pbroe模型

1. 指标分析

pb=股价/每股净资产（每天都会更新）

roe=每股收益/每股净资产（一个季度更新）

1. 数据准备

Pb取某个季度的均值，roe取某个季度的均值;在这里取的数据是2017年第一季度的值。

1. 方法概述

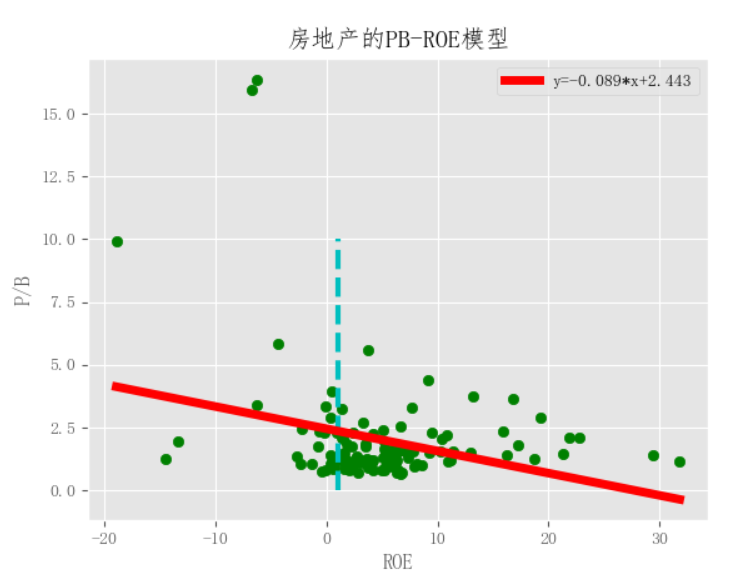
(1)pb作为因变量，roe作为特征变量,搭建一元线性回归分析模型；

(2)通常用一元线性回归方程预测的pb\_predict值,在投资者愿意为成长性更好的上市公司付出更高的股价的假设下，可以认为pb\_predict值是该支股票没有被高估以及低估的真实的pb值。

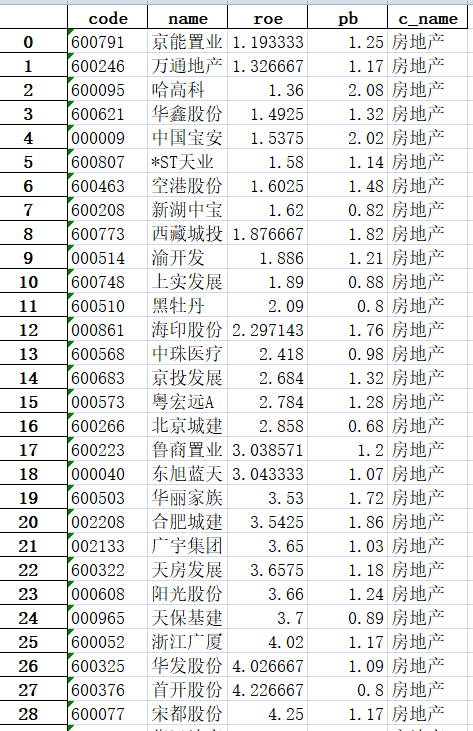
(3)基于第二点,任意的股票的pb值小于pb\_predict的是被低估的股票, 任意的股票的pb值大于pb\_predict的是被高估的股票, 任意的股票的pb值恰好等于pb\_predict的是未被高估或者低估的股票。

