# 利用企业知识图谱与机器学习智能选择最优投资组合

## 一、模型背景介绍

### 1．企业知识图谱

企业知识图谱是商业领域中一种新兴工具，它能够有效地整合来自不同数据源的企业数据[1]。本研究中的企业知识图谱是企业的商业网络，知识图谱中的实体即为企业，实体间的连接则为企业间的商业关系，包括供应关系，债务关系，控股关系等。企业知识图谱曾被成功地运用在商业领域中，比如在 2017[2]，研究人员曾经利用企业知识图谱挖掘企业和机构间的投资关系。本研究将结合企业知识图谱和机器学习，产生合理的选股策略，收获稳定的 alpha 收益。

### 2. 关联企业对股价的影响

以往的研究[3]发现，当满足以下特定的条件时，关联企业股价表现对当前企业有着一定的滞后影响：

- 两家企业有着相关联的基本面，且关于这些基本面的信息在市场上是广泛存在的，因此，当前企业的股价信息包含着其关联企业的股价表现信息；
- 投资者能够获取企业的信息以及企业之间的关系；
- 投资者不能有效地获取股票的所有公共信息变化，且某些投资者只能获得部分信息。根据这个法则，市场间的信息传播是缓慢的，从而导致了当前企业的股价表现和关联企业的股价表现之间存在滞后；

本研究中企业知识图谱包含的关系（上下游，控股，投资等），经检验均满足以上的法则，因此相对应的关联公司的股价表现都有可能对当前公司的股价表现产生一定的滞后影响。同时，由于信息可能在企业关系网中进行多重传播，例如金融风险和信用风险可能沿着供应链传播[4,5]，当前企业的信用风险不仅受到直接关联的上下游供应商的影响，还受到间接关联的上下游供应商的影响，因此，本研究认为间接关联的关联企业股价表现仍然影响当前企业的股价。因此，解释当前企业未来股价表现的因子中，本研究加入了间接关联企业的股价表现。

由于企业之间间接关联的存在，对于企业知识图谱中的一家企业，如果这家企业能通过某种路径连接到当前企业，那么本研究认为这家企业实际上就是当前企业的关联公司，其关联便是路径。例如在图 1 中的简单企业知识图谱所示，虽然简单企业知识图谱中只包含了供应商，客户这两类关系，但实质上对于当前企业 D 来说，却有着多重的关联公司，除去企业 C 是其的直接供应商外，企业 B 也通过一条路径 BCD 对企业 D 形成了间接供应关系，本研究中称企业 B 为企业 D 的二级供应商，同样企业 A 和 E 对企业 D 形成了三级供应关系。注意到，对于两家公司间可能存在着不同的路径可以互相抵达，因此本研究认为相同的一家关联企业，对于目标企业可能存在着不同的关联关系。

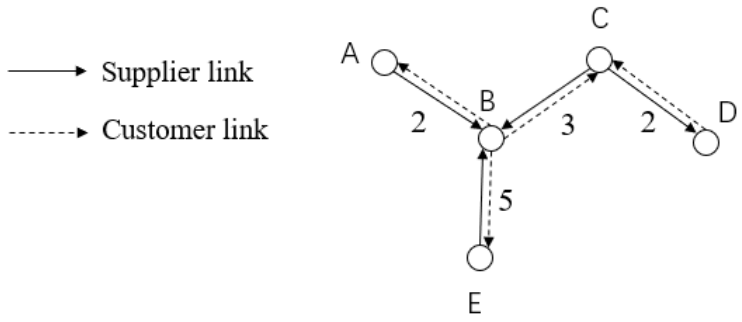


图 1 企业知识图谱简化示例

## 3．机器学习与深度学习预测股价

深度学习曾被广泛运用于预测股票收益上，并展示出了其相对于传统回归方法的优势。Cao[6]等人证明了在中国市场上，神经网络的预测效果比线性模型更好。前人的研究采用多类影响股价的因子作为深度学习的输入，并取得了良好的效果。最为经典的因子模型为FamaFrench[7]三因子模型，包括市场资产组合因子、市值因子、账面市值比因子。此外，Carhart 的市场动量因子也被证明有助于提升预测效果。但大多数研究中，考虑的仍是财务因子。例如 Olson and Mossman[8]用 61 个财务因子预测下一年的股票收益率，发现神经网络的效果比传统回归方法要好。Kryzanowski [9]等人采用财务指标和宏观经济变量预测下一年股票收益情况，神经网络预测分类正确度达到 72%。

在选择好因子后，常用的方法是选择监督式学习，输入因子，预测股票收益率。在选用具体分类器或回归模型上，过往的研究较多专注于比较传统回归方法与机器学习或深度学习，证明了机器学习或深度学习预测能力远高于传统回归模型。传统的预测模型是基于时间序列模型，假设股价变化是服从随机过程。但 FuliFeng[10]等人认为由于实际市场波动剧烈，该假设不成立。W.Huanga[11]认为支持向量机比传统的统计方法能够获得更好的结果。O.Hegazy 和 Salam[12]优化了支持向量回归的参数，在美股市场上回归误差显著减小。Krauss[13]等人使用三种不同的机器学习模型，深度神经网络，梯度提升树和随机森林来预测标准普尔 500 指数成分股的下一天的股票收益率，将这三者等权重结合起来作为集成，预测结果优于每个单独的模型。 在单个模型中，随机森林优于深度神经网络和梯度提升树。W.bao 和 rao[14]等人提出的模型兼顾了准确度和预测效果，首先对金融数据进行小波变换，然后通过自动编码器处理去除噪音后的数据，生成特征因子，最后用 lstm 网络模型做预测。过往研究中，输入因子后另一种方法是预测股票趋势。Nguyen[15] 等人采用支持向量机分类，输入股票留言板上股票的话题信息，预测股票的走势。

过往研究都专注于优化机器学习或深度学习的模型参数，提高预测或分类的准确度，而未在选择输入因子上做太大的创新，其只考虑了当前股票的走势（技术指标）或基本面因子，却并未考虑企业间的关联对其走势的影响。

本文创新地考虑了股票所在公司间，存在一定的关系，例如产业上下游关系，控股投资关系等等，并把关联企业的影响转化为关联企业影响因子。在实证回测中，该因子被证明是有助于预测的。

# 二、策略框架

本研究策略框架如图 2 所示，先是用网络爬虫和自然语言新闻挖掘等手段整合企业关

联数据，从而构建以全体 A 股上市公司为核心的企业知识图谱。而后本研究又从 Tushare(股票公开数据源)提取出股票数据，与企业知识图谱数据相结合计算出各类可能存在超额收益的关联企业影响力因子。而后我们将数据放入多类机器学习模型中进行训练，挑出在测试集上表现最佳的模型。最后根据该模型预测出下一周股票收益正概率的排序，选出排序较前的股票进行持仓。

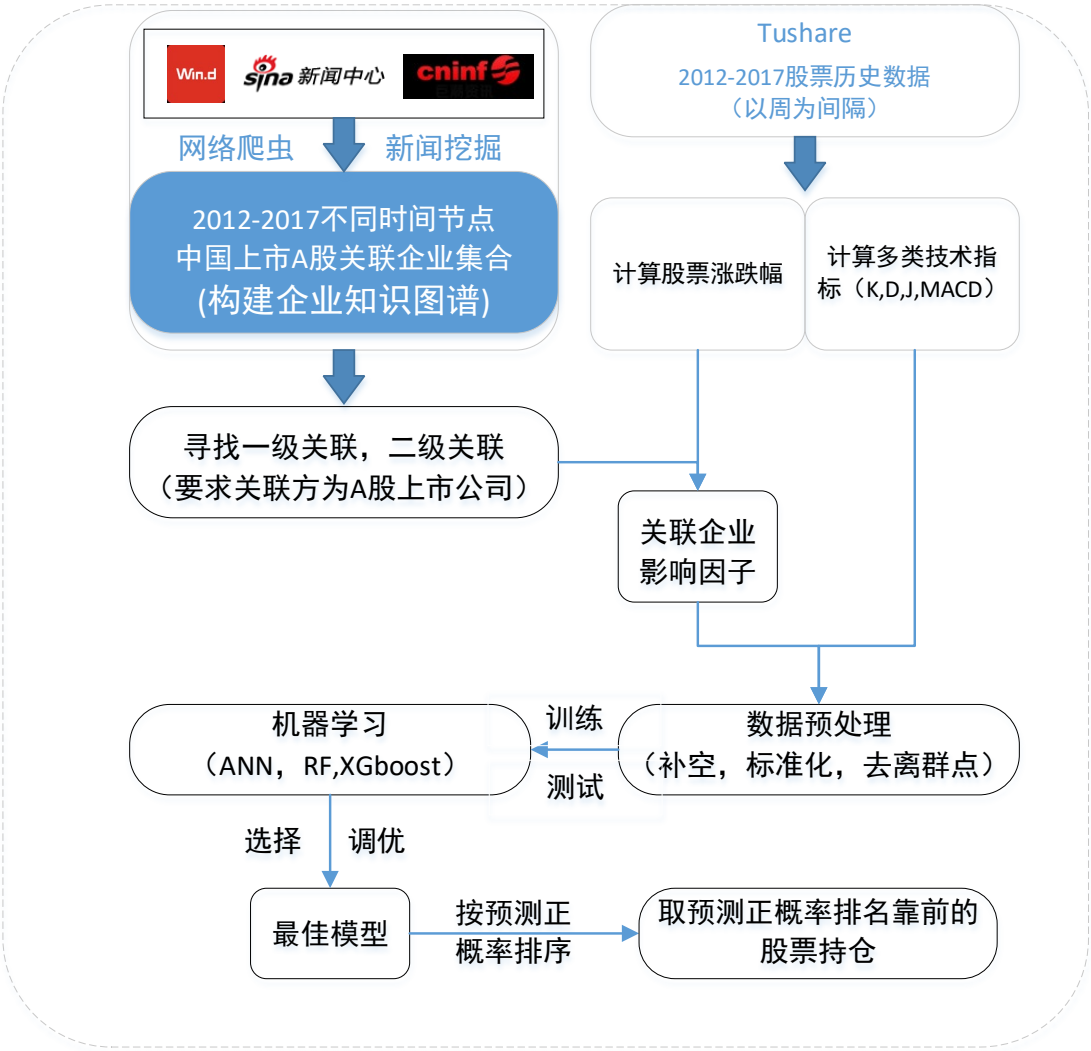


图 2：策略框架

# 三、数据取得以及预处理

## 1.数据获取以及数据描述

### 1.1 企业知识图谱搭建

本文研究的企业知识图谱是基于关联企业的信息，以全体 A 股为核心。本研究中企业之间的关联可能是动态变化的，每个关联都有相对应的时间节点。

首先，我们从 Wind 终端等公开数据源上，获得了上市 A 股的关联企业的信息。然后，作为补充，我们利用自然语言处理的方法从新浪新闻等新闻媒体网站挖掘关联企业的信息，比如公司公告已确认某公司为供应商，或和某公司解除供应关系。最终，在我们搭建的企业知识图谱中，企业关联个体的类型包括：个人，组织，A 股上市公司，非 A 股上市公司（可

