基于总体规模与投资效率指标的综合评价与量化投资分析

摘要

随着我国证券市场的不断壮大，证券及证券投资在社会经济生活中的地位也越来越重要，上市公司的数量也不断增加，目前在上海证券交易所和深圳证券交易所上市交易的公司已经达到3700多家。投资者面对如此众多的不同行业、背景的股票，除了基本政策面分析外，还希望对这些股票的基本面及市场交易机会进行客观理性地评估。

本文利用数据库收集的股票交易数据计算了上市公司的营业收入、营业利润、利润总额、净利润、资产总计、固定资产净额，投资效率指标包括净资产收益率、每股净资产、每股资本公积、每股收益，一共10个指标等，通过计算股票交易的技术分析指标，利用数据挖掘模型来预测下一个交易日股票价格涨跌方向，并基于预测的结果设计量化投资策略，从而为技术分析爱好者提供一定的实践价值为了最大化的获得投资收益，得出每个行业的综合排名得分，获得最优的量化投资组合。

**关键词：**数据库 综合得分排名 量化投资 技术分析指标

**目录**

[1 前言](#_Toc17454)

[1.1 研究背景](#_Toc2205)

[1.2 研究意义](#_Toc2205)

[1.3 目标及实现思路 3](#_Toc28268)

[2 基于总体规模与投资效率指标的综合评价 4](#_Toc11036)

[2.1指标选择 5](#_Toc21601)

[2.2 获取数据 5](#_Toc32132)

[2.3 数据处理 6](#_Toc2672)

[2.3.1 筛选指标值大于0的数据 6](#_Toc9507)

[2.3.2 去掉NAN值 6](#_Toc16732)

[2.3.3 数据标准化 6](#_Toc21928)

[2.4 主成分分析 7](#_Toc5133)

[2.5 综合排名 7](#_Toc1138)

[2.5.1 计算综合得分 7](#_Toc6193)

[2.5.2 整理排名结果 7](#_Toc14109)

**[3 技术分析指标选择与计算](#_Toc28946)** [9](#_Toc28946)

[3.1 移动平均数 10](#_Toc20344)

[3.2 指数平滑异同平均线 10](#_Toc24105)

[3.3 随机指标 11](#_Toc2137)

[3.4 能量潮指标 12](#_Toc19248)

[3.5 乖离率指标 12](#_Toc20344)

[3.6 涨跌趋势指标 13](#_Toc24105)

[3.7 相对强弱指标指标 14](#_Toc2137)

**4量化投资模型与策略实现 ……………………………………………...……………14**

[4.1 投资组合构建 15](#_Toc24105)

[4.2 基于逻辑回归的量化投资策略实现 17](#_Toc2137)

[5 GUI可视化系统设计与实现 17](#_Toc12775)

[5.1 需求分析 17](#_Toc21945)

[5.2 系统设计 21](#_Toc29008)

[5.3 系统实现 26](#_Toc7321)

[5.4 生产独立运行的EXE 27](#_Toc9984)

[5.5 系统演示 32](#_Toc4017)

[6 分析及总结 33](#_Toc30685)

[7 参考文献 34](#_Toc27342)

[8 程序附录 35](#_Toc6417)

# 前言

## 研究背景

随着我国证券市场的不断壮大，证券及证券投资在社会经济生活中的地位也越来越重要，上市公司的数量也不断增加，目前在上海证券交易所和深圳证券交易所上市交易的公司已经达到3700多家。投资者面对如此众多的不同行业、背景的股票，除了基本政策面分析外，还希望对这些股票的基本面及市场交易机会进行客观理性地评估。传统的基本面分析投资方法，主要是通过实地调研、阅读公司投资及经营方面的公告、分析研究财务报表等手段找到优质的上市公司并进行投资。在上市公司数量较少时，传统的基本面分析方法不失为一种有效的方法。然而，在庞大的上市公司数量及其数据面前，传统的基本面分析方法具有很大的局限性：一方面，在如此大量的上市公司数据面前，我们无法及时完成分析，也更难找出优质的上市公司；另一方面，在信息高度发达的大数据时代，信息更新非常快，我们更难以应接。因此，基于数量化的投资分析方法，即量化投资应运而生。所谓量化投资就是采用计算机技术及数据挖掘模型，实现自己的投资理念或投资方法的一种过程。量化投资分析方法能够帮助我们快速挖掘并分析数据，从而找到我们需要的信息，这种技能已经成为投资界人士所推崇的技能。本案首先通过财务报表及财务指标数据，采用数量化的分析方法，对上市公司基本面情况进行综合评价，从而选出质地较好的上市公司；其次，以选出的上市公司发行的A股股票作为研究对象，通过计算股票交易的技术分析指标，利用数据挖掘模型预测下一个交易日股票收盘价较开盘价的涨跌方向；最后，基于预测的结果设计量化投资策略并进行实证检验。

## 研究意义

在投资市场中，量力而行的投资和冷静的头脑是两个必备的因素，投资市场的发展有它自身的规律，掌握这两个必备的要点会让我们的投资获得理想的效果。

量化投资的前提是有理性的决策，它能够利用市场的规律赚取超额的收益。和一般的投资不同的是，量化投资是通过量化模型进行交易，投资者按照提前划定好的程序进行投资，不会因为情绪的变化而变化，我觉得这是量化投资最大的优势。

在投资前，我们都会先制定一个投资策略，看自己的投资方案是不是合理的，确定方案后形成一个量化投资的模型，综合以前投资历史回测和风险考量指标进行评测，进行三个月到半年的虚拟交易，如果结果和历史回测以及其它预期的结果一致，则这个量化投资的模型将得到正式的确认，最终这个模型将会上线进行交易运行。

一旦量化投资的模型开始运行，将充分遵照模型的指令进行投资，降低情绪影响，克服人性贪婪恐惧的弱点，做到有计划、有原则、有纪律地进行投资。一般而言，通过这样的流程制定的量化投资策略能够理性决策，获取市场非理性的收益，大概率战胜市场。

在投资的过程中，量化投资不仅排除了投资者的主观因素影响，并且还能达到更精准的效果，以精确的值达到一个最优结果，这也是量化投资的第二个优势。量化投资则有一套完整的逻辑和规则，可以进行有效的评价和识别。从这个意义上来说，量化投资采用量化工具将主动投资逻辑规范化，能够带来规范化的收益。

量化投资可以对海量信息进行处理，现在人脑是永远不可能记住这么多信息的，并且对海量数据之间的关联不可能做出迅速判别，而电脑可以做到。从发展趋势来看，量化模型获取的超额收益是非常显著的。通过数学验证建立模型，形成一种投资模式，达到大概率的成功

## 目标及实现思路

主要目标是通过计算股票交易的技术分析指标，利用数据挖掘模型来预测下一个交易日股票价格涨跌方向，并基于预测的结果设计量化投资策略，从而为技术分析爱好者提供一定的实践价值。在技术分析指选择方面，主要选择趋势型、超买超卖型、人气型等指标，包括5日、10日、20日移动平均线指标MA，指数平滑异同平均线指标MACD，随机指标K、D、J，6日、12日、20日相对强弱指标RSI，5日、10日、20日乖离率指标BIAS、能量潮指标OBV等，并将这些指标作为解释变量（自变量）。被解释变量（因变量）为下一个交易日的股票价格涨跌方向，上涨用+1表示，否则为-1，是一种分类型变量。以一定的计算周期计算其解释变量和被解释变量，作为训练样本，以其后的一定周期计算其解释变量，作为测试样本，并预测其涨跌方向（即被解释变量），最后根据预测的结果设计量化投资策略。这里选择的模型包括支持向量机模型、神经网络模型和逻辑回归模型，并对不同模型的效果进行了分析，从而做出最佳的选择。具体的实现思路及计算流程如下图所示。

股票价格涨跌趋势预测

技术分析指标选择及计算（X）：MA5、MA10、MA20、MACD、K、D、J、RSI6、RSI12、RSI20、BIAS5、BIAS10、BIAS20、OBV；预测指标计算（Y）：下一交易日收盘价-当日收盘价>0记为+1，<0记为-1，=0记为0。注：均采用函数的形式定义

以2015年的交易数据为例进行计算，并作训练样本和测试样本的划分。训练样本区间：2015-01-01~11-30；测试样本区间：2015-12-01~12-31

模型求解：

逻辑回归模型，获得模型的准确率、预测结果及预测准确率，并根据预测结果计算其收益率（如果预测结果为+1，则以当天收盘价买入，下一个交易日收盘价卖出，计算其收益率）

以基于总体规模与投资效率指标的综合评价获得2015年综合排名前10的股票构建投资组合，取2015年的交易数据为例进行计算，并作训练样本和测试样本的划分训练样本区间：2015-01-01~11-30测试样本区间：2015-12-01~12-31

分析投资组合中每个股票的收益率和组合的总收益率，并对分析模型（逻辑回归模型）进行评价。

# 基于总体规模与投资效率指标的综合评价

上市公司总体规模体现了公司的整体竞争能力、市场抗风险能力和影响力。总体规模较大的上市公司在市场上有其优势。除此之外我们还需要考虑其投资效率，如果投资效率低下，那么其优势也许就不存在了。下面主要选择反映公司总体规模和投资效率方面的财务数据及财务指标，利用主成分分析模型进行综合评价。

## 指标选择

我们获取的总体规模指标包括上市公司的营业收入、营业利润、利润总额、净利润、资产总计、固定资产净额，投资效率指标包括净资产收益率、每股净资产、每股资本公积、每股收益，一共10个指标。数据来源于国泰安CSMAR数据库，具体信息如表2-1所示。

表2-1 上市公司总体规模与投资效率指标

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段中文名称 | 字段说明 |
| B001101000 | 营业收入 | 企业经营过程中确认的营业收入 |
| B001300000 | 营业利润 | 与经营业务有关的利润 |
| B001000000 | 利润总额 | 公司实现的利润总额 |
| B002000000 | 净利润 | 公司实现的净利润 |
| A001000000 | 资产总计 | 资产各项目之总计 |
| A001212000 | 固定资产 | 固定资产原价 |
| F050501B | 净资产收益率 | 净利润/股东权益余额 |
| F091001A | 每股净资产 | 所有者权益合计期末值/实收资本期末值 |
| F091301A | 每股资本公积 | 资本公积期末值/实收资本期末值 |
| F090101B | 每股收益 | 净利润本期值/实收资本期末值 |

