现在的A股能买吗?

因子实战 第七集

如何用量化手段选择 抄底A股的时机?

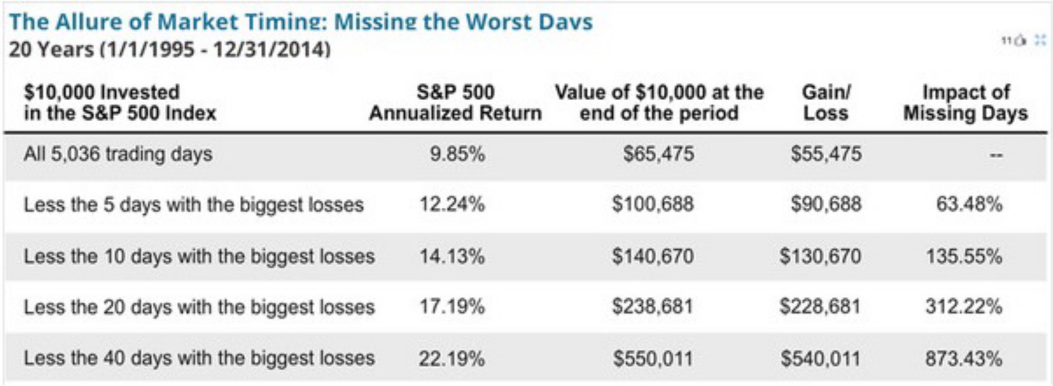
🎬 @大导演哈罗德

🏛 香港中文大学 金融工程 本科

📈 即将前往美国金融工程硕士之路（已经获得录取）

🌐 关注我的Bilibili，看所有人都能听得懂的量化学习内容！

🌟🌟🌟 量化不是束之高阁的灵丹妙药，量化是让散户投资者认识市场的工具 #哈罗德的量化频道 🌟🌟🌟



The Allure of Market Timing: Missing the Worst Days
20 Years (1/1/1995 - 12/31/2014)

$10,000 Invested in the S&P 500 Index	S&P 500 Annualized Return	Value of $10,000 at the end of the period	Gain/ Loss	Impact of Missing Days
All 5,036 trading days	9.85%	$65,475	$55,475	--
Less the 5 days with the biggest losses	12.24%	$100,688	$90,688	63.48%
Less the 10 days with the biggest losses	14.13%	$140,670	$130,670	135.55%
Less the 20 days with the biggest losses	17.19%	$238,681	$228,681	312.22%
Less the 40 days with the biggest losses	22.19%	$550,011	$540,011	873.43%

结论：如果能在正确时间进入市场，可以获得惊人的超额收益

但是如果进入市场的时间不对呢...?

In [ ]:
```python
import pandas as pd
import numpy as np
from scipy.stats import percentileofscore

# 读取 .pkl 文件
data = pd.read_pickle('Example.pkl')
grouped_df = data.groupby('S_INFO_WINDCODE')
```

In [ ]:
```python
data
```

Out[ ]:

	S_INFO_WINDCODE	TRADE_DT	CRNCY_CODE	S_VAL_MV	S_DQ_MV	S_PQ_HIGH_52W_	S_PQ_LOW_52W_
13352774	603917.SH	20180320	CNY	273168.0	68292.0	32.99	20.48
13352775	603917.SH	20180321	CNY	267456.0	66864.0	32.99	20.48
13352776	603917.SH	20180322	CNY	278656.0	69664.0	32.99	20.48
13352777	603917.SH	20180323	CNY	250768.0	62692.0	32.99	20.48
13352778	603917.SH	20180324	CNY	250768.0	62692.0	32.99	20.48