以在美股或港股上市)。 其中关联个体和本公司的关系包括：上游，下游，债务，股东，投资，并购，同行，概念，集团，高管，诉讼，产品。关联个体为关联的定义如表 1 所示，关联的统计数据如表 2 所示

| 关联名称 | 含义 |
|---|---|
| **上游** | 本公司供应链中的主要上游供应商 |
| **下游** | 本公司供应链中的主要下游客户 |
| **债务** | 与本公司有债务关系（担保，持债等）的主要个人或组织 |
| **股东** | 持有本公司股份的主要个人或组织 |
| **投资** | 被本公司持有股份的主要企业 |
| **并购** | 本公司对外主要的出售标的 |
| **同行** | 与本公司在 WIND 行业分类下归属于同一行业的企业 |
| **概念** | 与本公司属于同一概念板块的企业 |
| **集团** | 与本公司位于同一集团的组织 |
| **高管** | 本公司的主要高管 |
| **诉讼** | 与本公司发生过诉讼纠纷的组织 |
| **产品** | 其主要经营产品和本公司有重合的公司 |

表 1 关联企业的定义

| 全部 A 股 | 上游 | 下游 | 债务 | 股东 | 投资 | 并购 |
|---|---|---|---|---|---|---|
| | 14017 | 24237 | 15635 | 47719 | 18520 | 17450 |
| | 同行 | 概念 | 集团 | 高管 | 诉讼 | 产品 |
| | 72059 | 198534 | 13350 | 48395 | 3652 | 5109 |

表 2：企业知识图谱关联边数目统计

此外，在基本的关系之上，本研究还搜索了图中存在的一些二级关系，由于企业的间接关联影响会随着两企业之间的企业数目增加而衰减，本研究只考虑影响较大的二级关系。针对于全部 A 股，在企业知识图谱中搜索出以及其为起始点的所有二元路径，作为该企业的间接关系总和。本研究筛选出末端节点亦为 A 股上市公司的路径（注意中间节点除去 A 股之外，亦可以为个人，组织或非 A 股上市公司）

本研究中最终影响当前企业股价表现的关联公司不包括在美股或港股上市的公司，原因在于本研究认为不同市场之间存在着信息隔离，美股上市企业对 A 股上市企业的影响与 A 股上市企业之间的互相影响有区别，因此，研究中最后纳入考虑的关联公司，只是在中国大陆上市的公司（大部分为 A 股上市，有少部分在 B 股上市）。

1.2 获取股票数据

本研究从 Tushare 接口上获得了所有 A 股上市公司从 2012 年初到 2017 年底的数据。数据以周作为单位，具体表现为每周最后一个交易日的数据。原始数据的指标包括 time(时间), code(股票代码),open(开盘价), close(收盘价),volume(当日成交量), 其中股票价格采用前复权。我们用该数据按如下方式计算股价的 pct_change（涨跌幅），t 时刻的涨跌幅表示 t-1 时刻到 t 时刻的股价波动：

$$pct\_change_t = \frac{(close_t - close_{t-1})}{close_{t-1}}$$

# 2.数据预处理

2.1 数据补空

对于股票的基本价格数据（开盘价，收盘价等），如果是空值的话，一般是由于该股票停牌造成的，因此本研究就延用停牌前最新一期的价格数据。对于由停牌造成的股票当天成交量为空值的情况，本研究则用补 0 处理。

2.2 指标计算

在得到了股票的基本数据以及企业知识图谱数据之后，本研究通过一定的原则生成可能对股票收益有影响的因子。因子类型包括技术指标和关联企业影响因子。

本研究选取了多类技术指标进行计算，计算方式如表 3 所示。

| 指标中文名 | 指标英文名 | 指标计算公式 |
| --- | --- | --- |
| **随机指标** | KDJ | $RSJ=\frac{C_n-L_n}{H_n-L_n}\times 100$<br>$K_n=\frac{2}{3}K_{n-1}+\frac{1}{3}RSV_n$<br>$D_n=\frac{2}{3}D_{n-1}+\frac{1}{3}K_n$<br>$J_n=3K_n-2D_n$ |
| **指数平滑异同平均线** | MACD | $EMA_n=\alpha Price_{today}+(1-\alpha)EMA_{n-1}$<br>$DIF=EMA(12)-EMA(26)$<br>$DEA_n=\frac{8}{10}DEA_{n-1}+\frac{2}{10}DIF_n$<br>$MACD_{bar}=2(DIF-DEA)$ |
| **能量潮指标** | OBV | $OBV_n=OBV_{n-1}+sign\cdot Vol_n$ |
| **相对强弱指标** | RSI | $RS=\frac{avgup_n}{avgdown_n}$<br>$RSI=100\times\frac{RS}{RS+1}$ |
| **心理线** | PSY | $PSY=\frac{UP_n}{N}\times 100\%$ |
| **人气意愿指标** | BRAR | $AR_n=\frac{\sum_n(H-O)}{\sum_n(O-L)}\times 100$<br>$BR_n=\frac{\sum_n(H-C_T)}{\sum_n(C_T-L)}\times 100$ |
| **中间意愿指标** | CR | $P1=\sum(H-YM)$<br>$P2=\sum(YM-L)$<br>$CR_N=\frac{P1}{P2}\times 100$ |
| **乖离率** | BIAS | $BIAS=\frac{Close_n-avg(N)}{avg(N)}\times 100\%$ |
| **顺势指标** | CCI | $TP=\frac{High+Low+Close}{3}$<br>$MA_n=\frac{\sum_n close_n}{n}$<br>$MD=\frac{\sum_n(MA_n-close_n)}{n}$<br>$CCI(n)=\frac{TP-MA}{0.0015MD}$ |
| **威廉指标** | WR | $WR_n=100\times\frac{high_n-C}{high_n-low_n}$ |

| | | |
|---|---|---|
| **三重指数平滑移动平均** | TRIX | $TR_n = \frac{\sum_{i=1}^{n} close_{t-i}}{n}$<br>$TRIX_n = \frac{TR_n - TRIX_{n-1}}{TR_{n-1}} \times 100$<br>$MATRIX_M = \frac{\sum_{i=1}^{M} TRIX_{t-i}}{M}$ |

表 3：研究所采用的技术指标以及其计算方式

- 对于关联企业影响因子，首先我们认为关联企业因子影响的原理是当前企业$t+n$($n$=1, 或 2 或 3)时刻的股价可能会受到关联企业 t 时刻股价涨跌幅的影响。且对于单一关联来说，每一家企业可能影响当前企业股价表现，且关联企业的数目越多，影响力越大。因此我们将对应于某一关联 r 的关联企业影响因子的值认为是关联企业集合内企业 t 时刻的股价涨跌幅总和。设当前公司于 t 时刻在该关联 r 上的关联公司集合为 C，则该关联 r 在 t 时刻的影响力因子$I_r$为：

$$I_r = \sum_{i \in C} pct\_change_{i,t}$$

但是不同类型的关联表现出来的最优影响滞后时间可能是不同的。举例而言，二元关系的最佳影响滞后时间可能是两周，因为其信息的传播时间比一元路径来的更长。因此，对于预测 t+1 时刻的股价涨跌幅，我们将原有的因子做出一定的修正。设关联 r 的最优影响滞后时间为$T_r$，则其影响力因子$I_r$被修正为：

$$I_r = \sum_{i \in C} pct_{change_{i,\ t-T_r}}$$

为了得出每种关联的最优影响滞后时间，我们分别计算 t 时刻的$I_r$与 t+1,t+2,t+3 的股票涨跌幅（pct_change）之间的 pearson 相关系数，取出其相关系数绝对值最大的时间滞后来作为最优影响滞后时间，在所有的关系中，其相对应的最优影响滞后时间的统计量为表 4 所示

| 关联数目 | 1 周 | 2 周 | 3 周 |
|---|---|---|---|
| 137 | 137 | 0 | 0 |

表 4：不同关联的最优影响滞后时间统计数据

- 在计算出技术指标和关联企业影响力因子后，为了平稳同一指标间数据分布，减小不同指标间数据分布的差异，本研究按如下公式对因子进行标准化处理：记因子 x 的样本为 X，则：

$$\mathrm{X} = \frac{(\mathrm{X} - \bar{X})}{Std(X)}$$

2.3 离群点处理

为了排除异常值（离群点）对模型训练的噪声干扰，本研究需要用算法清除离群点。考虑到本研究的特征大多是连续的变量，因此本研究使用了孤立森林对离群点进行清除处理。在这里，我们认为离群点是特征空间中分布稀疏，离密度高的群体较远的点。在特征空间里，分布稀疏的区域表示事件在该区域发生的概率很低，因而可以认为落在这些区域里的数据是异常的。