## 获取数据

数据来源于国泰安CSMAR数据库，得到的数据如下图2-1：

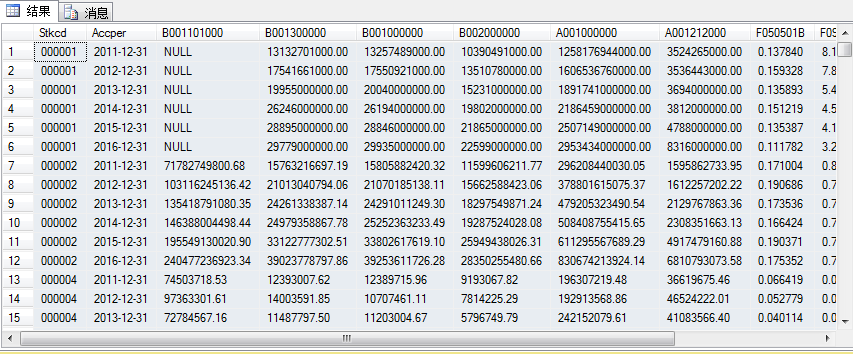


图2-1上市公司总体规模与投资效率指标

## 数据处理

### 筛选指标值大于0的数据

对上市公司评价，首先是选择质地比较好的公司，指标值小于0的公司可能存在公司资产为负值或者利润为负值等问题，这类公司首先排除在外。

### 去掉NAN值

NAN值即空值，存在指标取值缺失的公司也建议排除在外。

### 数据标准化

指标的单位存在不统一或者存在有些指标的取值很大、有些指标的取值很小的情况，因此需要对指标数据做标准化处理。

计算流程及思路如下。

（1）读取数据，其中第0列为标识列（股票代码），示例代码如下：

import pandas as pd

data=pd.read\_excel('Data.xlsx')

（2）筛选指标值大于0的数据以及去掉nan值，示例代码如下：

data=data[data>0]

data=data.dropna()

（3）数据标准化，注意标准化的数据需要去掉第0列（股票代码，标识列），这里数据标准化方法采用均值-方差法，示例代码如下：

from sklearn.preprocessing import StandardScaler

X=data.iloc[:,1:]

scaler = StandardScaler()

scaler.fit(X)

X=scaler.transform(X)

## 主成分分

对标准化之后的指标数据X做主成分分析，提取其主成分，要求累计贡献率在95%以上。其示例代码如下：

from sklearn.decomposition import PCA

pca=PCA(n\_components=0.95) #累计贡献率为95%

Y=pca.fit\_transform(X) #满足累计贡献率为95%的主成分数据

gxl=pca.explained\_variance\_ratio\_ #贡献率

通过主成分分析，可以获得其主成分，接下来就可以根据获得的主成分计算每个上市公司的综合得分了。根据综合得分，可以获得上市公司的综合排名。

## 综合排名

### 2.4.1 计算综合得分

综合得分等于提取的各个主成分与其贡献率的加权求和，示例代码如下：

import numpy as np

F=np.zeros((len(Y))) #预定义综合得分数组F

for i in range(len(gxl)):

f=Y[:,i]\*gxl[i] #第i个主成分与第i个主成分贡献率的乘积

F=F+f #数组累积求和

### 2.4.2 整理排名结果

为了方便进行排名，采用序列作为排名结果存储数据结构。排名包括两种方式，一种index为股票代码，方面后续计算收益率；一种index为股票中文简称，方便查看其排名结果。

第1种方式示例代码如下：

fs1=pd.Series(F,index=data['ts\_code'].values) #构建序列,值为综合得分F,index为股票代码

Fscore1=fs1.sort\_values(ascending=False) #结果排名,降序

第2种方式如下：

首先获取主成分分析指标数据对应的上市公司名称，可以通过data数据（经过预处理的财务指标数据）中的股票代码关联股票基本信息表（stkcode.xlsx）筛选获得。stkcode.xlsx数据表详细信息如表2-2所示。

表2-2 股票基本信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ts\_code** | **symbol** | **name** | **area** | **industry** |
| 000001.SZ | 000001 | 平安银行 | 深圳 | 银行 |
| 000002.SZ | 000002 | 万科A | 深圳 | 全国地产 |
| 000004.SZ | 000004 | 国农科技 | 深圳 | 生物制药 |
| 000005.SZ | 000005 | 世纪星源 | 深圳 | 环境保护 |
| 000006.SZ | 000006 | 深振业A | 深圳 | 区域地产 |
| 000007.SZ | 000007 | 全新好 | 深圳 | 酒店餐饮 |
| 000008.SZ | 000008 | 神州高铁 | 北京 | 运输设备 |

其中字段依次表示股票代码、股票代码、股票名称、地区、行业。示例代码如下：

stk=pd.read\_excel('stkcode.xlsx')

stk=pd.Series(stk['name'].values,index=stk['ts\_code'].values)

stk1=stk[data['ts\_code'].values] #主成分分析指标数据对应的上市公司名称

其次，以综合得分F为值，上市公司名称作为索引（index），构建序列，并按值做降序排序，以观察其排名结果，示例代码如下：

fs2=pd.Series(F,index=Co1.values)

Fscore2=fs2.sort\_values(ascending=False)

结果如图2-2、图2-3所示：

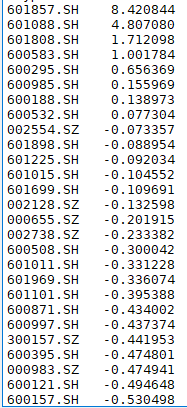
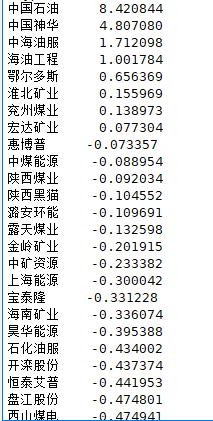
 

图2-2 第一种排名 图2-3第二种排名

# 技术分析指标选择与计算

关于股价研究的技术指标有很多，常用的技术指标概括起来分为四类：趋势型、超买超卖型、人气型以及大势型。选取指标应遵循以下几个原：

1.全面性

不同的指标其计算原理、衡量效果不同，为真实有效的反映股票价格变动情况，在选择指标时要综合考虑影响股价的成交量、成交价格等多方面因素。

2.简洁性

技术指标的选择要全面，但不能过于繁杂，抓住主要情况，深入浅出。

3.可操作性

从指标数据方面来说，其数据的来源要及时、可靠，没有数据只有理论模型就没有实用价值，失去了研究意义，数据不可靠会影响最终预测结果。从指标应用原则来说，一般在分析过程中可根据指标自身波动情况、指标曲线与K线走势是否背离、同一指标不同参数值曲线交叉三种情况进行预测，有些情况识别比较困难，考虑到普通投资者，应避免复杂。

根据以上指标选取原则，主要选取了6种在中国证券交易市场上比较流行且有效的技术指标：趋势型有 (移动平均线)、 (指数平滑异同平均线)；超买超卖型有 (随机指标)、 (相对强弱指标)、 (乖离率指标)；人气型的有 (能量潮指标)。这里没有选取大势型指标 ,而是以某个指数作为近似代替（研究的股票均取自相应指数样本股）。技术指标计算基于原始数据进行，其意义以及计算公将在下面详细介绍。

## 移动平均线

移动平均线（MA）就是将某一定时期的收盘价之和除以该周期，按时间的长短可以分为长期、中期、短期3种。移动平均线可以反映出价格走势。

计算公式为：



。Python计算移动平均的命

令为：pd.rolling\_mean(P，n)。

其中，P为价格序列值，n为周期数。例如，计算5日移动平均为：

pd.rolling\_mean(P,5)

## 指数平滑异同平均线

指数平滑异同平均线（MACD）是在移动平均线的基础上发展而成的，它利用两条不同速度（一条变动速率较快的短期移动平均线，一条变动速度较慢的长期移动平均线）的指数平滑移动平均线来计算二者之间的差别状况（DIF），作为研判行情的基础，然后再计算出的9日平滑移动平均线，即线。

计算公式为：









Python计算指数平滑移动平均的命令为：pd.ewma(P, n)。其中，P为价格序列值，n为周期数。例如，计算12日、26日指数平滑移动平均为：

Z12=pd.ewma(P, 12)

Z26=pd.ewma(P, 26)

则DIF、DEA、MACD计算算法如下：

DIF=Z12-Z26

If t=1

DEA[t]=DIF[t]

If t>1

DEA[t]=(2\*DIF[t]+8\*DEA[t-1])/10

MACD[t]=2\*（DIF[t]-DEA[t]）

## 随机指标

随机指标（KDJ）一般是用于股票分析的统计体系，根据统计学原理，通过一个特定的周期(常为9日、9周等)内出现过的最高价、最低价及最后一个计算周期的收盘价及这三者之间的比例关系，计算最后一个计算周期的未成熟随机值RSV，然后根据平滑移动平均线的方法计算K值、D值与J值，并绘成曲线图研判股票价格走势。

计算公式为：









、分别表示n日内最高收盘价和最低收盘价，n=9。

Python计算移动周期内的最大最小值命令为：

pd.rolling\_max(P,n)

pd.rolling\_min(P,n)

其中，P为价格序列值，n为周期数。例如，计算9日移动最大最小值为：

Lmin=pd.rolling\_min(P,9)

Lmax=pd.rolling\_max(P,9)

RSV=(L-Lmin)/(Lmax-Lmin)

则计算KDJ指标算法如下：

If t=1

K[t]=RSV[t]

D[t]=RSV[t]

If t>1

K[t]=2/3\*K[t-1]+1/3\*RSV[t]

D[t]=2/3\*D[t-1]+1/3\*K[t]

J[t]=3\*D[t]-2\*K[t]

## 能量潮指标

能量潮指标（OBV）又称为能量潮，也叫成交量净额指标，是通过累计每日的需求量和供给量并予以数字化，制成趋势线，然后配合证券价格趋势图，从价格变动与成交量增减的关系上来推测市场气氛的一种技术指标。

计算公式为：



其中,sgn是符号函数，其数值由下面的式子决定：

算法如下：

（1）记P、S分别为价格序列和成交量序列，预定义obv=np.zeros((len(P)))。

（2）for t = 1 to len(P)

if t=1

obv[t]=S[t]

if t>1

if P[t]>=P[t-1]

obv[t]=obv[t-1]+S[t]

if P[t]<P[t-1]:

obv[t]=obv[t-1]-S[t]

## 乖离率指标

乖离率指标（BIAS）通过计算市场指数或收盘价与某条移动平均线之间的差距百分比，以反映一定时期内价格与其MA偏离程度的指标，从而得出价格在剧烈波动时因偏离移动平均趋势而造成回档或反弹的可能性，以及价格在正常波动范围内移动而形成继续原有势的可信度。

计算公式为：



算法如下：

（1）预定义乘离率指标bias=np.zeros((len(P))),其中P为价格序列。

（2）计算n日移动平均价格man=pd.rolling\_mean(P,n)。

（3）for t= n to len(P)

bias[t]=（P[t]-man[t]）/man[t]

## 涨跌趋势指标

股价趋势预测主要是通过建立预测模型 进行的， 是自变量， 是因变量。本节主要是将这些技术指标作为输入自变量 ，而因变量 是根据股票每日的收盘价确定的。下一日收盘价减去当日收盘价，若大于0，则下日股价呈现上涨趋势，记为+1类，反之则股价呈现下跌趋势，记为−1类。因变量y的计算方法如下：