4． 案例

1. 房地产行业的pb\_roe模型



此处选择roe>0的股票进行筛选,筛选出被低估的股票截图如下，详细的股票信息见house.xlsx；



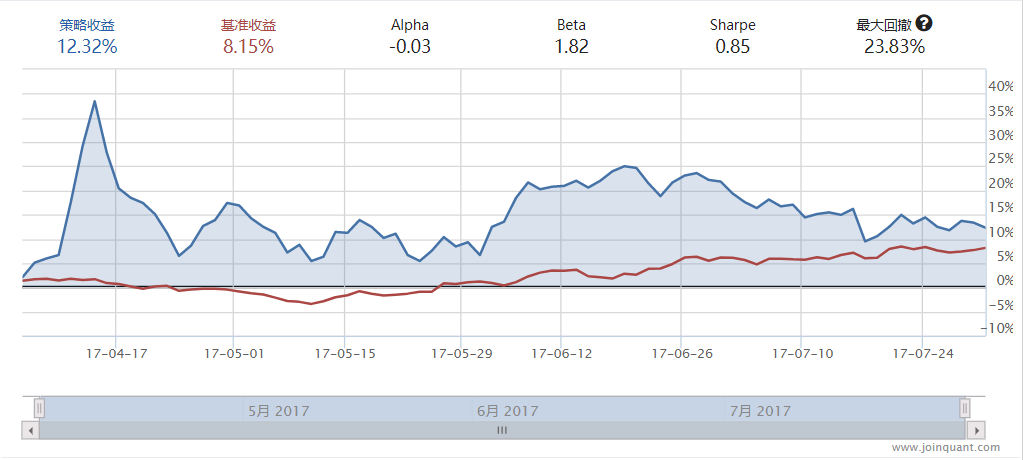
回测:

时间：2017-04-01~2017-07-31

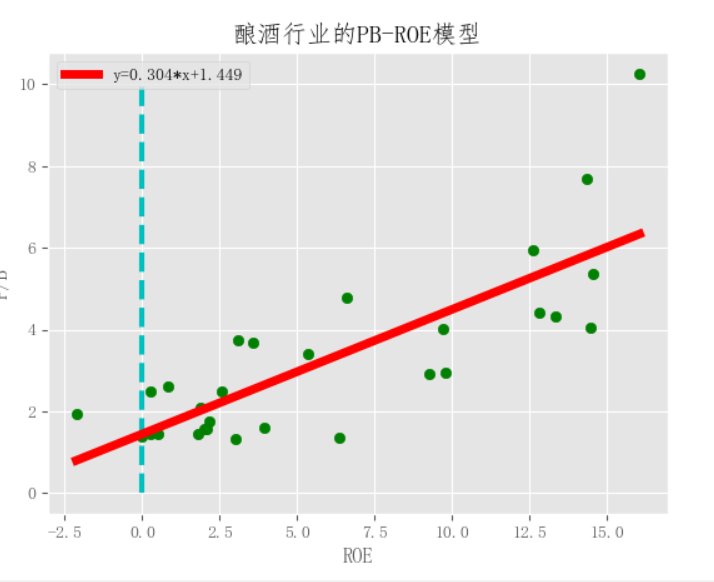
资金:100万元人民币  
权重:等权重

股票池:以上所有股票

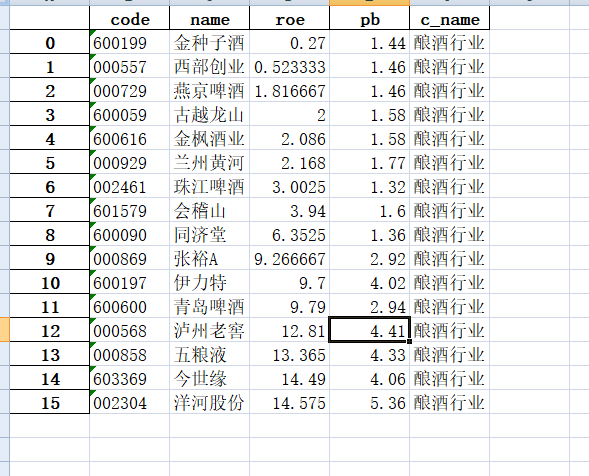
房地产行业被低估的股票组合,回测结果如下:



1. 酿酒行业的pb-roe模型



在roe阈值为0的情况下,酿酒行业被低估的股票,详细的股票信息见wine.xlsx如下:



