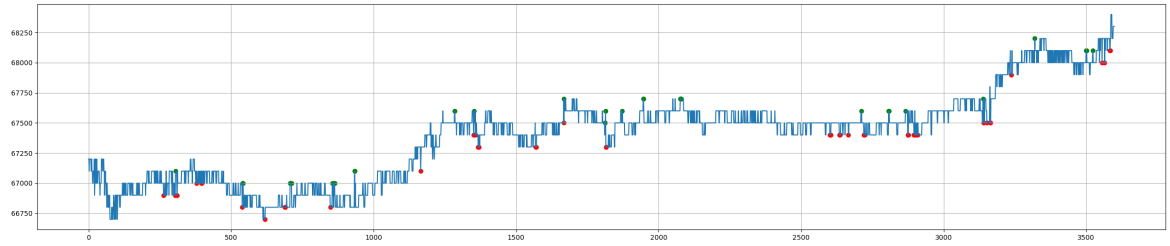
信号设置：由于每条数据都有一个对应的对于1秒后的next return预测值，可以根据预测值的高低设计买入和卖出的信号。考虑某一时间点，当其收益率预测值大于过去某一窗口`window`预测值的均值`mean`+标准差`std`*某一阈值`threshold`，即认为该信号有向上突破的异常趋势，视为买入信号；相反则是卖出信号，在测试集上回测，结果如下：（共4天），手续费双边0.2%

window/threshold	buy times	sell times	buy correct/false rate	sell correct/false rate	win_loss rate
120/2	565	841	0.2743/0.0017	0.2175/0.0035	0.3078
120/2.5	119	166	0.2857/0.0001	0.1987/0.0060	0.3302
600/2.5	83	102	0.2530/0.0120	0.2549/0.0392	0.5026
1800/2.5	103	71	0.4271/0.0097	0.3239/ 0.01408	0.9209
3600/2.5	85	75	0.4588/0.0117	0.3466/0.0133	1.0085

仅仅观察盈亏比，信号表现比较一般。当考察的时间窗口越长，信号效果越好，如当时间窗口为一个小时，盈亏比逐渐大于1，但信号次数减少，另外考虑到每天开盘后一个小时可能无法进行准确的信号生成，由此也会带来一定的收益潜在损失。

但是在高频收益率当中，实际1s频率的收益率大部分都是0（可能因为挂单未成交，价格未更新，0收益率大约占了80%），考虑信号质量，当单独考虑买卖信号时，买/卖方向正确与方向错误的比例实际相当可观，参考表格中`buy correct/false rate`和`sell correct/false rate`，也就是说如果进行预测，下一秒价格如有所更新，那么信号预测准确率其实比较准确。从这个角度分析，**实际模型提取的预**

测指标具有较好的方向性预测能力，但是并没有太强的波动率预测能力，即可以判断涨跌方向，但是对于何时涨跌能力有限，由于存在交易费用，盈亏比并不理想。



如上图所示，在20200311的前一个小时中，信号方向整体正确，代表买信号的红色和代表卖信号的绿色，很少出现错误预测的情况。

六、结论

4.1.2 4.1.3都基于基础指标库进行模型训练，分别采用线性模型（OLS和Ridge regression）和GBDT进行1s频率的next return预测，GBDT表现好于线性模型，并且GBDT的变量解释能力更强，通过树结构可以较直观获得指标重要性；

4.2 向指标库添加了包括各种订单簿不平衡指标，利用GBDT进行训练，并测试了对特征进行PCA的对比试验，发现相比相比基础指标库，r方提升明显，另外通过PCA预处理的特征，用GBDT训练后预测能力提升明显，并超过线性模型baseline，整体来看模型：PCA+GBDT>Ridge>GBDT, OLS ；

4.3在不同个股上分别训练模型并进行调参，利用GBDT+指标库+PCA可以实现一定程度上的预测，并且可以通过GBDT筛选出重要性较高的因子：latest_price，relative_spread，以及各种一档量价的对数差，QR，HR；

4.4在不同时间尺度上进行试验，发现指标库的有效性更多体现在高频，频率越低有效性越低；

最后进行了信号的合成与回测，发现信号有较强的方向性预测能力，但是对于何时涨跌的预测能力有限。

【创新点】

引入订单簿不平衡指标（见第二节）

通过gbdt进行模型预测与优化，并增强模型解释能力（见第四节）

【改进方向】

在进行预测时，每个个股其实都要在自己的数据集上重新训练，找最优参数。由于数据处理能力的限制，并且每个股票的订单流密度差异很大，生成数据集的时候其实不同个股的数据集大小不同，虽然考虑了同样的划分比例，但导致需要根据具体情况，调整生成GBDT的某些参数（如min_data_in_leaf），比较麻烦，之后可以考虑采用更一致的训练集，并进行滚动验证。