（1）预定义y= np.zeros(len(P1))，其中P1为开盘价格序列

（2）预定义标识变量z=np.zeros(len(y)-1),并计算其涨跌方向

z[P2[2:end]-P1[2:end]>0]=1 涨

z[P2[2:end]-P1[2:end]==0]=0 平

z[P2[2:end]-P1[2:end]<0]=-1 跌

P2为收盘价序列

（3）for t = 1 to len(z)

y[t]=z[t]

最终将该问题转化为分类问题或者模式识别问题，相关的模型很多，如支持向量机、逻辑回归、神经网络，均能实现分类。

## 相对强弱指标

相对强弱指标是利用一定时期内平均收盘涨数与平均收盘跌数的比值来反映股市走势的。“一定时期”的选择是不同的，一般讲，天数选择短，易对起伏的股市产生动感，不易平衡长期投资的心理准备，做空做多的短期行为增多。天数选择长，对短期的投资机会不易把握。因此RSI一般可选用天数为6天、12天、24天。

计算公式为：



公式中， 日内收盘涨数； 日内收盘跌数；。

算法如下：

（1）预定义涨跌标识向量z，即z=np.zeros(len(P)-1)，其中P为价格序列。

（2）涨跌标识向量赋值。

z[P(2:end)- P(1:end-1)≥0]=1 涨

z[P(2:end)- P(1:end-1)<0]=-1 跌

（3）涨跌情况统计。

z1=pd.rolling\_sum(z==1,N) N日移动计算涨数

z2=pd.rolling\_sum(z==-1,N) N日移动计算跌数

（4）RSI指标计算。

for t= N to len(P)-1

rsi[t]= z1[t]/(z1[t]+z2[t])

# 量化投资模型与策略实现

首先利用基于总体规模与投资效率指标综合评价方法获得的排名结果，提取各行业排名前10的上市公司股票构建投资组合，并获取投资组合中各个股票在2015年的交易数据。其次，基于获取的股票交易数据计算技术分析指标（自变量）和涨跌趋势指标（因变量），并划分训练数据（2015年1月~10月）和预测数据（2015年11月~12月），构建逻辑回归预测模型，这里要求模型的准确率在70%以上（即针对训练数据的预测精度）才执行量化投资策略；最后，根据模型的预测结果构建量化投资策略，即如果预测结果为+1，表示下一个交易日收盘价较开盘价可能会上涨，则以下一交易日开盘价买入，收盘价卖出，计算每只股票的收益率，最终对每只的股票收益率求和获得投资组合的收益率。

## 投资组合构建

现以采掘业为例，根据排名结果提取排名10只股票代码构建投资组合，并批量获取投资组合中每只股票代码的交易数据，示例代码如下：

import pandas as pd

import fun

import Re\_comput

import time

start = time.clock()

#基于总体规模与投资效率指标的综合评价方法，获得排名前10的股票

#并构建投资组合

def mataindustry(x1,x2):

'''

x1：股票代码

x2:行业数据

'''

# 获得行业名

a = list(x2.iloc[x2.iloc[:, 1].values == x1, 0])

if a == []:

a = "其他"

else:

a = str(a)[2:-2]

# 获得股票代码名

return a

dt=pd.read\_excel('上市公司总体规模与投资效率指标1.xlsx')

x2=pd.read\_excel('sw.xlsx')

hy='采掘' #通过树点击获得

year='2014' #用按键输入或下拉框获得

r=fun.Fr(dt,year)

c=r[0]

c['行业']=c['Stkcd'].apply(lambda x:mataindustry(x1=x,x2=x2))

c=c.iloc[c.iloc[:,2].values==hy,:]

data1=c.iloc[:10,:]

list\_code=[]

list\_22=[]

code=data1.iloc[:,0].values

code=list(code)

year='201'+str(int(year[-1])+1) #年数加1

path=year+'年所有上市股票交易数据.xlsx' #获取后一年的量化投资数据

DA=pd.read\_excel(path) #2015年所有上市股票交易数据

执行结果中，10只股票代码查询打印结果如图4-1所示：

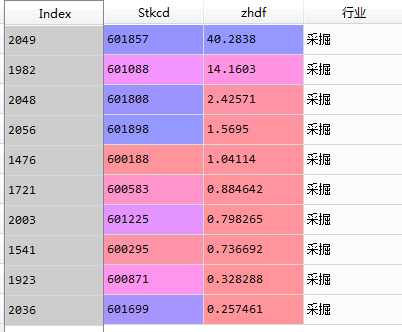


图4-1 采掘业前10的公司股票代码

执行结果中，20只股票代码交易数据表结构如图4-2所示：

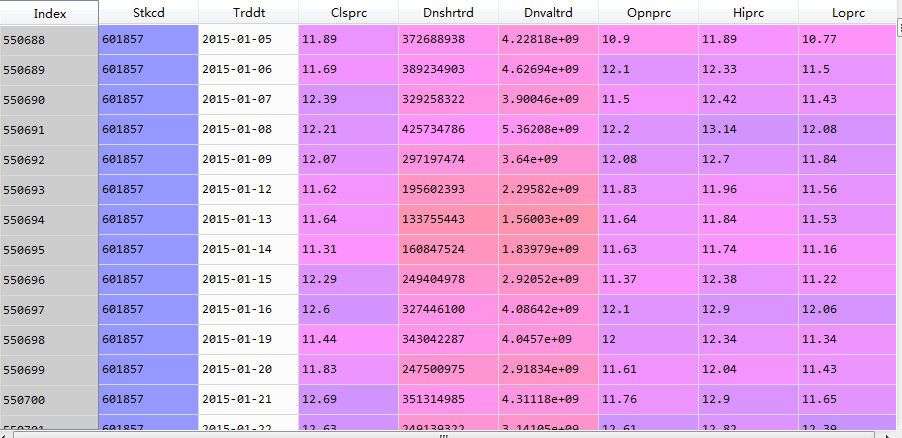


图4-2股票交易数据

## 基于逻辑回归的量化投资策略实现

首先读取投资组合的所有股票交易数据，并对每一只股票计算技术指标（自变量）和涨跌趋势指标（因变量），以2015年1月~10月的数据为训练样本，2015年11月~12月数据为预测样本，训练逻辑回归模型并对预测样本进行预测。如果预测模型准确率在70%（针对训练数据的预测准确率）以上，根据预测结果，如果为+1，则表示下一个交易日收盘价较开盘价可能会上涨，以下一个交易日开盘价买入，收盘价卖出，计算其投资收益率，完成一次交易机会，把所有的交易机会获得的投资收益率求和，即获得该只股票的收益率，所有股票的收益率之和就是投资组合的收益率。示例代码如下：

for i in range(len(code)):

data=DA.iloc[DA.iloc[:,0].values==code[i],:]

if len(data)>1:

list\_code.append(code[i])

z2=Re\_comput.Re(data,year)

list\_22.append(z2[2])

a=sum(list\_22)

D={'Stkcd':list\_code,'lr\_total':list\_22}

a={'Stkcd':'总收益','lr\_total':a}

a=pd.DataFrame(a)

D=pd.DataFrame(D)

D=np.vstack((D,a))

D=pd.DataFrame(D)

执行以上程序代码最终得到如图4-3所示：

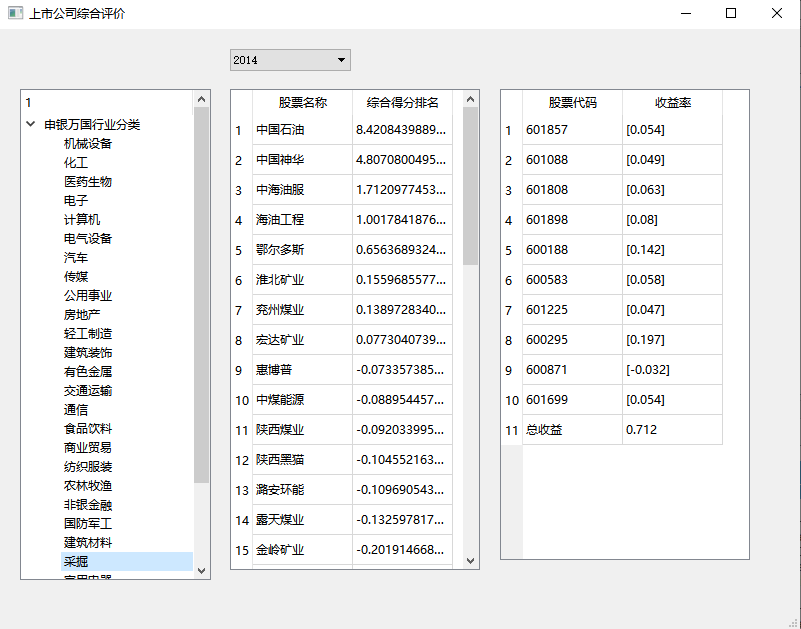


图4-3量化结果

# GUI可视系统设计与实现

## 系统设计

首先下载安装Pycharm软件，并配置代码生成工具QtDesigner（图5-1）和PyUCI（图5-2），配置完成后External Tools就出现这两个代码生成工具，如图5-3。