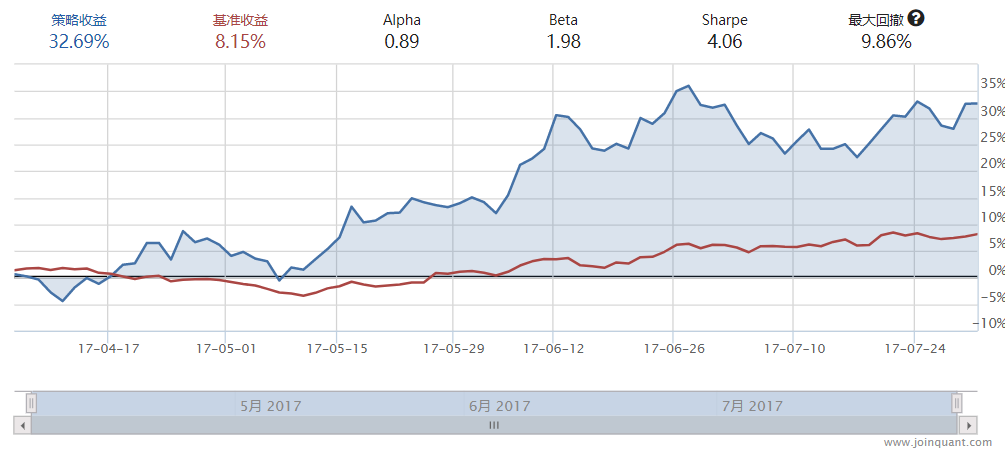
回测:

时间：2017-04-01~2017-07-31

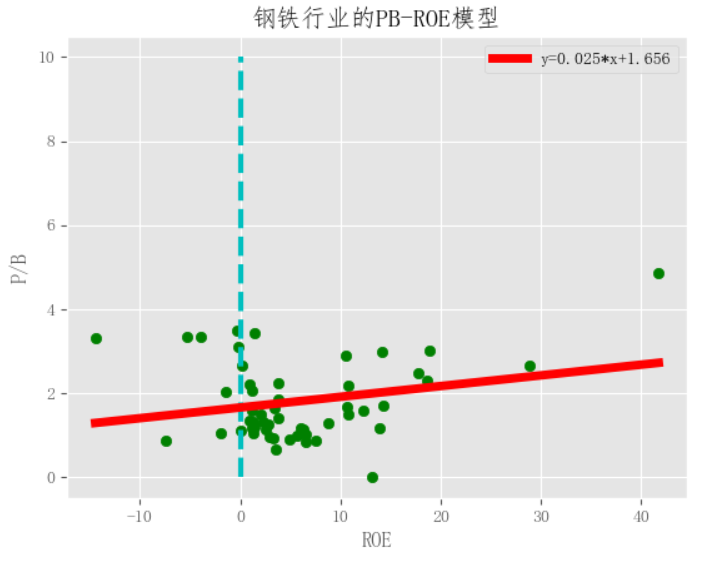
资金:100万元人民币  
权重:等权重

股票池:以上所有股票

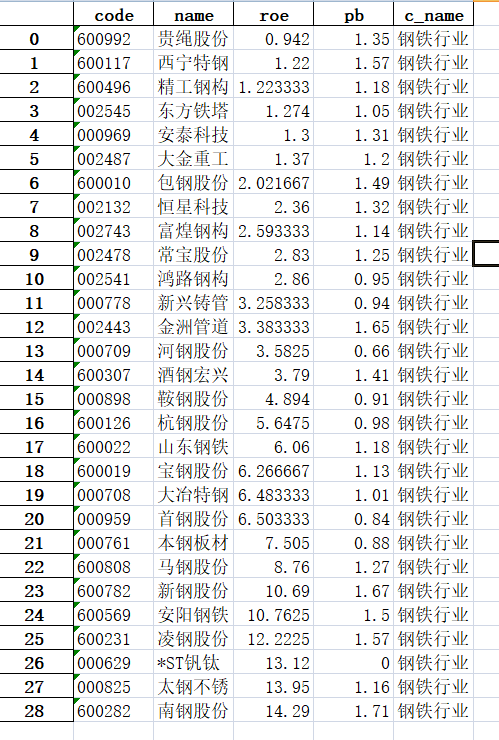
酿酒行业被低估的股票组合,回测结果如下:



1. 钢铁行业的pb-roe模型



在roe阈值为0的情况下,钢铁行业被低估的股票,详细的股票信息见gang.xlsx如下:



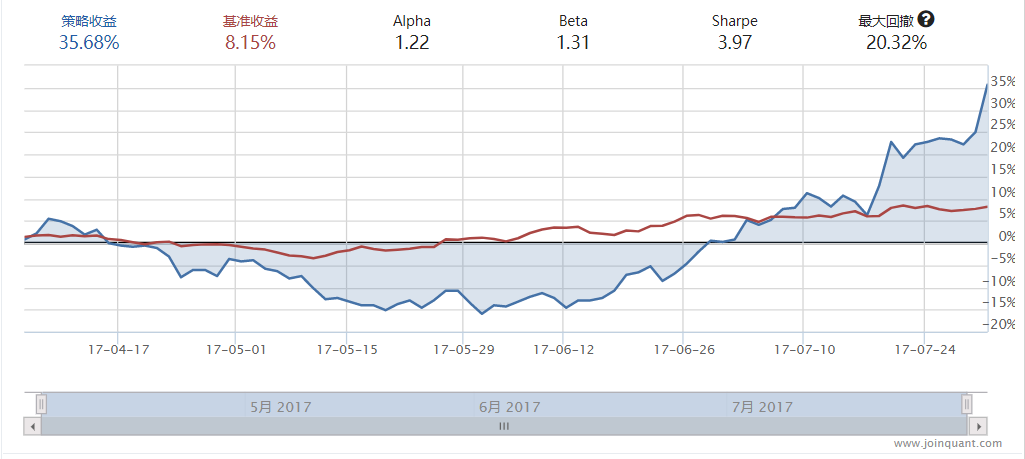
回测:

时间：2017-04-01~2017-07-31

资金:100万元人民币  
权重:等权重

股票池:以上所有股票

房地产行业被低估的股票组合,回测结果如下:



实现的代码如下:

import pandas as pd  
import numpy as np  
import pickle  
import tushare as ts  
from sklearn.linear\_model import LinearRegression  
import matplotlib.pyplot as plt  
from pylab import mpl  
import matplotlib  
from sklearn.preprocessing import MinMaxScaler  
mpl.rcParams['font.sans-serif'] = ['FangSong'] # 指定默认字体  
mpl.rcParams['axes.unicode\_minus'] = False # 解决保存图像是负号'-'显示为方块的问题  
  
class PB\_ROE():  
 def \_\_init\_\_(self,c\_name,score):  
 self.c\_name=c\_name  
 self.score=score  
  
 def get\_data(self,year,quarter):  
 data=ts.get\_stock\_basics()  
 data\_roe=pd.DataFrame()  
 for i in range(4):  
 quarter+=1  
 data\_roe=pd.concat((data\_roe,ts.get\_report\_data(year, quarter)))  
 self.save\_data(data,data\_roe)  
  
 def save\_data(self,\*datas):  
 num=0  
 for data in datas:  
 num+=1  
 with open('data/{}.pkl'.format(str(num)),'wb') as f:  
 pickle.dump(data,f)  
  