孤立森林方法的基本原理是将特征空间中的样本点进行分割。位于密度较大区域的样本点需要较多的分割次数才能被“孤立出来”，而位于密度较小区域的样本点则只需要较少的分割次数就可以被“孤立出来”，由此本研究可以用分割孤立的方法对数据进行一个离群点的检测，孤立森林将对每个样本都返回一个样本异常值评分，异常值越小的越有可能是异常样本。在得到了所有样本的异常评分后，本研究将样本点按异常评分从大到小进行排序，并剔除排序后 5%的样本点。

# 四、模型的搭建

## 1. 模型输入与输出

经过数据预处理之后，我们得到了具有一系列股票特征的样本。为了训练模型，本研究将股票特征分为模型的输入与输出。

### 1.1 模型的输入

由于本策略研究的是以周为时间间隔对持有股票进行调仓，若以周为频率的因子数据前后不发生积极变化，则其对于预测周与周之间的股价涨跌幅是不合适的。因此本研究舍弃了股票的基本面指标，原因在于基本面指标在相邻周之间变化差异不显著，难以解释相邻周之间的股价变动。

如图 2 所示，我们要预测的是股价的走势，股票的走势受到自身以及外在的影响，即用技术指标来表示当前股票趋势，用关联企业影响力因子来表示关联企业对当前股票股价的影响。注意关联企业影响力因子也在一定程度上反应宏观因子对股价的影响。诸如同业的关联股价影响因子，若与当前公司行业相同的股票在 t 时刻大多数是涨的，代表这个行业就被看好。

此外，因为股票的价格走势还受到股票重大新闻的影响，因此本研究需要针对每只股票剔除其重大新闻发布的时间点，重大新闻具体表现为大股东减持，股权质押，这类直接影响股票价格的事件。

在确定完模型的输入后，本研究利用 SKlearn 中的单变量特征选择的方法对每个因子和股票未来时刻的涨跌幅的相关性做了研究，具体结果如表 5 所示。发现因子和未来涨跌幅之间存在着显著关联，且关联企业影响力因子的关联度在所有因子中是非常显著的。同时，二级关联的相关性甚至要高于大多数一级关联，这可能是因为对于一家企业而言，其一级关联的企业数目是非常有限的，而二级关联的企业数目明显多于一级关联的企业数目，这使得二级关联的影响力总和要大于一级关联。

### 1.2 模型的输出

本研究将 t+1 时刻的股价波动作为 t 时刻的预测目标，也就是模型的输出。由于机器学习大多是一个分类模型，且考虑到模型只需要预测出较高收益的一部分股票，为了扩大预测正确的收益以及减小预测错误的代价，我们将模型的预测目标 y 值做如下的修改：

$$y_t = \begin{cases} 1, & pct\_change_{t+1} > threshold \\ 0, & else \end{cases}$$，其中 threshold 为某一待定非负数值

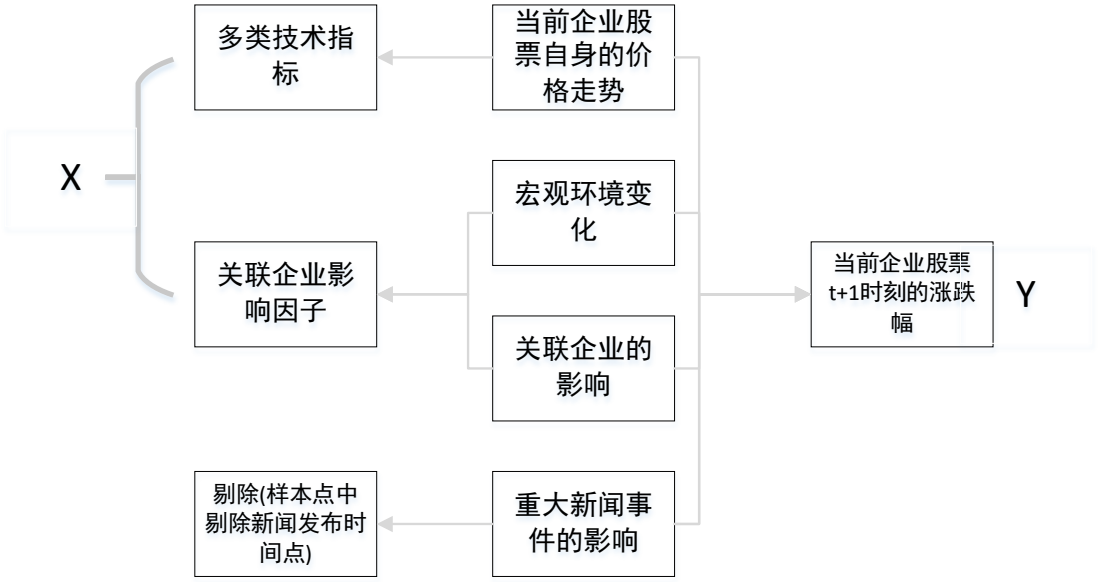


图 3：模型输入输出

## 2.训练集与测试集的划分

模型的训练集目的是让模型学习历史的数据分布，模型的测试集目的则是测试用历史数据训练的模型在未来的数据中的预测表现。因此，本研究中训练集和测试集的划分在时间上需有前后顺序。在这里我们考虑如下的训练集和测试集的划分方法：将 2012-2017 的数据中时间靠前的 75%的数据作为训练集，时间靠后的 25%的数据作为测试集。

## 3.回测及模型调优

在本研究中，我们综合比较了各类模型预测的能力和结果，并对每个模型进行格点法调参以确定最佳参数。我们用混淆矩阵（精确率，召回率）衡量模型的分类表现。值得注意的是，由于我们的模型不要求选出所有会上涨的股票，而要求选出的会上涨的股票必须大概率是真实向上的，因此在模型的分类结果中，预测正类的精确率比召回率更加重要。本研究也倾向于选择高精确率的模型来保证一定的收益。

本研究的回测平台为米筐（ricequant）。回测的周期为 2012 年 1 月 1 日到 2017 年 12 月 29 日，针对于该时间段内每周的最后一个交易日（一般为星期五），本研究的模型将根据预测的股票正概率的排序，选出排名靠前的股票（若预测正类的股票数小于 5，则取符合要求的全部股票）来作为该时点新添的股票持仓(在调优过程中，新持仓的股票为 1)。针对新持仓的股票，策略对每只股票购进 1000 股，于此同时卖出上周持仓的股票。回测策略初始化金额为 100000。

### 3.1 最佳 threshold 调优

在本研究之前对模型输出的定义中，运用了 threshold 参数。本研究要通过取值不同 threshold，并借助随机森林模型，来挑选使得模型分类表现以及回测收益率效果最好的 threshold 取值。在这里，我们将 threshold 做如下取值：0，0.02，0.05，0.1，0.2。分别为股票历史数据涨跌幅的前 50%分位数，前 36%分位数，前 18%分位数，前 7%分位数，前 2%分位数。各 threshold 取值的模型表现如表 5 所示。(注意本研究的训练集和测试集并非随机划分，所以每个模型只对应固定的一种分类表现。)

由于不同 threshold 对应的模型分类表现并不能直接反应回测时收益率的好坏，因为不同 threshold 的模型预测正类且正确的收益是不同的。我们用训练出的每个模型分别构建策

略，并在米筐平台上回测，得到了不同的收益率曲线，如图 4 到图 9 所示（黑色圆圈为测试集数据所在的时间段）

表 5：各 threshold 取值的模型分类表现

| threshold | Precision（0） | Recall（0） | Precision（1） | Recall（1） |
|---|---|---|---|---|
| 0 | 0.57 | 0.37 | 0.46 | 0.66 |
| 0.02 | 0.74 | 0.89 | 0.32 | 0.14 |
| 0.05 | 0.90 | 1.00 | 0.62 | 0.06 |
| 0.1 | 0.97 | 1.00 | 0.81 | 0.18 |
| 0.2 | 1.00 | 1.00 | 0.77 | 0.51 |
| 0.3 | 1.00 | 1.00 | 0.75 | 0.72 |


图 4：回测表现（threshold = 0）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 213.300% | 21.000% | 0.144 | 0.783 | 0.655 | 1.117 | 0.45 |
| **基准收益** | **基准年化收益** | **Volatility** | **MaxDrawdown** | **Tracking Error** | **Downside Risk** | **最大回撤/最大回撤持续期** |
| 74.500% | 9.700% | 0.328 | 55.800% | 0.277 | 0.192 | MaxDD 2015-06-11 00:00:00~2015-09-15 00:00:00, 96 days<br>MaxDDD 2015-06-11 00:00:00~2017-12-29 00:00:00, 932 days |