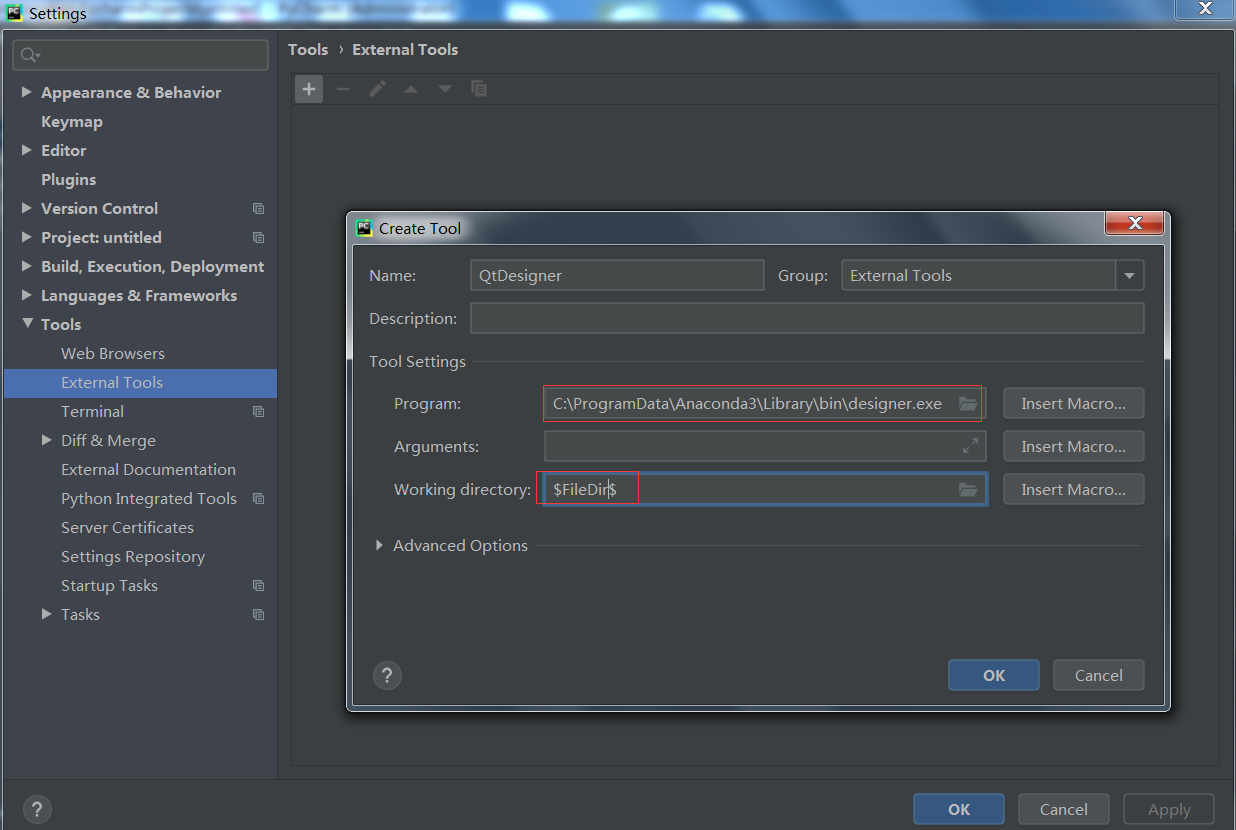


图5-1 配置QtDesigner

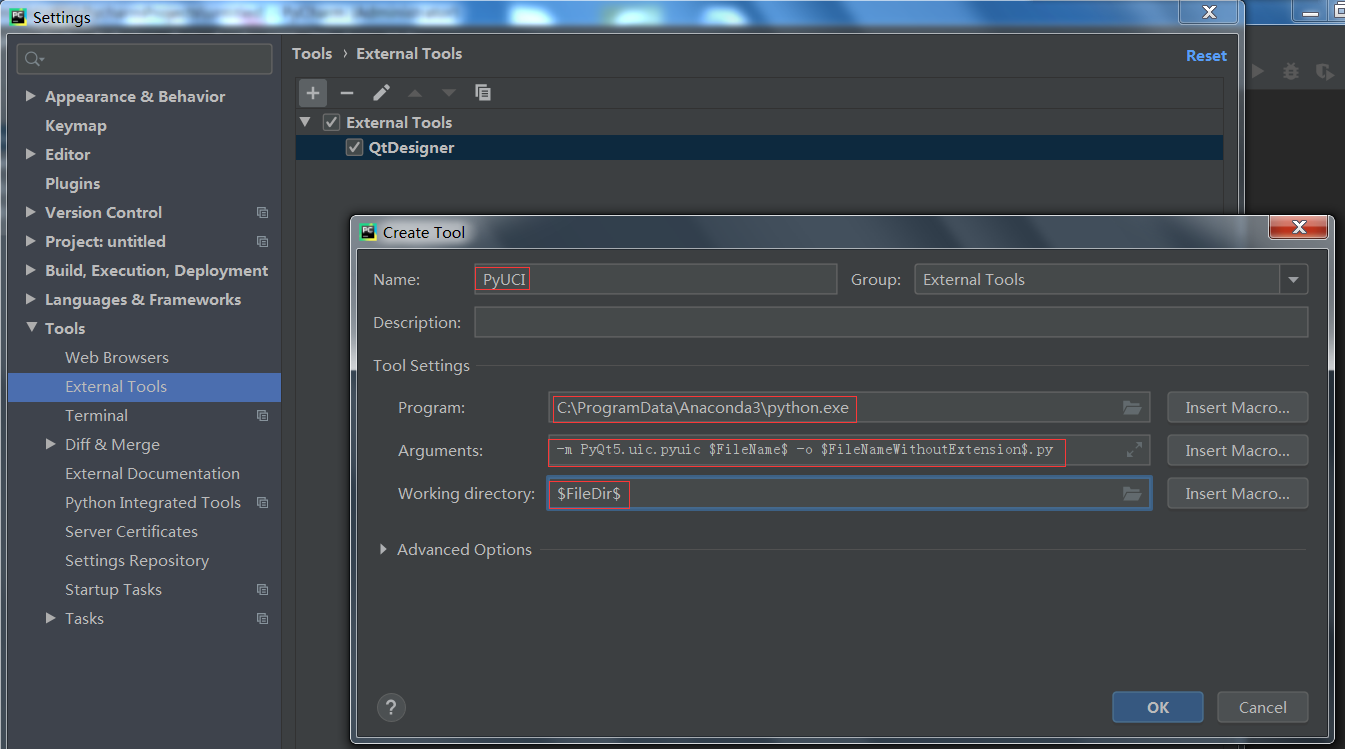


图5-2 配置PyUCI

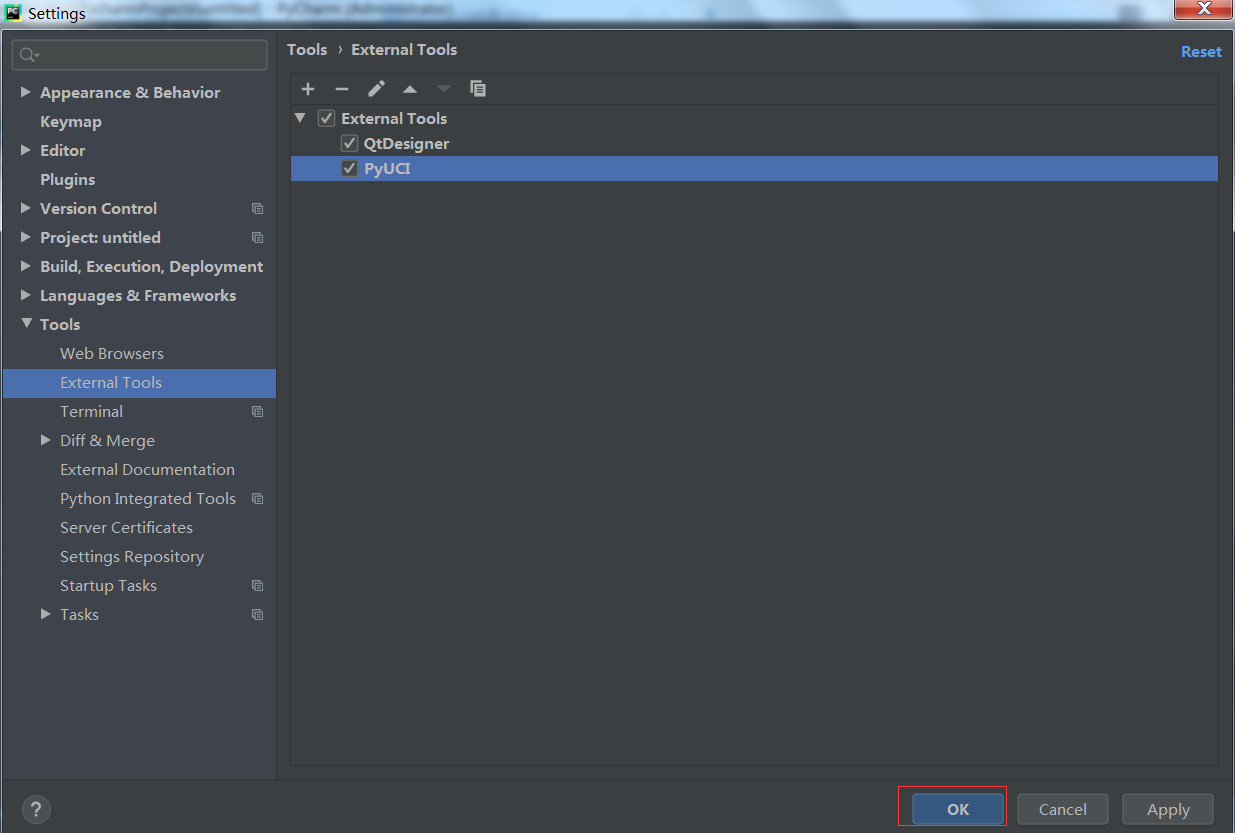


图5-3 配置工具

打开外部工具QtDesigner，弹出界面设计师软件窗口，在该窗口下即可设计系统界面，点击Main Window创建主窗体，接着从左边的控件栏目中拖拽一个Treewidget，用于存放树形结构的目录，然后拖拽两个tableView，一个用于输出并显示所选行业的综合得分排名，一个用于输出并显示投资收益；最后拖拽一个comboBox，用于选择年份。如图5-4。

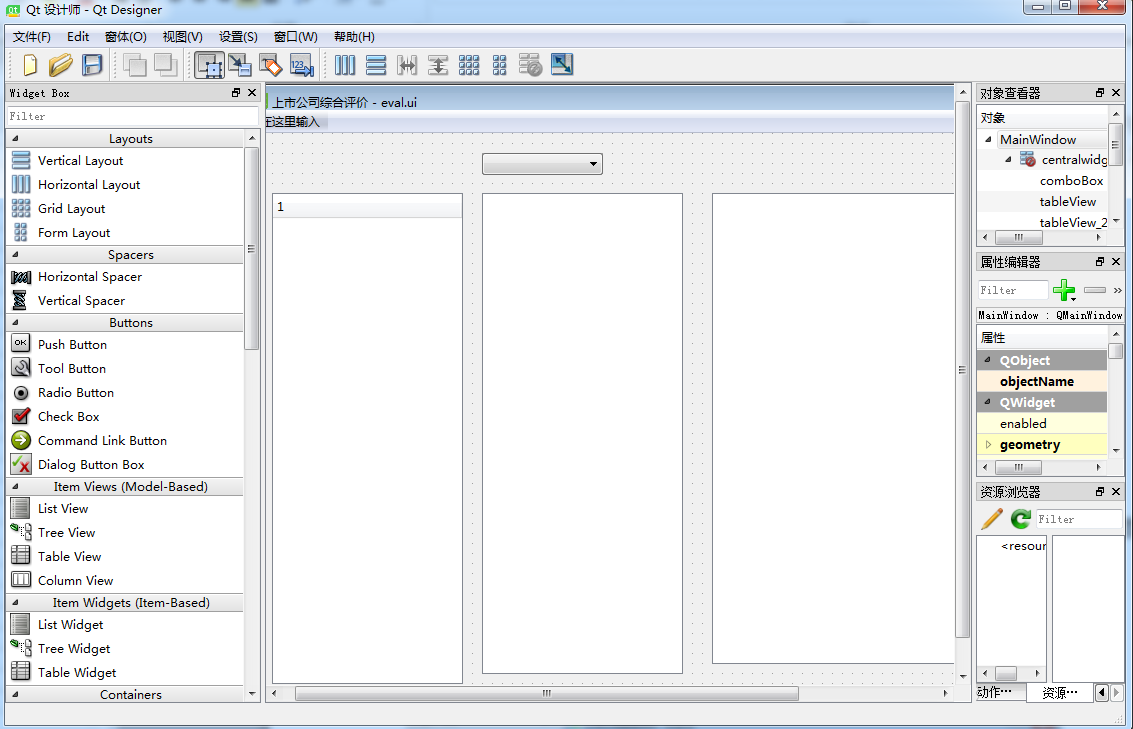


图5-4 创建主窗体

点击保存按钮，即可保存设计的界面，文件名后缀为.ui，默认情况下保存在项目文件下。在Pycharm中找到项目下的ui文件，点击右键External Tools下的PyUCI，就能生成后缀为.py的文件，就可以在py文件里面修改，编辑所要的功能。