 # 近3年的数据求均值代替最终值  
 def roe\_mean\_deal(self):  
 with open('data/2.pkl','rb') as f:  
 data=pickle.load(f)  
 data=data[['code','name','roe']]  
 cols = data.columns  
 all\_final\_data = pd.DataFrame()  
 # 对每一只股票近三年的数据进行数据清洗后求均值代替当前的  
 for code, code\_data in data.groupby('code'):  
 code\_data\_array = code\_data.values[:,-1].reshape(-1,1).mean(axis=0)  
 final\_data\_array = np.hstack((code\_data.values[0,:2], code\_data\_array))  
 # 构建DataFrame格式的数据  
 final\_data = pd.DataFrame(final\_data\_array.reshape(1, -1), columns=cols)  
 final\_data=final\_data.fillna(0)  
 # 返回最后的数据  
 all\_final\_data = pd.concat((all\_final\_data, final\_data))  
 return all\_final\_data  
  
 def pb\_roe\_data(self):  
 with open('data/1.pkl', 'rb') as f:  
 data = pickle.load(f)  
 # data=data[['code','name','pb']]  
 roe\_data=self.roe\_mean\_deal()  
 M=[]  
 for code in roe\_data['code']:  
 M.append(data['pb'][code])  
 roe\_data['pb']=M  
 return roe\_data  
  
 # 保存行业数据  
 def get\_industry\_data(self):  
 data=ts.get\_industry\_classified()  
 with open('data/data\_industry.pkl','wb') as f:  
 pickle.dump(data,f)  
 return 'ok'  
  
 # 将股票的行业数据放在最后一列，行业分组,根据用户所要进行的行业，将相应的数据进行保存。  
 def industry\_pb\_roe\_data(self):  
 pb\_roe\_data=self.pb\_roe\_data()  
 with open('data/data\_industry.pkl','rb') as f:  
 industry\_data=pickle.load(f)  
 industry\_data=industry\_data.drop\_duplicates('code')  
 industry\_data=pd.DataFrame(industry\_data.values[:,1:],index=industry\_data['code'],columns=industry\_data.columns[1:])  
 M,W=[],[]  
 # 行业数据放在最后一列  
 for code in pb\_roe\_data['code']:  
 try:  
 M.append(industry\_data['c\_name'][code])  
 except KeyError:  
 M.append(np.nan)  
 # 行业数据在最后一列  
 pb\_roe\_data['c\_name']=M  
 # 将用户想要研究的行业里面的股票放在列表里面  
 for stock in pb\_roe\_data.values:  
 if self.c\_name==stock[-1]:  
 W.append(stock)  
 # 列表转换为DataFrame的数据格式  
 industry\_pb\_row\_data=pd.DataFrame(W,columns=pb\_roe\_data.columns)  
 # 剔除最小值与最大值  
 industry\_pb\_row\_data=industry\_pb\_row\_data.sort\_values(by='pb').iloc[1:-1]  
 industry\_pb\_row\_data = industry\_pb\_row\_data.sort\_values(by='roe').iloc[1:-1]  
 return industry\_pb\_row\_data  
  
 def main(self):  
 # 数据的准备  
 data=self.industry\_pb\_roe\_data()  
 roe\_data=data.values[:,2].reshape(-1,1)  
 pb\_data=data.values[:,3].reshape(-1,1)  
 # roe\_data = MinMaxScaler([0, 10]).fit\_transform(roe\_data)  
 # pb\_data = MinMaxScaler([0, 10]).fit\_transform(pb\_data)  
 # 线性回归模型  
 model=LinearRegression()  
 model.fit(roe\_data,pb\_data)  
 # 回归系数  
 coef='%.3f'%float(model.coef\_)  
 # 回归方程常数项  
 inter='%.3f'%float(model.intercept\_)  
 # 预测值  
 y\_pre=model.predict(roe\_data)  
 # 将被低估或者被高估的结果保存在excel里面  
 self.lower\_or\_upper(data,y\_pre)  
 # 绘图  
 self.get\_image(roe\_data,pb\_data,y\_pre,coef,inter)  
 return 'ok'  
  