5 rows × 36 columns

In [ ]:

```
data.info()
```

In [ ]:

```
data.columns
```

Out[ ]:

```
Index(['S_INFO_WINDCODE', 'TRADE_DT', 'CRNCY_CODE', 'S_VAL_MV', 'S_DQ_MV',
       'S_PQ_HIGH_52W_', 'S_PQ_LOW_52W_', 'S_VAL_PE', 'S_VAL_PB_NEW',
       'S_VAL_PE_TTM', 'S_VAL_PCF_OCF', 'S_VAL_PCF_OCFTTM', 'S_VAL_PCF_NCF',
       'S_VAL_PCF_NCFTTM', 'S_VAL_PS', 'S_VAL_PS_TTM', 'S_DQ_TURN',
       'S_DQ_FREETURNOVER', 'TOT_SHR_TODAY', 'FLOAT_A_SHR_TODAY',
       'S_DQ_CLOSE_TODAY', 'S_PRICE_DIV_DPS', 'S_PQ_ADJHIGH_52W',
       'S_PQ_ADJLOW_52W', 'FREE_SHARES_TODAY', 'NET_PROFIT_PARENT_COMP_TTM',
       'NET_PROFIT_PARENT_COMP_LYR', 'NET_ASSETS_TODAY',
       'NET_CASH_FLOWS_OPER_ACT_TTM', 'NET_CASH_FLOWS_OPER_ACT_LYR',
       'OPER_REV_TTM', 'OPER_REV_LYR', 'NET_INCR_CASH_CASH_EQU_TTM',
       'NET_INCR_CASH_CASH_EQU_LYR', 'UP_DOWN_LIMIT_STATUS',
       'LOWEST_HIGHEST_STATUS'],
      dtype='object')
```

#

In [ ]:

```
riskfree = pd.read_excel('中国_10年期国债收益率.xlsx', names=['date', 'interest'])
riskfree = riskfree.drop(0).reset_index()
riskfree['date'] = pd.to_datetime(riskfree['date'])
riskfree['date'] = riskfree['date'].dt.strftime('%Y%m%d')

closeprice = pd.read_pickle('IndexQuote_ClosePrice.txt')

closeprice_500 = closeprice['000905.SH'].loc['20181030':'20221129']

contents_000905  = pd.read_pickle('IndexComponent_000905.txt')
contents_000905 = contents_000905.sort_index()

trade_date_list_000905 = list(contents_000905.index)

start_date = '20110101'
end_date = '20230531'
```

In [ ]:

```
riskfree
```

Out[ ]:

	index	date	interest
0	1	20150104	3.6405
1	2	20150105	3.6406
2	3	20150106	3.6456
3	4	20150107	3.6505
4	5	20150108	3.6457
...	...	...	...
2106	2107	20230613	2.6583
2107	2108	20230614	2.6353
2108	2109	20230615	2.6654
2109	2110	20230616	2.7033
2110	2111	20230619	2.7145

2111 rows × 3 columns

In [ ]:

```
contents_000905
```

Out[ ]:

	000005.SZ	000006.SZ	000008.SZ	000009.SZ	000012.SZ	000016.SZ	000021.SZ	000025.SZ	000027.S
20040102	0	1	0	0	0	1	0	0	
20040105	0	1	0	0	0	1	0	0	
20040106	0	1	0	0	0	1	0	0	
20040107	0	1	0	0	0	1	0	0	
20040108	0	1	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	.
20230118	0	0	0	1	1	0	1	0	
20230119	0	0	0	1	1	0	1	0	
20230120	0	0	0	1	1	0	1	0	
20230130	0	0	0	1	1	0	1	0	
20230131	0	0	0	1	1	0	1	0	