## 系统实现

本系统实现的功能是点击Tree Widget控件上的其中一个目录就会出现相对应的数据，并将数据结果展现在相应的控件上。

在初始化函数setupUi内设置Tree Widget形成的目录，部分代码如图5-5所下：

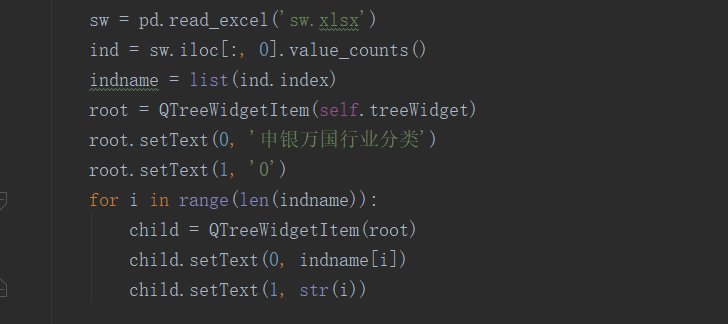


图5-5设计树目录

其中root表示根目录，child表示子目录，setText表示设置文本格式。

根据函数需要导入模块，如图5-6所示。通过PyUCI生成的代码，如图5-7和图 5-8，图5-14所示。点击comboBox按键，选择所需要的年份，如图5-9所示，建立comboBox按键与树控件的联系，如图5-10所示，点击树控件，得到相应的行业目录，如图 5-11所示。点击不同行业，得每个行业的综合得分排名及量化投资收益。通过tableView显示结果，分别如图 5-12、5-13所示。

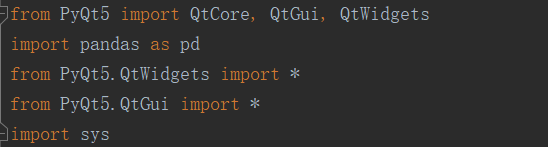
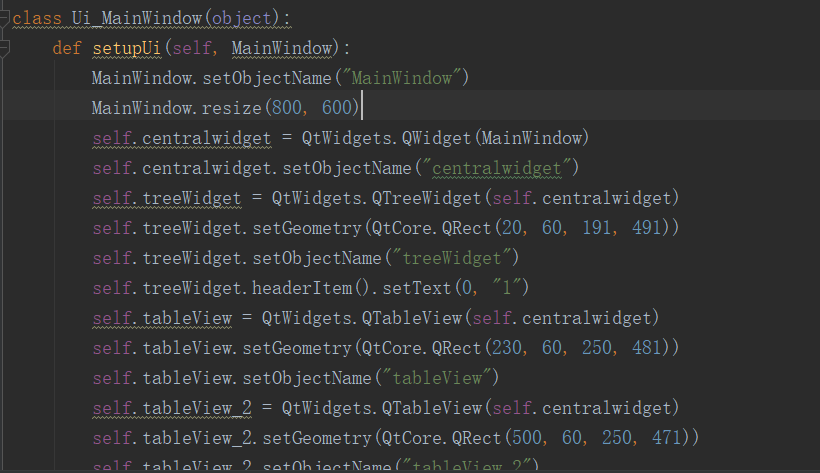
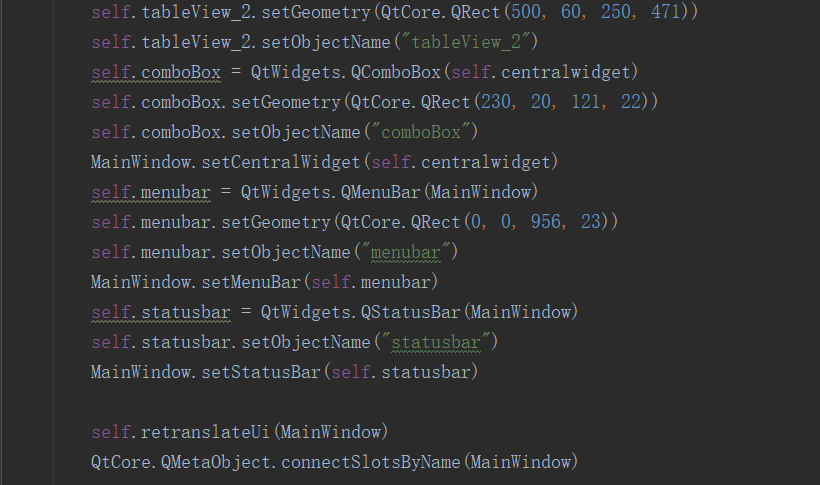


图5-6导入模块



5-7 PyUCI生成代码1



5-8 PyUCI生成代码2

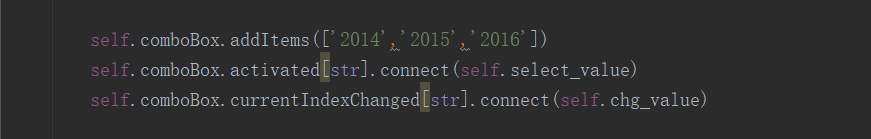


图5-9设计年份

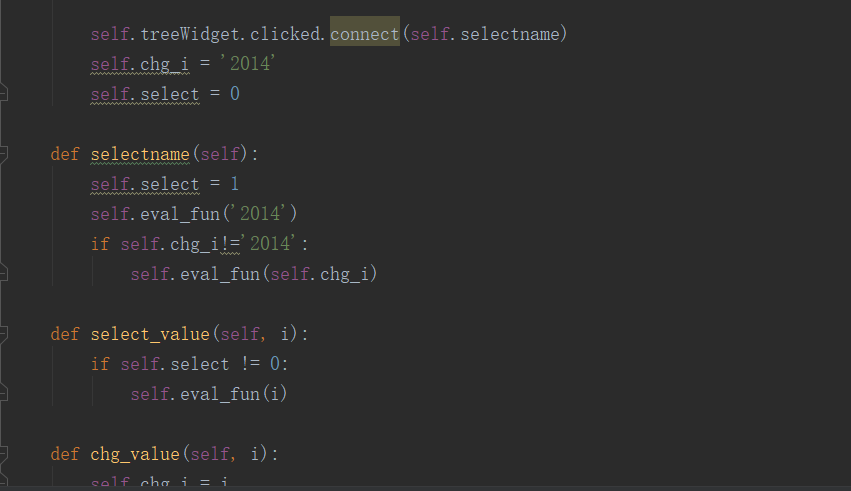
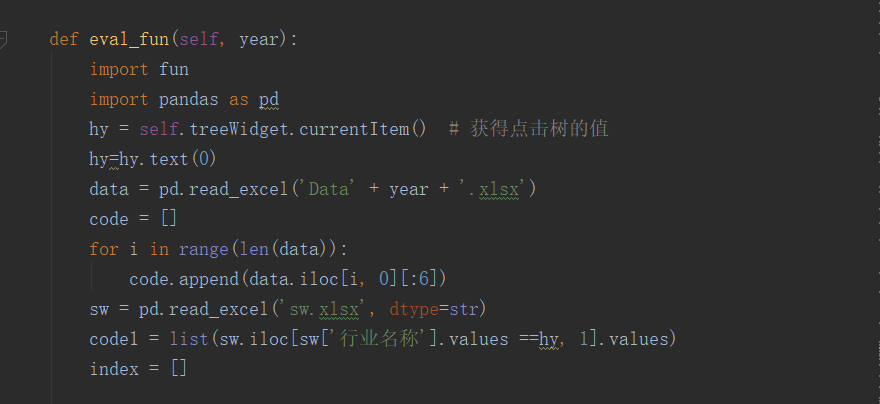
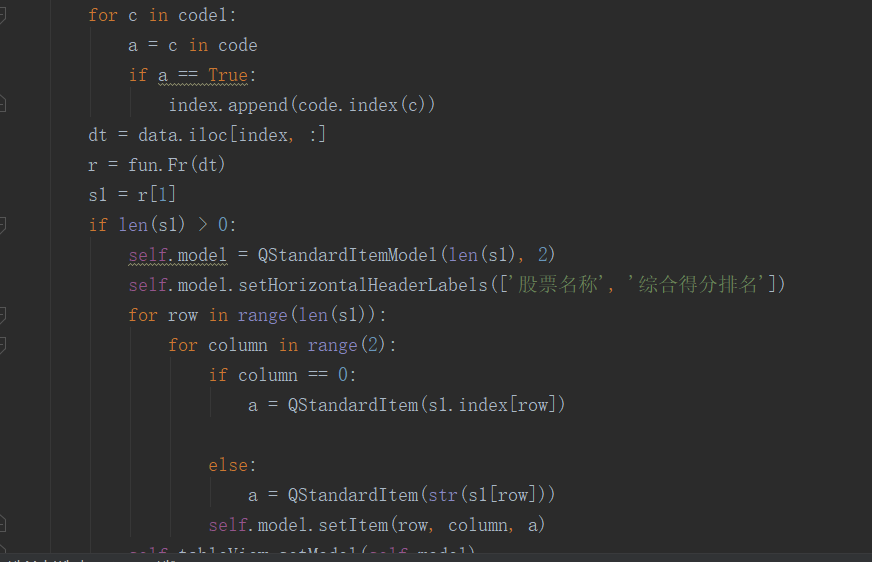


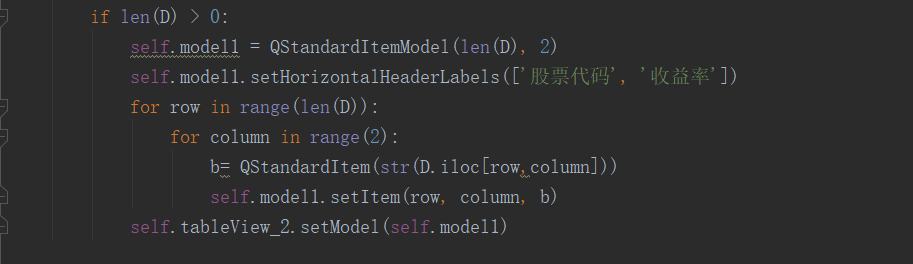
图5-10建立联系



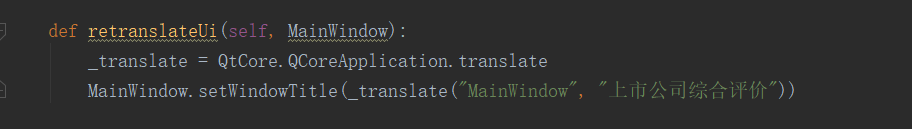
5-11 点击树控件得到行业目录



5-12 综合得分排名



5-13量化投资收益



5-14 PyUCI生成代码3

定义之后对Ui\_MainWindow类进行实例化，完成系统的程序编辑。通过系统的\_main\_函数入口实现调用，首先创建一个系统应用app，每个GUI界面都有一个系统应用app负责运行，其次创建一个主窗体对象，接着引用前面设计的界面Python类了，并通过类中的初始化函数实现主窗体，最终通过主窗体的show属性显示出来，最后一步即退出系统应用app。代码如图5-15所示：