【附录】

模型参数选择：

boosting_type：用于指定弱学习器的类型，默认值为 ‘gbdt’，表示使用基于树的模型进行计算。还可以选择为 ‘gblinear’ 表示使用线性模型作为弱学习器。
其他的参数可以选择：
‘gbdt’，使用梯度提升树

‘rf’，使用随机森林
‘dart’，不太了解，官方解释为 Dropouts meet Multiple Additive Regression Trees
‘goss’，使用单边梯度抽样算法，速度很快，但是可能欠拟合。
推荐设置为 'gbdt’

objective：指定目标可选参数如下：
“regression”，使用L2正则项的回归模型（默认值）。
“regression_l1”，使用L1正则项的回归模型。
“mape”，平均绝对百分比误差。
“binary”，二分类。
“multiclass”，多分类。
num_class用于设置多分类问题的类别个数。

min_child_samples：叶节点样本的最少数量，默认值20，用于防止过拟合。

learning_rate / eta：LightGBM 不完全信任每个弱学习器学到的残差值，为此需要给每个弱学习器拟合的残差值都乘上取值范围在(0, 1] 的 eta，设置较小的 eta 就可以多学习几个弱学习器来弥补不足的残差。推荐的候选值为：[0.01, 0.015, 0.025, 0.05, 0.1]

max_depth：指定树的最大深度，默认值为-1，表示不做限制，合理的设置可以防止过拟合。
推荐的数值为：[3, 5, 6, 7, 9, 12, 15, 17, 25]。

num_leaves：指定叶子的个数。

feature_fraction / colsample_bytree：构建弱学习器时，对特征随机采样的比例，默认值为1。推荐的候选值为：[0.6, 0.7, 0.8, 0.9, 1]

bagging_fraction / subsample：默认值1，指定采样出 subsample * n_samples 个样本用于训练弱学习器。注意这里的子采样和随机森林不一样，随机森林使用的是放回抽样，而这里是不放回抽样。 取值在(0, 1)之间，设置为1表示使用所有数据训练弱学习器。如果取值小于1，则只有一部分样本会去做GBDT的决策树拟合。选择小于1的比例可以减少方差，即防止过拟合，但是会增加样本拟合的偏差，因此取值不能太低。注意： bagging_freq 设置为非0值时才生效。推荐的候选值为：[0.6, 0.7, 0.8, 0.9, 1]

bagging_freq / subsample_freq：数值型，默认值0，表示禁用样本采样。

lambda_l1：L1正则化权重项，增加此值将使模型更加保守。推荐的候选值为：[0, 0.01~0.1, 1]

lambda_l2：L2正则化权重项，增加此值将使模型更加保守。推荐的候选值为：[0, 0.1, 0.5, 1]

min_gain_to_split / min_split_gain：指定叶节点进行分支所需的损失减少的最小值，默认值为0。设置的值越大，模型就越保守。推荐的候选值为：[0, 0.05 ~ 0.1, 0.3, 0.5, 0.7, 0.9, 1]

min_sum_hessian_in_leaf / min_child_weight：指定子节点中最小的样本权重和，如果一个叶子节点的样本权重和小于min_child_weight则拆分过程结束，默认值为1。推荐的候选值为：[1, 3, 5, 7]

metric：
用于指定评估指标，可以传递各种评估方法组成的list。常用的评估指标如下：
‘mae’，用于回归任务，效果与 ‘mean_absolute_error’， ‘l1’ 相同。
‘mse’，用于回归任务，效果与 ‘mean_squared_error’， ‘l2’ 相同。
‘rmse’，用于回归任务，效果与 ‘root_mean_squared_error’， ‘l2_root’ 相同。
‘auc’，用于二分类任务。
‘binary’，用于二分类任务。
‘binary_logloss’，用于二分类任务。
‘binary_error’，用于二分类任务。
‘multiclass’，用于多分类。
‘multi_logloss’， 用于多分类。

‘multi_error’， 用于多分类。
seed / random_state 指定随机数种子。

详见中文文档 LightGBM中文文档

联系方式

郗琛 2023.3.9

m18191419729@163.com