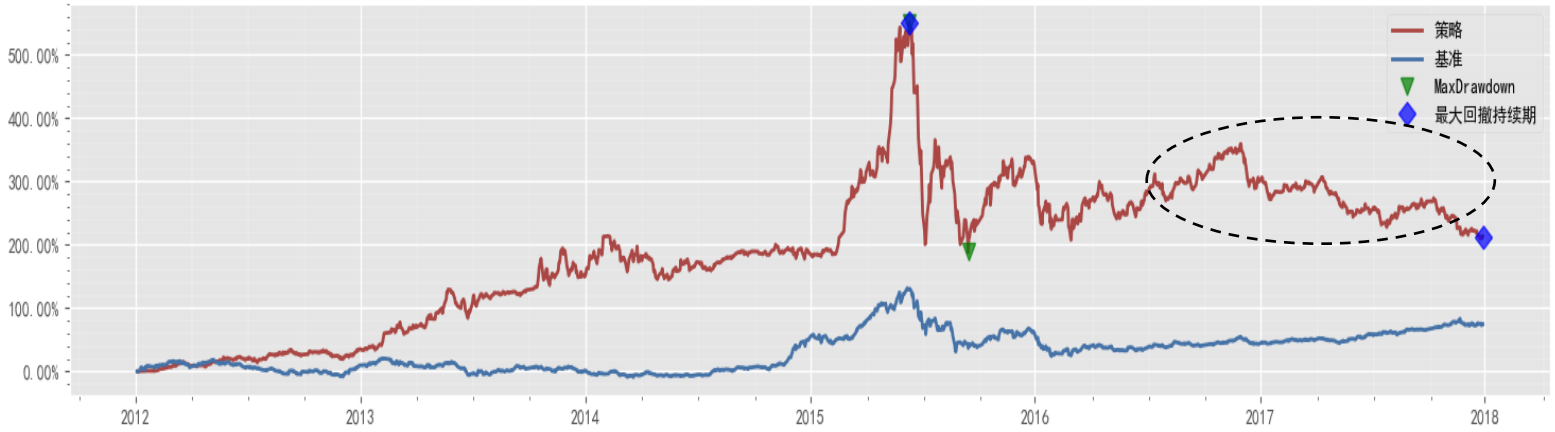


图 5：回测表现（threshold = 0.02）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 556.500% | 36.900% | 0.259 | 0.479 | 1.616 | 2.226 | 1.099 |
| **基准收益** | **基准年化收益** | **Volatility** | **MaxDrawdown** | **Tracking Error** | **Downside Risk** | **最大回撤/最大回撤持续期** |
| 74.500% | 9.700% | 0.187 | 39.000% | 0.193 | 0.136 | MaxDD 2015-06-12 00:00:00~2015-09-15 00:00:00, 95 days<br>MaxDDD 2015-06-12 00:00:00~2017-12-29 00:00:00, 931 days |

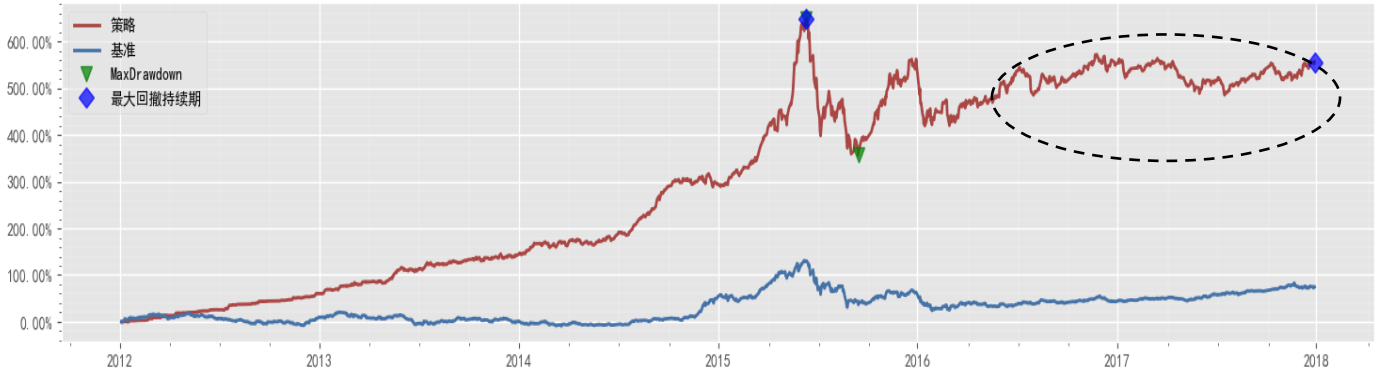


图 6：回测表现（threshold = 0.05）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 605.500% | 38.600% | 0.277 | 0.435 | 1.591 | 2.143 | 1.048 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.199 | 37.300% | 0.216 | 0.148 | MaxDD 2015-06-11 00:00:00~2015-09-15 00:00:00, 96 days<br>MaxDDD 2015-06-11 00:00:00~2017-12-29 00:00:00, 932 days |

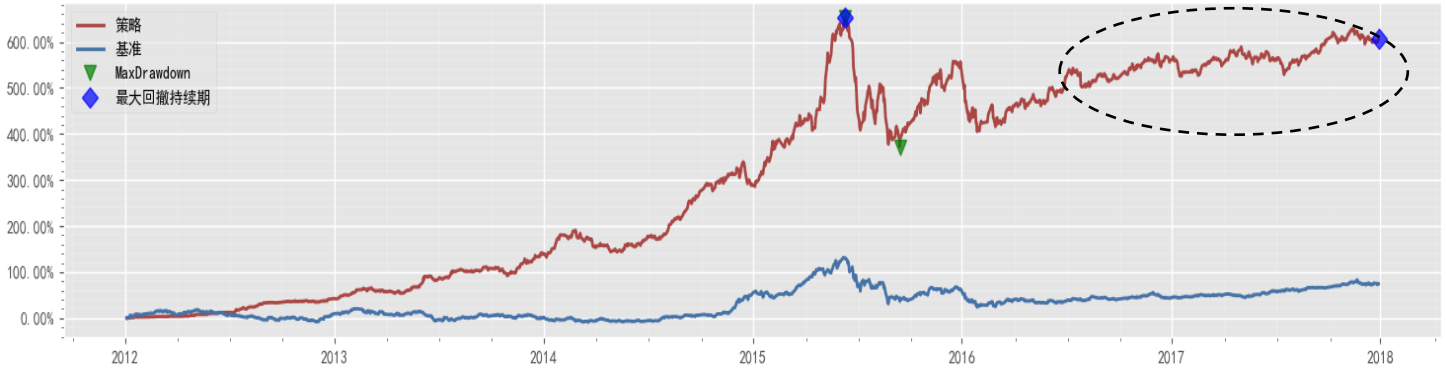


图 7：回测表现（threshold = 0.1）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 780.500% | 43.800% | 0.305 | 0.596 | 1.603 | 2.594 | 1.349 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.224 | 47.500% | 0.199 | 0.138 | MaxDD 2015-06-03 00:00:00~2015-09-15 00:00:00, 104 days<br>MaxDDD 2015-06-03 00:00:00~2017-12-29 00:00:00, 940 days |

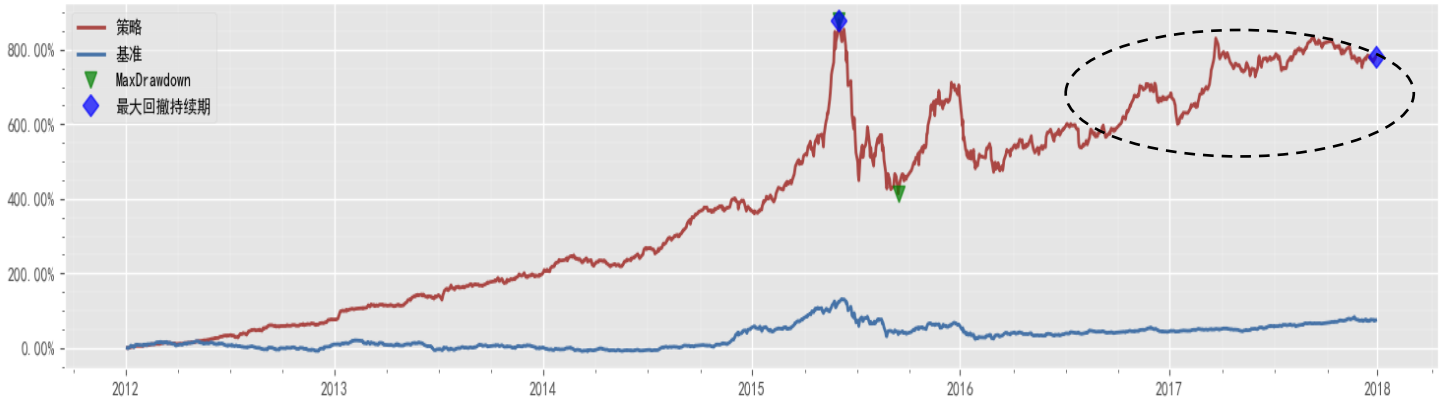


图 8：回测表现（threshold = 0.2）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 798.700% | 44.300% | 0.313 | 0.735 | 1.316 | 2.218 | 1.206 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.288 | 56.700% | 0.24 | 0.171 | MaxDD 2015-06-12 00:00:00~2015-09-02 00:00:00, 82 days<br>MaxDDD 2015-06-12 00:00:00~2017-12-29 00:00:00, 931 days |

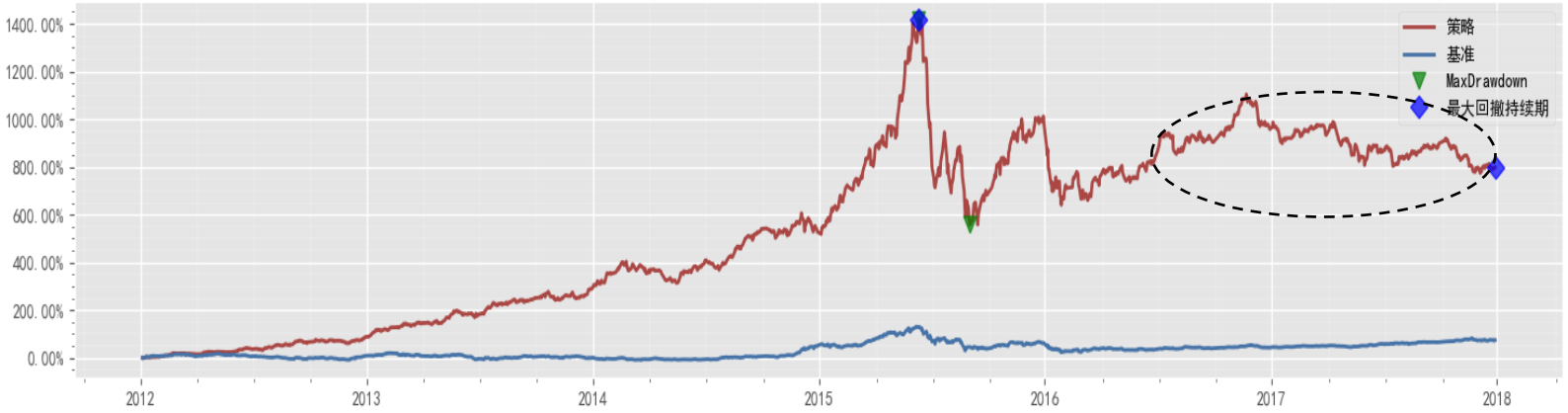


图 9：回测表现（threshold=0.3）

从表 5 的模型分类表现中可以看出， threshold=0.1 或 0.2 的模型分类表现最优。而从模型回测表现中看来，随着 Threshold 的增大，模型的回测收益逐渐增大，当 threshold=0.3 时模型的回测收益达到了 798%。然而可以注意到的是，当 threshold=0.3 时，模型在测试集时间段的表现并不突出，以此可以推断出 threshold=0.3 的模型高收益是因为训练集上过拟合，因此本研究不认为 0.3 为 threshold 的最佳取值。通过综合考虑回测收益，最大回撤，夏普比率，以及模型在测试集时间段上的表现，我们认为 threshold=0.2 是在目前取值集合中的最优取值（回测收益 780.5%，历史年化收益 43.800%，夏普比率 1.591，最大回撤 37%）