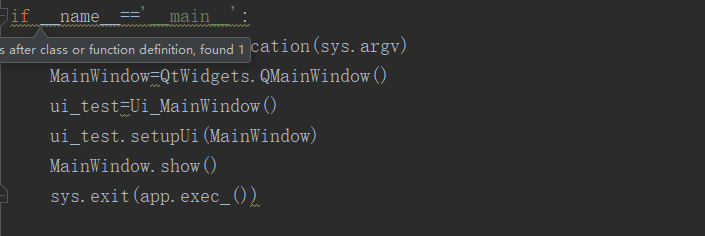


图5-15 \_\_main\_\_函数

## 需求分析

基于逻辑回归原理和GUI图形用户界面，创建函数程序，开发基于逻辑回归算法的GUI可视化系统。即安装PyCharm和配置开发环境，设计GUI界面，python程序编辑.py文件，在Anaconda Prompt编译.py文件，生成.exe文件，可以运行.exe文件即完现GUI可视化系统。

python程序.py文件的编辑思路，通过数据库获取数据，对数据进行预处理，计算相关技术分析指标，运用函数得到公司综合得分排名，再根据获取的前10名公司进行量化，得到一个最优的量化投资策略。具体流程如图 5-16所示。

图5-16流程图

## 生产独立运行的EXE

事实上，前面开发的系统实现并没有脱离Python的开发环境独立运行，在现实应用中一般需要将其编译成一个可独立运行的软件系统。为了更方便地查看数据，将其编译为可独立运行的软件系统，接下来，我们将详细说明如何将其编译为可独立于Python开发环境运行的软件系统。

1. 安装pyinstaller

使用pip install安装命令即可pyinstaller编译包，如图5-17所示可以在Anaconda Prompt下实现安装。

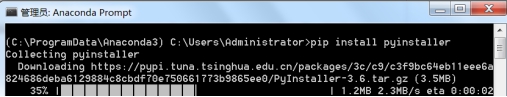


图5-17 安装pyinstaller

1. 安装依赖包

安装好pyinstaller编译包后，仍需安装三个依赖包，pywin32-ctypes、altgraph和pefile，如图5-18。

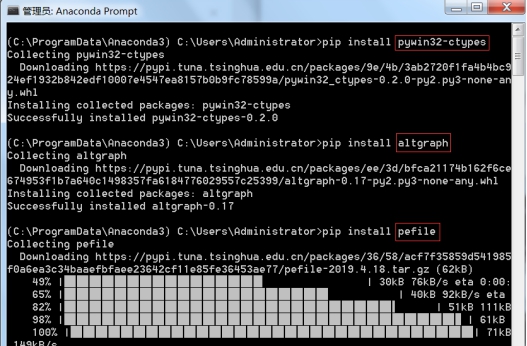


图5-18安装依赖包

1. 生成EXE文件

首先在Anaconda Prompt下利用操作命令切换至当前的项目文件夹路径，如图5-19所示，其中该项目文件下如图5-20所示。



图5-19操作命令切换路径

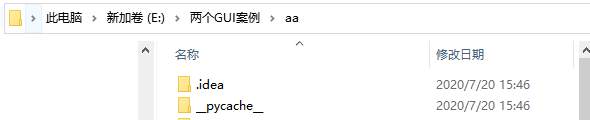


图5-20项目文件夹路径

利用命令：“pyinstaller -F 需要编译的文件”即可进行编译，其中本项目需要编译的程序文件为eval.py。如图5-21所示。



图5-21编译文件

有时候并不能一次性编译成功，在本次编译中，出现了超出最大递归深度（如图5-22）、编译成功后运行EXE还可能会缺包等情况（如图5-23），

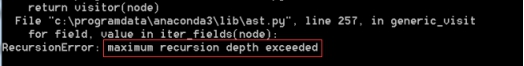


图5-22超出最大递归深度

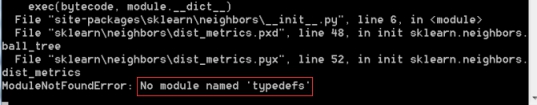


图5-23缺包

实际上，即使编译不成功在项目文件夹下也会产生一个与项目名称相同的.spec文件，如图5-24所示。

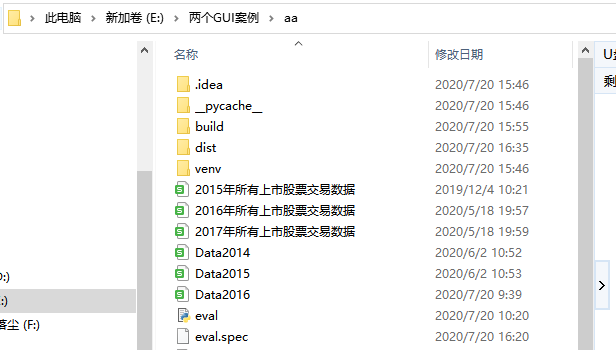


图5-24项目文件

该文件可以用Python开发环境pycharm或者Spyder打开。解决以上两个问题的方法是对该文件进行修改，比如第一个问题是修改最大递归深度限制，第二个问题是由于某些开发包在编译过程中不兼容编译器所致，可以对相关的包单独导入，如图5-25所示。第一个框是导入系统模块，并设置最大递归深度为5000，第二个框为单独导入相关包，完整代码为：

hiddenimports=['cython','sklearn','sklearn.ensemble','sklearn.neighbors.typedefs','sklearn.neighbors.quad\_tree','sklearn.tree.\_utils','scipy.\_lib.messagestream'],

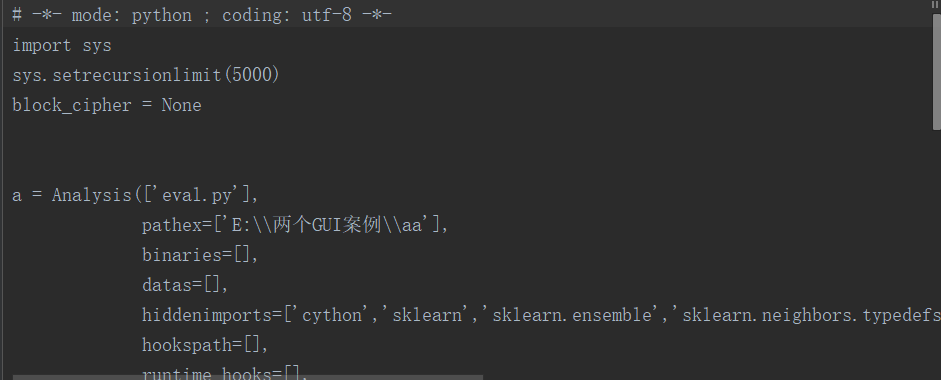


图5-25修改文件

对eval.spec文件修改完成之后，可以利用pyinstaller命令对这个文件进行再次编译即可，即：“pyinstaller eval.spec”，如图5-26所示。



图5-26编译eval.spec

编译成功后，在项目文件夹下会存在bulid、dist这两个文件夹，如图5-27所示，还有eval.spec，其中可执行文件存在于dist文件夹。

此时，该EXE文件还不能独立运行，还需要开发环境中的platforms文件拷贝至该目录下，同时，将项目所需的数据和其他调用的文件复制到目录中，如图5-28所示。

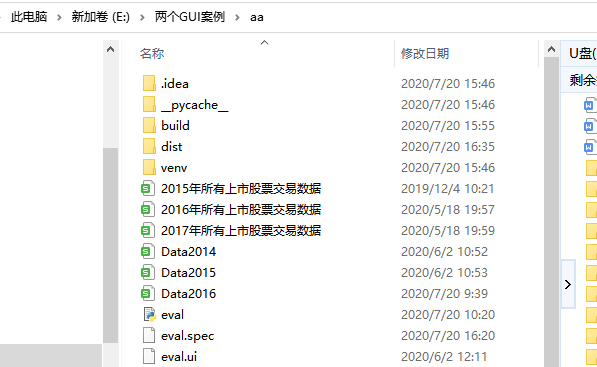


图5-27项目文件

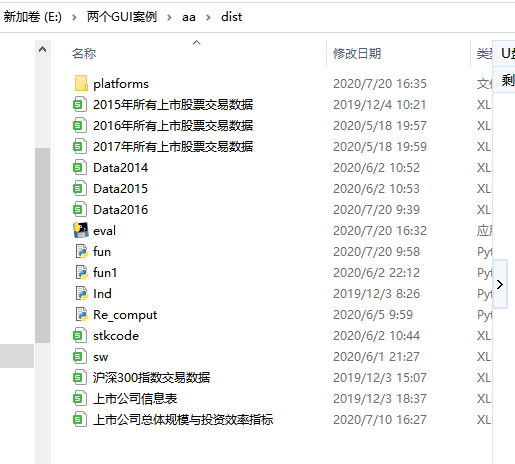


图5-28

## 系统演示

进入EXE文件路径下，点击生成的EXE系统文件，会出现一个窗口。点击树控件及comboBox按键，选择所需行业及年份，结果显示如图5-29所示。

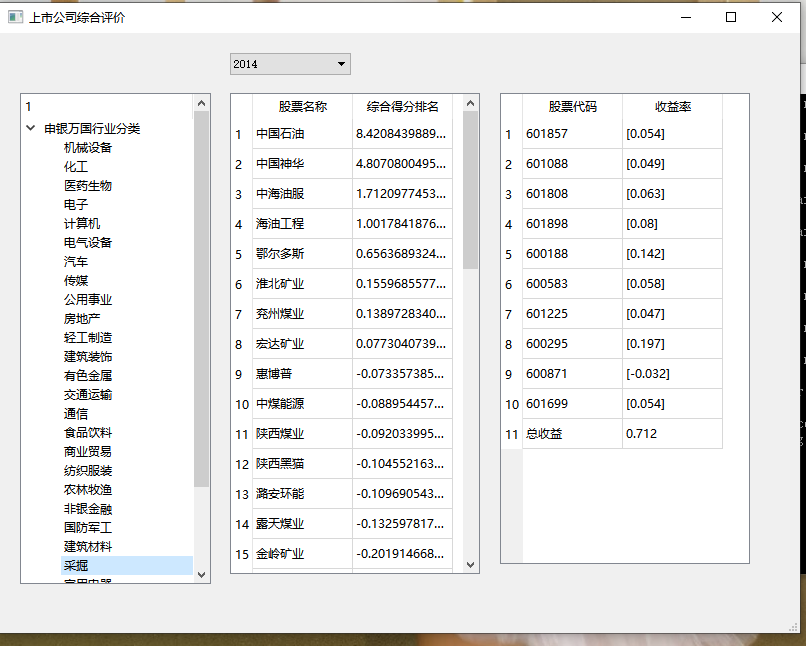


图5-29系统演示结果

# 分析及总结

本文基于财务及交易数据，介绍了量化投资的全过程，包括上市公司代码选择、技术指标计算、预测模型构建及投资策略设计等。利用常见的技术分析指标对个股价格涨跌趋势进行了预测，其技术分析指标包括移动平均线、指数平滑异同平均线、随机指标、相对强弱指标、乖离率指标、能量潮指标。最后基于总体规模与投资效率指标的综合评价方法获取排名靠前的股票样本作为研究对象，主要目的是为了在优质股中进行投资，并在此基础上设计了量化投资策略。需要说明的是，本文是基于历史数据进行实证检验，并没有考虑到实际的市场政策风险、行业风险及突发事件影响，不构成投资建议。同时，本文中的交易方式是T+0，而我国市场并没有实行T+0交易机制，只能通过存量股票或者其他的金融投资工具来实现。本文的意义在于通过真实的财务及交易数据，详细介绍了量化投资的全过程，同时对量化投资爱好者、个人投资者或机构投资者也具有较好的参考作用。但是也在这里提醒投资者一句话，股市有风险，投资需谨慎。