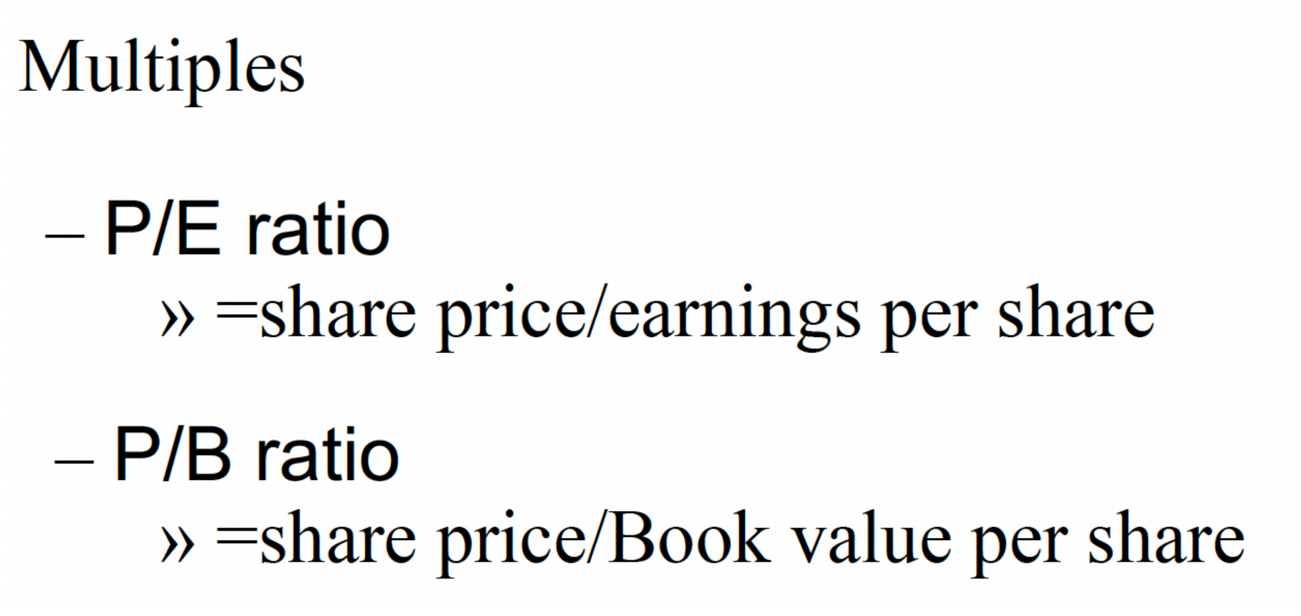
4633 rows × 1600 columns

In [ ]:
```
closeprice
```

Out[ ]:

	000001.SH	000016.SH	000063.SH	000300.SH	000827.SH	000852.SH	000903.SH	000905.SH	000906
19500103	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
19500104	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
19500105	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
19500106	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
19500109	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	
20230118	3224.4060	2810.0860	2950.8459	4130.3143	2244.6876	6613.0050	4018.3248	6151.4557	4464.3
20230119	3240.2794	2820.8067	2964.9399	4156.0077	2234.3433	6676.0803	4046.8916	6204.3261	4494.8
20230120	3264.8138	2836.8070	2986.4187	4181.5267	2279.2669	6736.8515	4079.3644	6251.4337	4524.0
20230130	3269.3180	2839.1216	2978.3743	4201.3450	2313.6305	6786.4530	4100.8178	6283.3272	4545.9
20230131	3255.6692	2807.9519	2963.7905	4156.8578	2317.8727	6805.2756	4062.4385	6289.1500	4510.9

19468 rows × 67 columns



Multiples

- P/E ratio
  - » =share price/earnings per share
- P/B ratio
  - » =share price/Book value per share

In [ ]:

```
contents_000905 = contents_000905.loc['20181030':'20201030']

# 这里的每一天指的是交易日
every_day_dict_500 = {}

for i in range(0,  len(contents_000905)):

    # 找出对应的那天
    date = contents_000905.iloc[i].name

    stocks = [column for column in contents_000905.columns if contents_000905.iloc[i][column]]

    market_value_sum = 0

    PB_net_value_sum = 0

    PE_NET_PROFIT_parent = 0

    for stock in stocks:

        try:
            current_stock = grouped_df.get_group(stock).sort_values('TRADE_DT')
        except:
            continue

        try:
            date_index = current_stock[current_stock['TRADE_DT'] == date].index[0]
        except:
            continue


        S_VAL_MV = current_stock.loc[date_index, 'S_VAL_MV'] * 10000

        NET_ASSETS_TODAY = current_stock.loc[date_index, 'NET_ASSETS_TODAY']

        NET_PROFIT_PARENT_COMP_TTM = current_stock.loc[date_index, 'NET_PROFIT_PARENT_COMP_TTM']

        market_value_sum += S_VAL_MV

        PB_net_value_sum += NET_ASSETS_TODAY

        PE_NET_PROFIT_parent += NET_PROFIT_PARENT_COMP_TTM

    riskfree_ = riskfree.loc[riskfree['date'] == date, 'interest'].values[0]

    PB = market_value_sum / PB_net_value_sum

    PE = market_value_sum / PE_NET_PROFIT_parent

    PBPE = (PB * PE) ** 0.5

    RISK_PREMIERE = 1/PE - riskfree_

    every_day_dict_500[date] = [PB,PE,PBPE, RISK_PREMIERE]
```