### 3.2 机器学习模型调优

确定了 threshold,相当于确定了模型的输入和输出，然后本研究利用 Sklearn 中的单变量特征选择的方法对每个因子和股票未来时刻的涨跌幅的相关性做了研究，具体结果如表 6 所示。发现因子和未来涨跌幅之间存在着显著关联，且关联企业影响力因子的关联度在所有因子中是非常显著的。我们同时可以发现，二级关联的相关性甚至要高于大多数一级关联，这可能是因为对于一家企业而言，其一级关联的企业数目可能是非常有限的，而二级关联的企业数目则明显多于一级关联的企业数目，这使得二级关联的影响力总和要大于一级关联。

| 债务下游 | 上游下游 | 集团上游 | K | 并购上游 | 下游股东 | RSV | 同行 | 股东债务 | 同行概念 |
|---|---|---|---|---|---|---|---|---|---|
| 0.045136 | 0.041721 | 0.040156 | 0.024870 | 0.023733 | 0.022855 | 0.022674 | 0.020394 | 0.017545 | 0.015693 |
| J | 集团债务 | BIAS_6 | 同行上游 | 并购投资 | 下游高管 | 下游上游 | 高管诉讼 | 上游产品 | 概念同行 |
| 0.015400 | 0.015278 | 0.014714 | 0.014665 | 0.014446 | 0.014297 | 0.014173 | 0.014093 | 0.014091 | 0.013968 |

表 6：Top10%相关性高的因子

在探究完输入输出相关性后，我们继续探究不同机器学习模型在分类上的表现，以此来选出最佳的分类模型。本研究综合考虑了不同类型的模型，诸如网络模型（深度信念网络（DNN））、回归模型（逻辑回归模型（Logistic Regression）,支持向量机（SVM））、树模型（随机森林(Random Forest)，梯度提升树(GBDT)，极端梯度提升树（XGboost））.针对于每

个模型，本研究采用格点法调参（GridSearch）的方式来确定模型的最优参数，本研究确定的模型最优参数以及其表现如表 7 和表 8 所示。

| | Precision(0) | Recall(0) | Precision(1) | Recall(1) |
|---|---|---|---|---|
| **DNN** | 0.98 | 1.00 | 0.90 | 0.06 |
| **Random Forest** | 0.99 | 1.00 | 0.79 | 0.48 |
| **GBDT** | 1.00 | 1.00 | 0.77 | 0.51 |
| **XGboost** | 1.00 | 1.00 | 0.77 | 0.53 |
| **LR** | 0.99 | 1.00 | 0.59 | 0.48 |

表 7：各模型的最优表现

| 模型 | 最佳参数 |
|---|---|
| **DNN** | 隐藏层数：3,节点数：（200，100，160），隐藏层激活函数（“sigmoid”,”relu”,”relu”）,drop_out=0.5 |
| **Random Forest** | n_estimators= 60,min_samples_split=4,max_depth=5,random_state=0 |
| **GBDT** | n estimator = 100, subsample = 1, learning rate = 0.1, min_samples_split = 2, max_depth = 3 |
| **XGboost** | max_depth=5, eta = 0.1, gamma = 0, min_child_weight = 6, n_estimators=150 |
| **LR** | C=5.0 |

表 8：各模型的最优参数

综合考虑预测正概率的精确率以及召回率，由于 DNN 较高的精确率表现以及 Random Forest 较好的综合表现。本研究选取 DNN 和 Random Forest 生成的模型进行回测，回测规则与之前相同。DNN 和 Random Forest 模型生成的回测曲线如图 10 和图 11 所示。

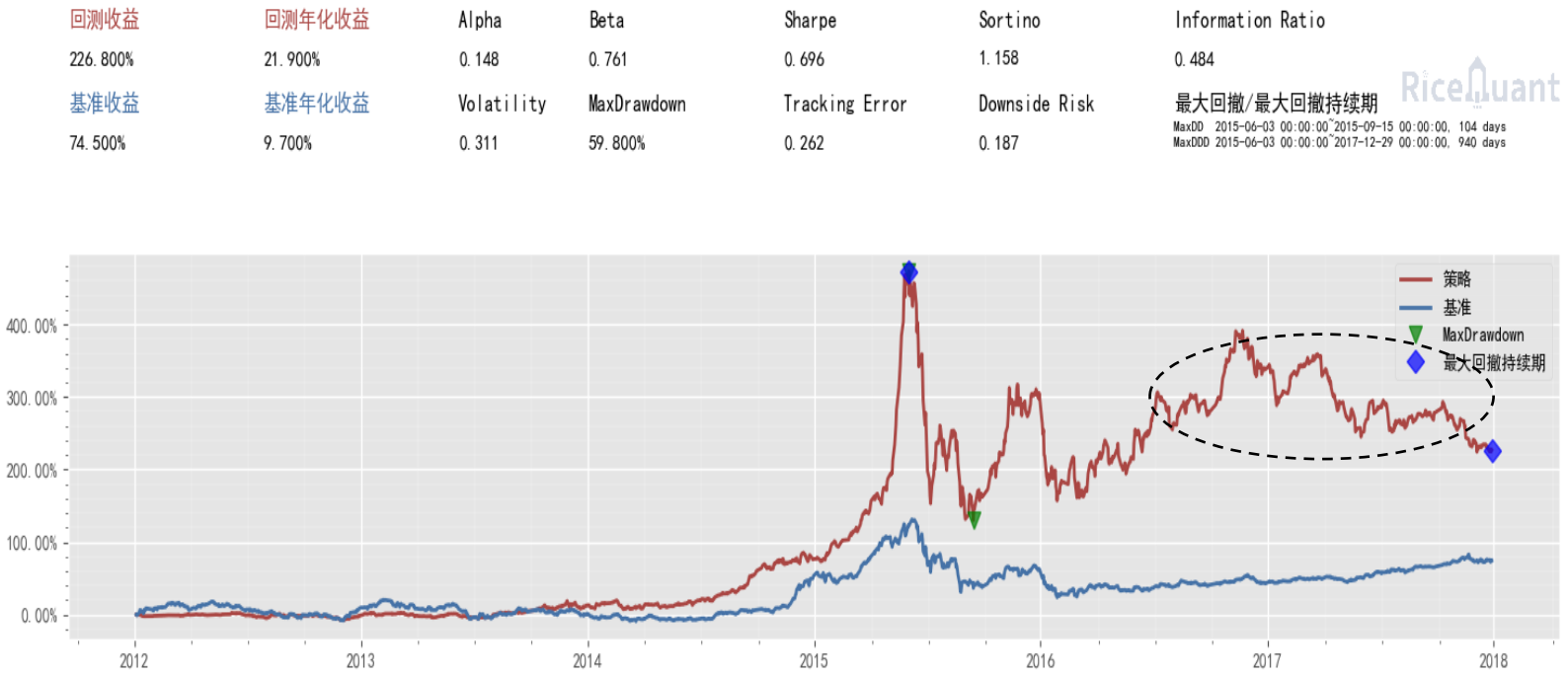


图 10：DNN 模型回测表现

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 780.500% | 43.800% | 0.305 | 0.596 | 1.603 | 2.594 | 1.349 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.224 | 47.500% | 0.199 | 0.138 | MaxDD 2015-06-03 00:00:00~2015-09-15 00:00:00, 104 days<br>MaxDDD 2015-06-03 00:00:00~2017-12-29 00:00:00, 940 days |

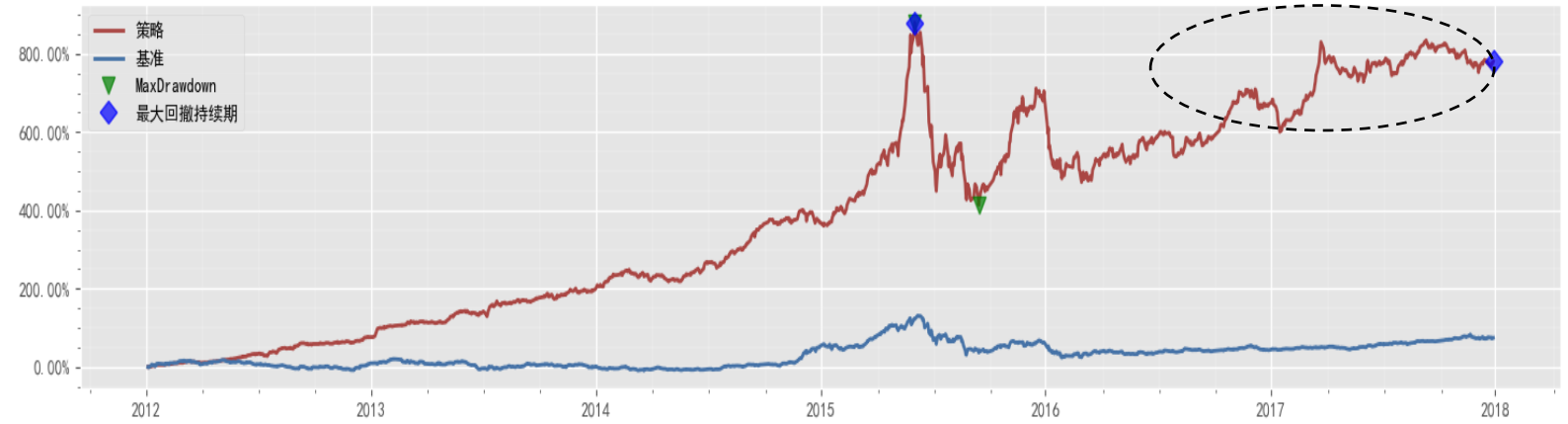


图 11：RF 模型回测表现

可以看出 RF 在测试集时间段上的表现要优于 DNN，DNN 存在着较为严重的过拟合现象，因此，我们选择 RF 为模型。

3.3 持股数调优

以上的模型都是我们基于每次只选排名第一的股票进行持仓，接下来我们要确定最优的持仓股数，在这里我们选每次的持仓股数为 1，2，3，并探究不同的持仓股数模型的回测表现。不同的持仓股数对应的回测结果如图 12，图 13，图 14 所示：