# 参考文献

1. 李昊 股市技术分析指标大全[J] 中国民主法制出版社; 第1版
2. 德]安德里亚斯·穆勒 / [美]莎拉·吉多Python机器学习基础教程[M]  人民邮电出版社
3. 王达菲 股票技术分析大全[M] 中国华侨出版社 2013年10月1日
4. 外汇经纪人 一些技术指标的计算公式及方法[J/OL]

网址：<http://blog.sina.com.cn/s/blog_511104aa01008tuj.html> 2008年3月24日

1. 王维波,栗宝鹃,张晓东 Python Qt GUI与数据可视化编程[M] 人民邮电出版社 2019年9月

# 程序附录

**获取沪深300指数,** **沪深300.sql:**

SELECT [Indexcd],[Idxtrd01],[Idxtrd05]

FROM [CSMAR\_Data].[dbo].[IDX\_Idxtrd]

where Indexcd='000300'

order by Idxtrd01

**获取上市公司交易数据,交易数据.sql:**

SELECT [Stkcd],[Trddt],[Clsprc],[Dnshrtrd],[Dnvaltrd],[Opnprc],[Hiprc],[Loprc]

FROM [CSMAR\_Data].[dbo].[TRD\_Dalyr]

where Trddt between '2015-01-01' and '2015-12-31'

order by Stkcd,Trddt

**获取指标数据，指标.sql:**

select W.\*,Q.F050501B,Q.F091301A,Q.F091001A,Q.F090101B from

(select a.\*,b.A001000000,b.A001212000 from

(SELECT [Stkcd],[Accper],[B001101000],[B001300000],[B001000000] ,[B002000000]

FROM [CSMAR\_Data].[dbo].[FS\_Comins]

Where Typrep='A 'and Accper in ('2011-12-31','2012-12-31','2013-12-31',

'2014-12-31','2015-12-31','2016-12-31')

)a

inner join

(SELECT [Stkcd],[Accper],[Typrep],[A001000000],[A001212000]

FROM [CSMAR\_Data].[dbo].[FS\_Combas]

where Typrep='A' and Accper in ('2011-12-31','2012-12-31','2013-12-31',

'2014-12-31','2015-12-31','2016-12-31')

)b

on a.Stkcd=b.Stkcd and a.Accper=b.Accper

)W

inner join

(select c.\*,d.F091001A,d.F091301A,d.F090101B from

(SELECT [Stkcd],[Accper],[F050501B]

FROM [CSMAR\_Data].[dbo].[FI\_T5]

where Typrep='A' and Accper in ('2011-12-31','2012-12-31','2013-12-31',

'2014-12-31','2015-12-31','2016-12-31')

)c

inner join

(SELECT [Stkcd],[Accper],[F091001A],[F091301A],[F090101B]

FROM [CSMAR\_Data].[dbo].[FI\_T9]

where Typrep='A' and Accper in ('2011-12-31','2012-12-31','2013-12-31',

'2014-12-31','2015-12-31','2016-12-31')

)d

on c.Stkcd=d.Stkcd and c.Accper=d.Accper

)Q

on w.Stkcd=q.Stkcd and W.Accper=Q.Accper

order by W.Stkcd,W.Accper

**fun.py**

def Fr(data):  
 import pandas as pd  
 #data=pd.read\_excel('Data.xlsx')  
 data=data[data>0]  
 data=data.dropna()  
 from sklearn.preprocessing import StandardScaler   
 X=data.iloc[:,1:]  
 scaler = StandardScaler()  
 scaler.fit(X)   
 X=scaler.transform(X)   
 from sklearn.decomposition import PCA   
 pca=PCA(n\_components=0.95) #累计贡献率为95%  
 Y=pca.fit\_transform(X) #满足累计贡献率为95%的主成分数据  
 gxl=pca.explained\_variance\_ratio\_ #贡献率  
 import numpy as np  
 F=np.zeros((len(Y)))  
 for i in range(len(gxl)):  
 f=Y[:,i]\*gxl[i]  
 F=F+f  
   
 fs1=pd.Series(F,index=data['ts\_code'].values)  
 Fscore1=fs1.sort\_values(ascending=False) #降序，True为升序  
   
 co=pd.read\_excel('stkcode.xlsx')  
 #Co=pd.Series(co['name'].values,index=co['ts\_code'].values)  
 #Co1=Co[data['ts\_code'].values]  
 #fs2=pd.Series(F,index=Co1.values)  
 dict1={'ts\_code':data['ts\_code'].values,'F':F}  
 df1=pd.DataFrame(dict1)  
 df2=pd.merge(df1,co.iloc[:,[0,2]],how='inner',on='ts\_code')  
 fs2=pd.Series(df2['F'].values,index=df2['name'].values)  
 Fscore2=fs2.sort\_values(ascending=False) #降序，True为升序  
 return (Fscore1,Fscore2)

**fun1.py**

def Fr1(data, year):  
 import pandas as pd  
 # 读取所需年份数据，其中第0列为标识列（股票代码）  
 data2 = data.iloc[data['Accper'].values == str(year) + '-12-31', [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]]  
 # 筛选指标值大于0的数据以及去掉NAN值  
 data2 = data2[data2 > 0]  
 data2 = data2.dropna()  
 # 数据标准化，注意标准化的数据需要去掉第0列（股票代码，标识列），这里数据标准化方法采用均值-方差法。  
 from sklearn.preprocessing import StandardScaler  
 X = data2.iloc[:, 1:]  
 scaler = StandardScaler()  
 scaler.fit(X)  
 X = scaler.transform(X)  
 # 主成分分析  
 from sklearn.decomposition import PCA  
 pca = PCA(n\_components=0.95) # 累计贡献率为95%  
 Y = pca.fit\_transform(X) # 满足累计贡献率为95%的主成分数据  
 gxl = pca.explained\_variance\_ratio\_ # 贡献率  
 import numpy as np  
 F = np.zeros((len(Y)))  
 for i in range(len(gxl)):  
 f = Y[:, i] \* gxl[i]  
 F = F + f  
 Fscore1 = pd.DataFrame(data2['Stkcd'].values, columns=['Stkcd'])  
 Fscore1['zhdf'] = F  
 Fscore1 = Fscore1.sort\_values('zhdf', ascending=False) # 降序，True为升序  
  
 co = pd.read\_excel('上市公司信息表.xlsx')  
 Co = pd.Series(co['Stknme'].values, index=co['Stkcd'].values)  
 Co1 = Co[data2['Stkcd'].values]  
 Fscore2 = pd.DataFrame(Co1.values, columns=['Stknme'])  
 Fscore2['zhdf'] = F  
 Fscore2 = Fscore2.sort\_values('zhdf', ascending=False) # 降序，True为升序  
 return (Fscore1, Fscore2)

**指标函数，Ind.py**

import pandas as pd  
#移动平均线指标的计算  
def MA(data,N1,N2,N3):  
 MAN1=pd.rolling\_mean(data['Clsprc'].values,N1)   
 MAN2=pd.rolling\_mean(data['Clsprc'].values,N2)   
 MAN3=pd.rolling\_mean(data['Clsprc'].values,N3)   
 return (MAN1,MAN2,MAN3)  
#指数平滑异同平均线的计算  
def MACD(data):   
 import numpy as np   
 EMA12 = pd.ewma(data['Clsprc'].values, 12)  
 EMA26 = pd.ewma(data['Clsprc'].values, 26)  
 DIF=EMA12- EMA26  
 DEA=np.zeros((len(DIF)))  
 MACD=np.zeros((len(DIF)))  
 for t in range(len(DIF)):  
 if t==0:  
 DEA[t]= DIF[t]  
 if t>0:  
 DEA[t]=(2\*DIF[t]+8\*DEA[t-1])/10  
 MACD[t]=2\*(DIF[t]-DEA[t])  
 return MACD  
#随机指标KDJ的计算  
def KDJ(data,N):  
 import numpy as np   
 Lmin=pd.rolling\_min(data['Loprc'].values,N)  
 Lmax=pd.rolling\_max(data['Hiprc'].values,N)  
 RSV=(data['Clsprc'].values-Lmin)/(Lmax-Lmin)  
 K=np.zeros((len(RSV)))  
 D=np.zeros((len(RSV)))  
 J=np.zeros((len(RSV)))  
 for t in range(N,len(data)):  
 if t==0:  
 K[t]=RSV[t]  
 D[t]=RSV[t]  
 if t>0:  
 K[t]=2/3\*K[t-1]+1/3\*RSV[t]  
 D[t]=2/3\*D[t-1]+1/3\*K[t]  
 J[t]=3\*D[t]-2\*K[t]  
 return (K,D,J)  
#相对强弱指标的计算  
def RSI(data,N):  
 import numpy as np  
 z=np.zeros(len(data)-1)   
 z[data.iloc[1:,2].values-data.iloc[0:-1,2].values>=0]=1  
 z[data.iloc[1:,2].values-data.iloc[0:-1,2].values<0]=-1  
 z1=pd.rolling\_sum(z==1,N)  
 z2=pd.rolling\_sum(z==-1,N)  
 rsi=np.zeros((len(data)))  
 for t in range(N-1,len(data)-1):  
 rsi[t]=z1[t]/(z1[t]+z2[t])  
 return rsi  
#乖离率指标  
def BIAS(data,N):  
 import numpy as np  
 bias=np.zeros((len(data)))  
 man=pd.rolling\_mean(data.iloc[:,2].values,N)  
 for t in range(N-1,len(data)):  
 bias[t]=(data.iloc[t,2]-man[t])/man[t]  
 return bias  
#能量潮指标的计算  
def OBV(data):  
 import numpy as np  
 obv=np.zeros((len(data)))  
 for t in range(len(data)):  
 if t==0:  
 obv[t]=data['Dnshrtrd'].values[t]  
 if t>0:  
 if data['Clsprc'].values[t]>=data['Clsprc'].values[t-1]:  
 obv[t]=obv[t-1]+data['Dnshrtrd'].values[t]  
 if data['Clsprc'].values[t]<data['Clsprc'].values[t-1]:  
 obv[t]=obv[t-1]-data['Dnshrtrd'].values[t]  
 return obv  
#涨跌趋势（分类指标/预测指标）的计算  
def cla(data):  
 import numpy as np  
 y=np.zeros(len(data))   
 z=np.zeros(len(y)-1)  
 for i in range(len(z)):  
 z[data.iloc[1:,2].values-data.iloc[0:-1,2].values>0]=1  
 z[data.iloc[1:,2].values-data.iloc[0:-1,2].values==0]=0  
 z[data.iloc[1:,2].values-data.iloc[0:-1,2].values<0]=-1  
 y[i]=z[i]   
 return y