 def get\_image(self, roe\_data,pb\_data,y\_pre,coef,inter):  
 matplotlib.style.use('ggplot')  
 plt.title('{}的PB-ROE模型'.format(self.c\_name))  
 plt.ylabel('P/B')  
 plt.xlabel('ROE')  
 plt.scatter(roe\_data, pb\_data, c='g')  
 plt.plot(roe\_data, y\_pre, c='red', linewidth=5,label='y=%s\*x+%s'%(coef,inter))  
 plt.vlines(self.score,0,10, colors="c", linestyles="dashed",linewidth=3)  
 plt.legend()  
 plt.show()  
  
 def lower\_or\_upper(self,data,y\_pre):  
 cols=data.columns  
 data\_array=data.values  
 pb\_data=data\_array[:,3].reshape(-1,1)  
 M,M1,W,W1=[],[],[],[]  
 for i in range(len(cols)):  
 M.append(data\_array[:,i].reshape(-1,1)[[pb\_data - y\_pre.reshape(-1, 1) < 0]])  
 lower\_data=pd.DataFrame(np.array(M).T,columns=cols)  
 # 将满足大于用户指定的roe最小值筛选出来  
 for stock\_low in lower\_data.values:  
 if stock\_low[2]>self.score:  
 M1.append(stock\_low)  
 try:  
 new\_lower\_data=pd.DataFrame(np.array(M1),columns=cols)  
 # 用户指定的roe, 剔除roe小于该得分的股票  
 # 保存被低估的股票  
 new\_lower\_data.to\_excel('lower\_stock.xlsx')  
 except:  
 print('请给出较小的roe值')  
  
 # 将满足大于用户指定的roe最小值筛选出来  
 for j in range(len(cols)):  
 W.append(data\_array[:,j].reshape(-1, 1)[[pb\_data - y\_pre.reshape(-1, 1) > 0]])  
 upper\_data = pd.DataFrame(np.array(W).T, columns=cols)  
 for stock\_upper in upper\_data.values:  
 if stock\_upper[2] > self.score:  
 W1.append(stock\_upper)  
 try:  
 new\_upper\_data = pd.DataFrame(np.array(W1), columns=cols)  
 # 存在被高估的股票  
 new\_upper\_data.to\_excel('upper\_stock.xlsx')  
 except:  
 print('没有满足的股票，请调整roe')  
 return 'ok'  
  
if \_\_name\_\_=='\_\_main\_\_':  
 # 根据不同行业设置不同的阈值,大于该阈值的值是我们要进行判别高估或者低估的股票。  
 a=PB\_ROE('钢铁行业',0)  
 a.main()  
 # print(a.final\_regress\_model())  
  
# 0 综合行业  
# 33 公路桥梁  
# 53 化纤行业  
# 79 机械行业  
# 290 生物制药  
# 445 石油行业  
# 469 玻璃行业  
# 488 仪器仪表  
# 536 交通运输  
# 623 飞机制造  
# 637 农林牧渔  
# 701 建筑建材  
# 792 塑料制品  
# 825 商业百货  
# 918 纺织行业  
# 960 医疗器械  
# 991 有色金属  
# 1063 供水供气  
# 1088 发电设备  
# 1153 造纸行业  
# 1177 船舶制造  
# 1185 煤炭行业  
# 1226 食品行业  
# 1284 陶瓷行业  
# 1292 纺织机械  
# 1300 钢铁行业  
# 1360 环保行业  
# 1387 酿酒行业  
# 1420 次新股  
# 1472 电器行业  
# 1530 传媒娱乐  
# 1570 化工行业  
# 1720 房地产  
# 1843 金融行业  
# 1894 其它行业  
# 2077 开发区  
# 2087 电子信息  
# 2334 服装鞋类  
# 2383 电子器件  
# 2535 电力行业  
# 2597 汽车制造  
# 2700 家具行业  
# 2716 农药化肥  
# 2762 酒店旅游  
# 2798 水泥行业  
# 2824 物资外贸  
# 2845 摩托车  
# 2851 印刷包装  
# 2871 家电行业