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 780.500% | 43.800% | 0.305 | 0.596 | 1.603 | 2.594 | 1.349 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.224 | 47.500% | 0.199 | 0.138 | MaxDD 2015-06-03 00:00:00~2015-09-15 00:00:00, 104 days<br>MaxDDD 2015-06-03 00:00:00~2017-12-29 00:00:00, 940 days |

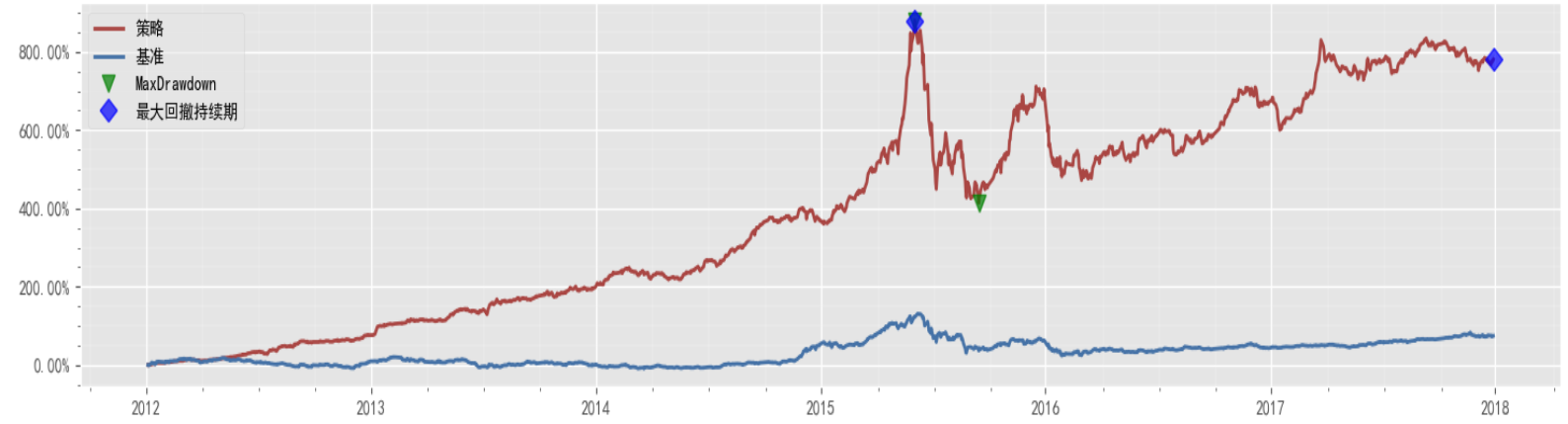


图 12：回测表现（持仓数为 1）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 1490.000% | 58.700% | 0.402 | 0.648 | 1.981 | 3.5 | 1.901 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.232 | 44.700% | 0.195 | 0.131 | MaxDD 2015-06-12 00:00:00~2015-09-15 00:00:00, 95 days<br>MaxDDD 2015-06-12 00:00:00~2017-12-29 00:00:00, 931 days |

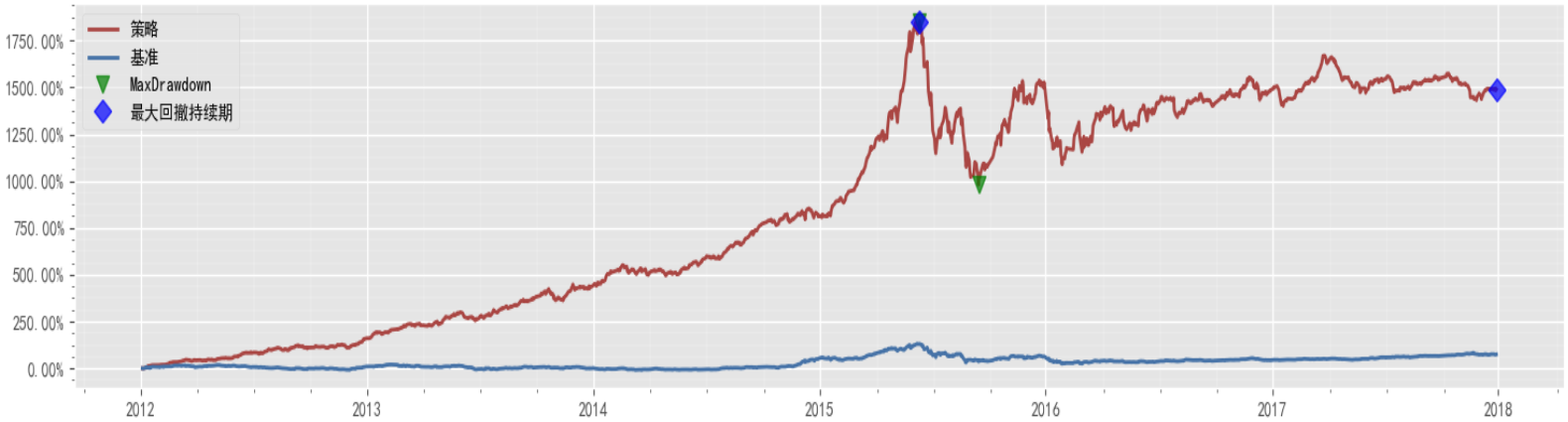


图 13：回测表现（持仓数为 2）

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 206.000% | 20.500% | 0.126 | 0.858 | 0.668 | 1.254 | 0.489 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.304 | 62.800% | 0.231 | 0.162 | MaxDD 2015-06-02 00:00:00~2017-12-27 00:00:00, 939 days<br>MaxDDD 2015-06-02 00:00:00~2017-12-29 00:00:00, 941 days |

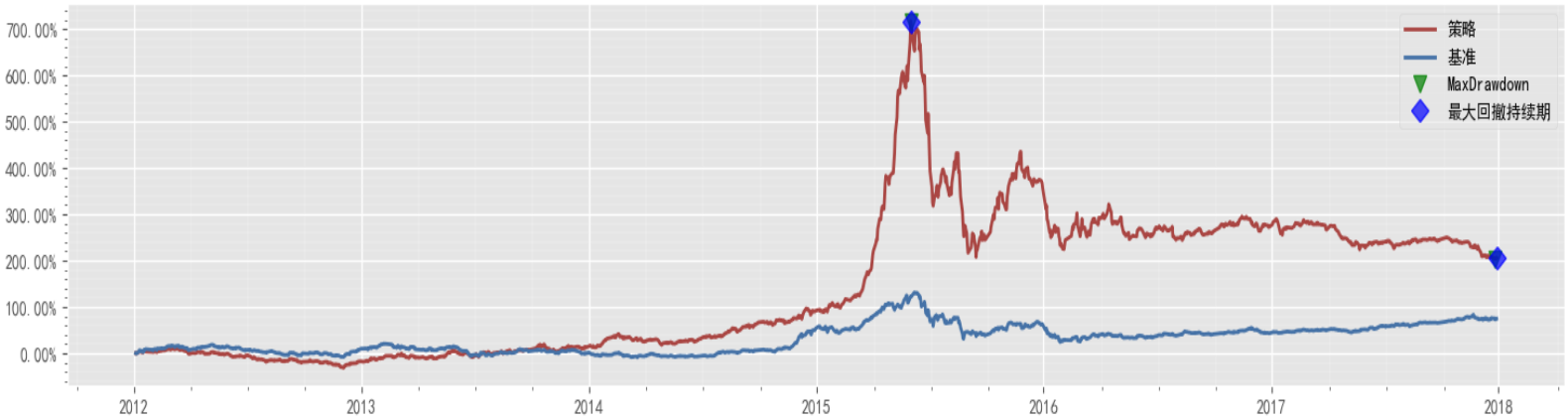


图 14：回测表现（持仓数为 3）

我们发现,当持仓数为 2 时，策略的回测效果最好，因此本研究将 2 作为每回新持仓的股数。

## 五、回测

在确定完模型以及所有参数后，我们的最终回测策略如下所示：利用随机森林模型预测股票涨幅达到 0.2 以上的正概率，选出排名前 2 的股票（若预测正类的股票数小于 2，则取符合要求的全部股票）来作为该时点新添的股票持仓。针对新持仓的股票，策略对每只股票