**Re\_comput.py**

def Re(data,year):  
 #data--表示某只股票代码的交易数据  
 #n--表示选用的方法，0--神经网络，1--支持向量机，2--逻辑回归  
 #返回结果为一个元组(R,r\_list,r\_total,sc)  
 #其中R--表示预测准确率，r\_list--表示策略每次投资收益  
 # r\_total--表示总收益，sc--表示模型准确率  
  
 ##1.计算指标  
 import pandas as pd  
 import Ind  
 import numpy as np  
 MA= Ind.MA(data,5,10,20)   
 macd=Ind.MACD(data)  
 kdj=Ind.KDJ(data,9)  
 rsi6=Ind.RSI(data,6)  
 rsi12=Ind.RSI(data,12)  
 rsi24=Ind.RSI(data,24)  
 bias5=Ind.BIAS(data,5)  
 bias10=Ind.BIAS(data,10)  
 bias20=Ind.BIAS(data,20)  
 obv=Ind.OBV(data)   
 y=Ind.cla(data)  
 pm={'交易日期':data['Trddt'].values}  
 PM=pd.DataFrame(pm)  
 DF={'MA5':MA[0],'MA10':MA[1],'MA20':MA[2],'MACD':macd,  
 'K':kdj[0],'D':kdj[1],'J':kdj[2],'RSI6':rsi6,'RSI12':rsi12,  
 'RSI24':rsi24,'BIAS5':bias5,'BIAS10':bias10,'BIAS20':bias20,'OBV':obv}  
 DF=pd.DataFrame(DF)  
 s1=PM.join(DF)  
 y1={'涨跌趋势':y}  
 ZZ=pd.DataFrame(y1)  
 s2=s1.join(ZZ)  
 ss=s2.dropna()  
 Data=ss[ss.iloc[:,6].values!=0]  
 ##2.训练样本与测试样本的划分  
 x1=Data['交易日期']>=year+'-01-01'  
 x2=Data['交易日期']<=year+'-11-30'  
 xx=x1&x2  
 index=xx.values==True  
 index1=xx.values==False  
 x\_train=Data.iloc[index,1:15]   
 y\_train=Data.iloc[index,[15]]  
 x\_test=Data.iloc[index1,1:15]  
 y\_test=Data.iloc[index1,[15]]  
 ##3.数据标准化处理  
 from sklearn.preprocessing import StandardScaler   
 scaler = StandardScaler()  
 scaler.fit(x\_train)   
 x\_train=scaler.transform(x\_train)  
 x\_test=scaler.transform(x\_test)   
 #逻辑回归模型  
 from sklearn.linear\_model import LogisticRegression as LR  
 clf = LR()  
 clf.fit(x\_train, y\_train)   
 result=clf.predict(x\_test)  
 sc=clf.score(x\_train, y\_train)  
 sc=np.round(sc,3)  
 result=pd.DataFrame(result)  
 ff=Data.iloc[index1,0]  
 #将预测结果与实践结果整合在一起，进行比较  
 pm1={'交易日期':ff.values,'预测结果':result.iloc[:,0].values,  
 '实际结果':y\_test.iloc[:,0].values}  
 result1=pd.DataFrame(pm1)  
 z=result1['预测结果'].values-result1['实际结果'].values  
 R=len(z[z==0])/len(z)  
 R=np.round(R,3)   
 ##5.收益率计算  
 r\_list=[]  
 for t in range(len(result1)-1):  
 if result1['预测结果'].values[t]==1:  
 p1=data.loc[data['Trddt'].values== result1['交易日期'].values[t],'Clsprc'].values  
 dt=data.loc[data['Trddt'].values>result1['交易日期'].values[t],['Trddt','Clsprc']]  
 dt=dt.sort\_values('Trddt')  
 p2=dt['Clsprc'].values[0]  
 r=(p2-p1)/p1  
 r\_list.append(r)  
 r\_total=sum(r\_list)  
 r\_total=np.round(r\_total,3)  
 return (R,r\_list,r\_total,sc)

**系统，eval.py**

from PyQt5 import QtCore, QtGui, QtWidgets  
import pandas as pd  
from PyQt5.QtWidgets import \*  
from PyQt5.QtGui import \*  
import sys  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(800, 600)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.treeWidget = QtWidgets.QTreeWidget(self.centralwidget)  
 self.treeWidget.setGeometry(QtCore.QRect(20, 60, 191, 491))  
 self.treeWidget.setObjectName("treeWidget")  
 self.treeWidget.headerItem().setText(0, "1")  
 self.tableView = QtWidgets.QTableView(self.centralwidget)  
 self.tableView.setGeometry(QtCore.QRect(230, 60, 250, 481))  
 self.tableView.setObjectName("tableView")  
 self.tableView\_2 = QtWidgets.QTableView(self.centralwidget)  
 self.tableView\_2.setGeometry(QtCore.QRect(500, 60, 250, 471))  
 self.tableView\_2.setObjectName("tableView\_2")  
 self.comboBox = QtWidgets.QComboBox(self.centralwidget)  
 self.comboBox.setGeometry(QtCore.QRect(230, 20, 121, 22))  
 self.comboBox.setObjectName("comboBox")  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 956, 23))  
 self.menubar.setObjectName("menubar")  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 self.comboBox.addItems(['2014','2015','2016'])  
 self.comboBox.activated[str].connect(self.select\_value)  
 self.comboBox.currentIndexChanged[str].connect(self.chg\_value)  
  
 sw = pd.read\_excel('sw.xlsx')  
 ind = sw.iloc[:, 0].value\_counts()  
 indname = list(ind.index)  
 root = QTreeWidgetItem(self.treeWidget)  
 root.setText(0, '申银万国行业分类')  
 root.setText(1, '0')  
 for i in range(len(indname)):  
 child = QTreeWidgetItem(root)  
 child.setText(0, indname[i])  
 child.setText(1, str(i))  
  
 self.treeWidget.clicked.connect(self.selectname)  
 self.chg\_i = '2014'  
 self.select = 0  
  
 def selectname(self):  
 self.select = 1  
 self.eval\_fun('2014')  
 if self.chg\_i!='2014':  
 self.eval\_fun(self.chg\_i)  
  
 def select\_value(self, i):  
 if self.select != 0:  
 self.eval\_fun(i)  
  
 def chg\_value(self, i):  
 self.chg\_i = i  
  
 def mataindustry(self,x1, x2):  
 *'''  
 x1：股票代码  
 x2:行业数据  
 '''* # 获得行业名  
 w = list(x2.iloc[x2.iloc[:, 1].values == x1, 0])  
 if w == []:  
 w = "其他"  
 else:  
 w = str(w)[2:-2]  
 # 获得股票代码名  
 return w  
  
 def eval\_fun(self, year):  
 import fun  
 import pandas as pd  
 hy = self.treeWidget.currentItem() # 获得点击树的值  
 hy=hy.text(0)  
 data = pd.read\_excel('Data' + year + '.xlsx')  
 code = []  
 for i in range(len(data)):  
 code.append(data.iloc[i, 0][:6])  
 sw = pd.read\_excel('sw.xlsx', dtype=str)  
 code1 = list(sw.iloc[sw['行业名称'].values ==hy, 1].values)  
 index = []  
  
 for c in code1:  
 a = c in code  
 if a == True:  
 index.append(code.index(c))  
 dt = data.iloc[index, :]  
 r = fun.Fr(dt)  
 s1 = r[1]  
 if len(s1) > 0:  
 self.model = QStandardItemModel(len(s1), 2)  
 self.model.setHorizontalHeaderLabels(['股票名称', '综合得分排名'])  
 for row in range(len(s1)):  
 for column in range(2):  
 if column == 0:  
 a = QStandardItem(s1.index[row])  
  
 else:  
 a = QStandardItem(str(s1[row]))  
 self.model.setItem(row, column, a)  
 self.tableView.setModel(self.model)  
  
 import pandas as pd  
 import fun1  
 import numpy as np  
 import Re\_comput  
 # 构建投资组合  
 dt = pd.read\_excel('上市公司总体规模与投资效率指标.xlsx')  
 x2 = pd.read\_excel('sw.xlsx')  
 r = fun1.Fr1(dt,year)  
 c = r[0]  
 c['行业'] = c['Stkcd'].apply(lambda x: self.mataindustry(x1=x, x2=x2))  
 c = c.iloc[c.iloc[:, 2].values == hy, :]  
 data1 = c.iloc[:10, :]  
 list\_code = []  
 list\_22 = []  
 code = data1.iloc[:, 0].values  
 code = list(code)  
 year = '201' + str(int(year[-1]) + 1) # 年数加1  
 path = year + '年所有上市股票交易数据.xlsx' # 获取后一年的量化投资数据  
 DA = pd.read\_excel(path) # 2015年所有上市股票交易数据  
 for i in range(len(code)):  
 data3 = DA.iloc[DA.iloc[:, 0].values == code[i], :]  
 if len(data3) > 1:  
 list\_code.append(code[i])  
 z2 = Re\_comput.Re(data3, year)  
 list\_22.append(z2[2])  
 a=sum(list\_22)  
 D = {'Stkcd': list\_code, 'lr\_total': list\_22}  
 a={'Stkcd':'总收益','lr\_total':a}  
 a=pd.DataFrame(a)  
 D=pd.DataFrame(D)  
 D=np.vstack((D,a))   
 D=pd.DataFrame(D)   
 if len(D) > 0:  
 self.model1 = QStandardItemModel(len(D), 2)  
 self.model1.setHorizontalHeaderLabels(['股票代码', '收益率'])  
 for row in range(len(D)):  
 for column in range(2):   
 b= QStandardItem(str(D.iloc[row,column]))  
 self.model1.setItem(row, column, b)  
 self.tableView\_2.setModel(self.model1)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "上市公司综合评价"))  
  
if \_\_name\_\_=='\_\_main\_\_':  
 app=QtWidgets.QApplication(sys.argv)  
 MainWindow=QtWidgets.QMainWindow()  
 ui\_test=Ui\_MainWindow()  
 ui\_test.setupUi(MainWindow)  
 MainWindow.show()  
 sys.exit(app.exec\_())