购进 1000 股，于此同时卖出上周持仓的股票。我们计算出的每时刻应该持有的股票存储在 strategy_list.txt 文件中。回测策略初始化金额为 100000。策略的回测结果如图 15 所示。策略的回测收益为 1490%，回测年化收益为 58.7%，Alpha 为 0.402，Beta 为 0.648，夏普比率为 1.981，索提诺比率为 3.5，信息比为 1.901，而相对应时间段的基础收益（沪深 300）为 74.5%，基准年化收益为 9.7%，回测周期为 2012 年 1 月 1 日至 2017 年 12 月 29 日，一共历时六年。

| 回测收益 | 回测年化收益 | Alpha | Beta | Sharpe | Sortino | Information Ratio |
|---|---|---|---|---|---|---|
| 1490.000% | 58.700% | 0.402 | 0.648 | 1.981 | 3.5 | 1.901 |
| 基准收益 | 基准年化收益 | Volatility | MaxDrawdown | Tracking Error | Downside Risk | 最大回撤/最大回撤持续期 |
| 74.500% | 9.700% | 0.232 | 44.700% | 0.195 | 0.131 | MaxDD 2015-06-12 00:00:00~2015-09-15 00:00:00, 95 days<br>MaxDDD 2015-06-12 00:00:00~2017-12-29 00:00:00, 931 days |

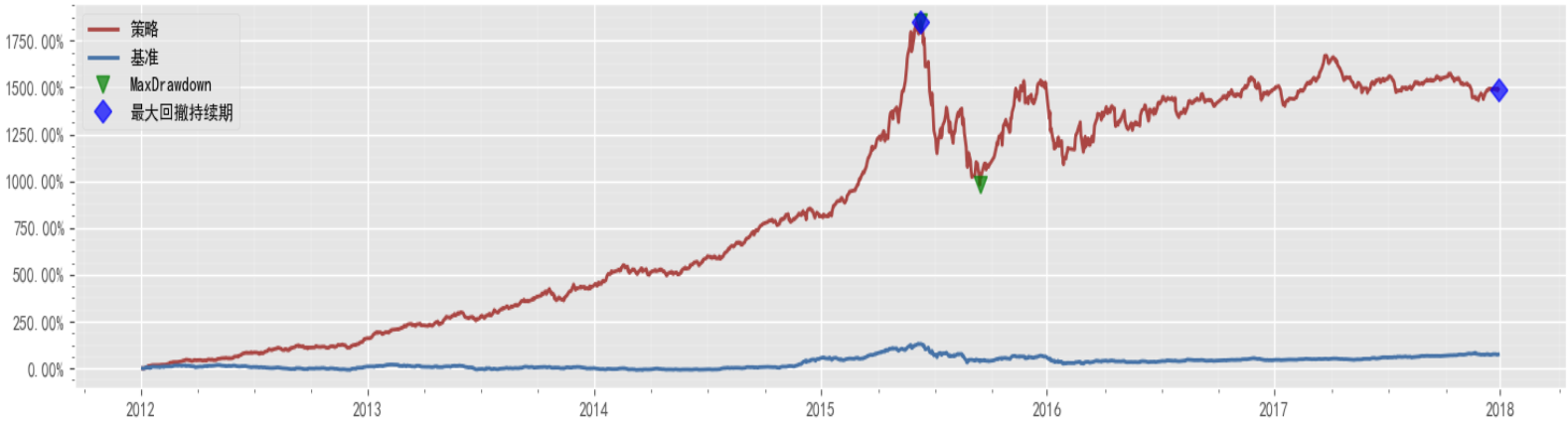


图 15：最终回测表现

## 六、模型的创新与改进

本研究在此前机器学习预测股价的文章上创新性地引入了关联企业影响力因子，关联企业影响力因子显著地提升了模型的分类表现，也使得输入能够更好地解释股价的中短期波动。此外，本研究通过对不同参数进行调整，发现有的参数对于回测收益的变化非常显著，例如之前用于确定模型输出的 threshold 参数。本研究通过取值不同的参数，找到模型表现最佳的参数。本研究同时还对比了不同的机器学习模型之间的分类效果，来挑选出最适合本研究分类情境的模型。

本研究将股票收益问题转化为分类问题的同时，为了扩大分类正确的收益以及减小分类错误的代价，因此本研究对模型的输入值做了创新的修改，来避免此前提到的少量分类错误会给收益带来严重影响。

此外，本研究还有着一定的未来改进方向，例如：

- 企业关系的更精细化度量：由于数据问题，本研究无法非常精确地确定出关联企业之间紧密度的大小，本研究只能选出相对而言比较紧密的几个企业。同样企业之间的股价互动会随着企业的基本面，受关注度等影响，受关注度越高的企业的股价波动很有可能会更快影响到其他的企业，因此更加精细地度量企业的关系，将是未来的一个重大发展方向。
- 集成学习模型：本研究中未考虑 stacking 下的集成模型，可能集成模型会取得比基本模型更好的分类效果。

## 参考文献：

[1] Miao, Q., Y. Meng and B. Zhang (2015). Chinese enterprise knowledge graph construction based on Linked Data. Semantic Computing (ICSC), 2015 IEEE International Conference on, IEEE.

[2] Hu, X., Tang, X., and Tang, F. 2017, July. Analysis of Investment Relationships Between Companies and Organizations Based on Knowledge Graph. In International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Springer, Cham. pp. 208-218.

[3]Market Segmentation and Cross-predictability of Returns

[4]Credit risk propagation along supply chains:Evidence from CDS market

[5]Inter-firm linkages and the weath effects of financial distress along the supply chain.

[6]Cao Q, Leggio K B, Schniederjans M J. A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market[J]. Computers & Operations Research, 2005, 32(10): 2499-2512.

[7]Fama E F, French K R. Common risk factors in the returns on stocks and bonds [J]. Journal of Financial Economics, 1993, 33(1):3-56.

[8]Olson D, Mossman C. Neural network forecasts of Canadian stock returns using accounting ratios[J]. International Journal of Forecasting, 2003, 19(3): 453-465.

[9]Kryzanowski L, Galler M, Wright D W. Using Artificial Neural Networks to Pick Stocks[J]. J.bras.patol.med.lab, 2005, 39(4):361-364.

[10]Feng F, He X, Wang X, et al. Temporal Relational Ranking for Stock Prediction[J]. arXiv preprint arXiv:1809.09441, 2018.

[11]Huang W, Nakamori Y, Wang S Y. Forecasting stock market movement direction with support vector machine[J]. Computers & Operations Research, 2005, 32(10):2513-2522.

[12]Hegazy O, Soliman O S, Salam M A. LSSVM-ABC Algorithm for Stock Price prediction[J]. International Journal of Computer Trends & Technology, 2014, 7(2).

[13]Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions[J]. European Journal of Operational Research, 2018, 270(2): 654-669.

[14] Bao W, Yue J, Rao Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory[J]. PloS one, 2017, 12(7): e0180944.

[15] Nguyen T H, Shirai K. Topic modeling based sentiment analysis on social media for stock market prediction[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015, 1: 1354-1364.

附录:

## 1. 股票价格提取部分

```
import tushare as ts
import pandas
with open("stock_list",encoding="utf-8") as f:#读取股票列表
    stock_list = eval(f.read())
for stock in stock_list:#提取股票价格
    Data = ts.get_k_data(stock,ktype='W')
    Data.to_csv("stock_price/"+stock+".csv")
```

## 2. 技术指标计算部分

```
#计算几类技术指标 KDJ,
import tushare
import os
import pandas as pd
from datetime import datetime
import numpy
stock_files = os.listdir("new_stock_price")
for stock_file in stock_files:
  try:
    code = stock_file.replace(".csv","")

    Data = pd.read_csv("new_stock_price/"+stock_file) #第 1 指标列为时间
    #--------计算 K,D,J 指标----------#
    Data["RSV"] = (Data['close'] - Data['low'])/(Data['high'] - Data['low'])*100 #计算 RSV
    K_list = []
    D_list = []
    J_list = []
    for i in range(0,len(Data)):
      if i == 0:
          K_list.append(50 * 2 / 3 + Data["RSV"][i] / 3)
          D_list.append(50 * 2 / 3 + K_list[i] / 3)
          J_list.append(3 * K_list[i] - 2 * D_list[i])
      else:
          K_list.append(K_list[i-1] * 2 / 3 + Data["RSV"][i] / 3)
          D_list.append(D_list[i-1] * 2 / 3 + K_list[i] / 3)
          J_list.append(3 * K_list[i] - 2 * D_list[i])
    Data["K"]=K_list
    Data["D"]=D_list
    Data["J"]=J_list
    #-------计算 MACD--------------#
    EMA_12_list = []
```

```
    for i in range(0,len(Data)):
      if i == 0:
          EMA_12_list.append(11* Data["close"][0]/ 13 + 2* Data["close"][i]/ 13)
      else:
          EMA_12_list.append(11 * EMA_12_list[i-1] / 13 + 2 * Data["close"][i] / 13)
    Data["EMA12"] = EMA_12_list
    EMA_26_list = []
    for i in range(0,len(Data)):
      if i == 0:
          EMA_26_list.append(25* Data["close"][0]/ 27 + 2* Data["close"][i]/ 27)
      else:
          EMA_26_list.append(25 * EMA_26_list[i-1] / 27 + 2 * Data["close"][i] / 27)
    Data["EMA26"] = EMA_26_list
    Data["DIF"] = Data["EMA12"] - Data["EMA26"]
    DEA_list = []
    for i in range(0,len(Data)):
      if i == 0:
          DEA_list.append(8* Data["DIF"][0]/ 10 + 2* Data["DIF"][i]/ 10)
      else:
          DEA_list.append(8 * DEA_list[i-1] / 10 + 2 * Data["DIF"][i] / 10)
    Data["DEA"] = DEA_list
    Data["MACD"] = 2*(Data["DIF"]-Data["DEA"])
    #----------计算 OBV-----------#
    #OBA_list = []
    # time_base = "2016-11-10"##why
    # time_index = Data[(Data["date"]==time_base)].index
    # OBA_list.append(0)
    # for i in range(1,len(Data)):
    #   if Data["close"][i] > Data["close"][i-1]:
    #     OBA_list.append(Data["volume"][i] + Data["volume"])
    #   if Data["close"][i] < Data["close"][i-1]:
    #     OBA_list.append(Data["volume"] - Data["volume"][i])
    Data["OBA"] = ((Data['close']-Data['low'])-(Data['high']-
Data['close']))/(Data['high']-Data['low'])*Data["volume"]
    #----------计算 RSI-----------#
    #相对强弱指标
    #国内单边做多的股市 : 强弱指标值一 般分布在 20 — 80; 80-100 极强 卖出; 50-80 强 买入;
20-50 弱 观望; 0-20 极弱 买入。
    delta = Data["close"].diff()
    dUp, dDown = delta.copy(), delta.copy()
    dUp[dUp < 0] = 0
    dDown[dDown > 0] = 0
    #sliding window in 14
    #dUp
```

```
    Rolup = dUp.rolling(window=14,min_periods=1).mean()
    Roldown = dDown.rolling(window=14,min_periods=1).mean()
    RS = Rolup / Roldown
    RSI = 100*RS/(1+RS)
    Data['RSI'] = RSI

    #---------计算 PSY----------#
    #PSY=N 日内的上涨天数 /N×100%
    # N = 10#just assume, it doesn't say n = what...
    PSY = dUp.rolling(window=10,min_periods=1).apply(lambda x:
numpy.count_nonzero((x>0)))#,raw = False)
    Data['PSY'] = PSY

    #---------计算 AR----------#
    #人气指标 (AR)和意愿指标 (BR)两个指标构成
    #N 日 AR=(N 日内( H-O)之和除以 N 日内( O-L)之和 )*100 其中，H 为当日最高价，L 为当日最低
价，O 为当日开盘价，N 为 设定的时间参数，一般原始参数日设定为 26 日
    #N = 26
    numo = (Data['high']-Data['open']).rolling(window=26,min_periods=1).sum()
    deno = (Data['open']-Data['low']).rolling(window=26,min_periods=1).sum()
    AR = numo/deno*100
    Data['AR'] = AR
    #N 日 BR=N 日内(H-CY)之和除以 N 日内(CY-L)之和*100
    #C_y close price of previous day

    #---------计算 BR----------#
    Cy = Data['close'].shift(1)
    Cy[0] = Cy[1] #处理因为是 yesterday 的 close 导致的 0 is NAN
    numo1 = (Data['high']-Cy).rolling(window=26,min_periods=1).sum()
    deno1 = (Cy-Data['low']).rolling(window=26,min_periods=1).sum()
    BR = numo1/deno1
    Data['BR'] = BR
    #---------计算 CR----------#
    M = (2*Data['close']+Data['high']+Data['low'])/4
    Ym = M.shift(1)
    #Ym[0] = Ym[1] #处理因为是 yesterday 的 middle 导致的 0 index is NAN
    #N = 26
    P1 = (Data['high']-Ym).rolling(window = 26,min_periods=1).sum()
    P2 = (Ym-Data['low']).rolling(window = 26,min_periods=1).sum()
    CR = P1/P2*100
    CR[CR<0] = 0
    Data['CR'] = CR

    #---------计算 BIAS----------#
```

```
    #BIAS=[( 当日收盘价 -N 日平均价 )/N 日平均价 ]*100%
    avg = (((Data['open']+Data['close'])/2).rolling(window = 6,min_periods=1).sum())/6
    BIAS_6 = (Data['close']-avg)/avg
    Data['BIAS_6'] = BIAS_6
    avg = (((Data['open']+Data['close'])/2).rolling(window = 12,min_periods=1).sum())/12
    BIAS_12 = (Data['close']-avg)/avg
    Data['BIAS_12'] = BIAS_12
    avg = (((Data['open']+Data['close'])/2).rolling(window = 24,min_periods=1).sum())/24
    BIAS_24 = (Data['close']-avg)/avg
    Data['BIAS_24'] = BIAS_24

    #----------计算 CCI--------------#
    TP = (Data['high']+Data['low']+Data['close'])/3
    MA = Data['close'].rolling(window = 14,min_periods=1).sum()/14
    MD = (MA-Data['close']).rolling(window = 14,min_periods=1).sum()/14
    CCI = (TP - MA)/MD/0.015
    Data['CCI'] = CCI
    #----------计算 WR---------------#
    n_days = 10
    lowest = Data['low'].rolling(min_periods=1, window=n_days).min()
    highest = Data['high'].rolling(min_periods=1, window=n_days).max()
    WR = 100*(highest-Data['close'])/(highest-lowest)
    Data['WR'] = WR
    #----------计算 TRIX---------------#
    #1 TR=收盘价的 N 日指数移动平均值
    #2TRIX=(TR-昨日 TR)/昨日 TR*100
    #3MATRIX=TRIX 的 M 日简单移动平均
    #4 参数 N 设为 12，参数 M 设为 20。

    TR = Data['close'].ewm(span=12, min_periods=1).mean()
    TRy = TR.shift(1)
    TRIX = (TR-TRy)/TRy*100
    MATRIX = TRIX.rolling(window=20, min_periods=1).mean()
    Data['M_TRIX']= MATRIX
    Data.to_csv(stock_file,encoding="gbk")
  except:
    print(code,"not work")
```

## 3. 关联企业影响因子计算部分

```
import os
import pandas as pd
import numpy as np
from collections import Counter
```

```
stock_files = os.listdir("stock_price")

with open("stock_graph",encoding="gbk") as f:#读取股票一级关系集合
    stock_graph = eval(f.read())
with open("new_stock_graph",encoding="gbk") as f:#读取股票二级关系集合
    stock_graph_2 = eval(f.read())
with open("count",encoding="gbk") as f:
    count = eval(f.read())

#---将两个字典合并---#
dic={}
for code in stock_graph.keys():
    dic[code] = {**stock_graph[code], **stock_graph_2[code]}
stock_graph = dic
print(stock_graph.keys())


Total_stock = {}
basic_column  =["open","close","high","low","volume","code","pct_change"]
for stock in stock_files:
    try:
        code = stock[0:9]
        Total_stock[code] = pd.read_csv("stock_price/"+stock).set_index("date")
        Total_stock[code].rename(columns=lambda x: x+code,inplace = True)
    except:
        print("fail",stock)
for code in Total_stock.keys():
  print(code)
  try:
    relevants = []
    for relation in stock_graph[code].keys():
        relevants = relevants + stock_graph[code][relation]
    relevants.append(code)
    relevants = list(set(relevants))
    Total = pd.concat([Total_stock[relevant] for relevant in relevants],
join_axes=[Total_stock[code].index],axis=1)
    Total.fillna(0,inplace=True)
    for relation in count.keys():
        Total[relation+code] = 0
    relevants.remove(code)
    for relation in stock_graph[code].keys():
        relevants = stock_graph[code][relation]
        list_sum = np.sum([Total["pct_change"+relevant] for relevant in
relevants],axis=0)
```

```
            Total[relation+code] = list_sum
        indicator_list = basic_column + list(count.keys())


        Data = Total[list(map(lambda x:x+code,indicator_list))]

        Data.rename(columns = lambda x:x.replace(code,""),inplace =True)
        Data.to_csv("new_stock_price/"+code+".csv")
    except:
        print(code,"not work")
```

## 4. 策略部分（strategy_lis.txt 存储对应时间的持仓信息）

```
# 可以自己 import 我们平台支持的第三方 python 模块，比如 pandas、numpy 等。
from rqalpha.api import *
from datetime import date
from datetime import time
from datetime import datetime
from datetime import timedelta

# 在这个方法中编写任何的初始化逻辑。context 对象将会在你的算法策略的任何方法之间做传递。
with open("mylist") as f:
    mylist = eval(f.read())


mydict={
    i['date']:i['top5'] for i in mylist
    }

def init(context):
    # 在 context 中保存全局变量
    #context.s1 = "000001.XSHE"
    # 实时打印日志
    context.count = 0
    logger.info("RunInfo: {}".format(context.run_info))


# before_trading 此函数会在每天策略交易开始前被调用，当天只会被调用一次
def before_trading(context):
    pass


# 你选择的证券的数据更新将会触发此段逻辑，例如日或分钟历史数据切片或者是实时数据切片更新
def handle_bar(context, bar_dict):
```

```
    # 开始编写你的主要的算法逻辑

    # bar_dict[order_book_id] 可以拿到某个证券的 bar 信息
    # context.portfolio 可以拿到现在的投资组合信息

    # 使用 order_shares(id_or_ins, amount)方法进行落单
    #if(mylist["date"] in context.now)
    #if 这个 context.now == 现在的 date 了
    #就进行买
    currdate = context.now.strftime("%Y-%m-%d")
    currwday = context.now.strftime("%A")
    lastweek = (context.now - timedelta(days=7)).strftime("%Y-%m-%d")
    #print(lastweek)
    if currwday == "Friday":
        try:
            if context.count>0:
                lastweek_share = mydict[lastweek]
                for l in range(len(lastweek_share)):
                   try:
                    order_shares(str(lastweek_share[l]), -1000)
                    print("selling stock"+str(lastweek_share[l]))
                   except:
                       print("not this stock")
                logger.info(lastweek_share)

            today_share = mydict[currdate]
            for s in range(len(today_share)):
              try:
                order_shares(str(today_share[s]),1000)
                print("buying stock"+str(today_share[s]))
              except:
                  print("CAN'T BUT B:"+today_share[s])
            logger.info(today_share)



            context.count+=1
        except:
            print("time wrong")

# after_trading 函数会在每天交易结束后被调用，当天只会被调用一次
def after_trading(context):
    pass
```

## 5. 运行策略部分

```
# run_file_demo
from rqalpha import run_file

config = {
  "base": {
    "data_bundle_path":"E:\\Strategy\\bundle",
    "start_date": "2012-01-01",
    "end_date": "2018-01-01",
    "benchmark": "000300.XSHG",
    "accounts": {
        "stock": 100000
    }
  },
  "extra": {
    "log_level": "verbose",
  },
  "mod": {
    "sys_analyser": {
      "enabled": True,
      "plot": True
    }
  }
}

strategy_file_path = "backtest.py"

run_file(strategy_file_path, config)
```