In [ ]:

```
# Convert the dictionary into a DataFrame
df_500 = pd.DataFrame.from_dict(every_day_dict_500, orient='index', columns=['PB', 'PE', 'PBPE', 'RI
```

In [ ]:

```
df_500
```

Out[ ]:

	PB	PE	PBPE	RISK_PREMIERE
20181030	1.553238	17.577739	5.225170	-3.464010
20181031	1.577496	18.278784	5.369796	-3.463792
20181101	1.587840	18.398638	5.405006	-3.440648
20181102	1.638703	18.988664	5.578241	-3.469537
20181105	1.638643	18.987659	5.577993	-3.486134
...	...	...	...	...
20201026	2.083567	31.352610	8.082405	-3.192005
20201027	2.085930	31.400594	8.093173	-3.180053
20201028	2.088107	31.344075	8.090103	-3.162496
20201029	2.085904	31.225431	8.070517	-3.174675
20201030	2.030671	30.282798	7.841837	-3.191278

487 rows × 4 columns

In [ ]:

```
closeprice_500
```

Out[ ]:

```
20181030    4204.5424
20181031    4272.5518
20181101    4298.9837
20181102    4437.9456
20181104    4437.9456
              ...
20221123    6115.6049
20221124    6120.0659
20221125    6087.8591
20221128    6055.2843
20221129    6163.8214
Name: 000905.SH, Length: 1087, dtype: float64
```

In [ ]:

```
df = df_500

dfcloseprice = pd.DataFrame(closeprice)
dfcloseprice = closeprice_500
# print(dfcloseprice)
# 这里我们可以理解为工作日
DAY = 50
XDAYLATER3 = 20

bottom_thresholds = [1, 3, 5, 8, 10]
top_thresholds = [99, 97, 95, 92, 90]
bottom_signals = {}
top_signals = {}
results = {}

for i in range(DAY, len(df)):
    # end is like 20221124
    end = df.iloc[i].name

    interval_data = df.loc[:end][-DAY:]

    today_pb = df.iloc[i]['PB']
    pb_percentile = percentileofscore(interval_data['PB'], today_pb)

    today_pe = df.iloc[i]['PE']
    pe_percentile = percentileofscore(interval_data['PE'], today_pe)

    today_pbpe = df.iloc[i]['PBPE']
    pbpe_percentile = percentileofscore(interval_data['PBPE'], today_pbpe)

    today_risk_premiere = df.iloc[i]['RISK_PREMIERE']
    risk_premiere_percentile = percentileofscore(interval_data['RISK_PREMIERE'], today_risk_premiere

    index = dfcloseprice.index.get_loc(str(end)) + XDAYLATER3

    first_price = dfcloseprice.iloc[index-20]
    try:
        # Fetch the price XDAYLATER3 days later
        end_price = dfcloseprice.iloc[index + XDAYLATER3]
    except:
        break


    # Check if any percentiles exceed the top thresholds
    for threshold in top_thresholds:
        if end_price is None:
            state = 'None'
        elif first_price > end_price:
            state = 1
        else:
            state = 0


        if pb_percentile > threshold:
            top_signals.setdefault(threshold, []).append((end, threshold, pb_percentile, today_pb, '
            # break
        if pe_percentile > threshold:
            top_signals.setdefault(threshold, []).append((end, threshold, pe_percentile, today_pe, '
            # break
        if pbpe_percentile > threshold:
            top_signals.setdefault(threshold, []).append((end, threshold,pbpe_percentile, today_pbpe
            # break
        if risk_premiere_percentile > threshold:
            top_signals.setdefault(threshold, []).append((end, threshold,risk_premiere_percentile, t


    # Check if any percentiles fall below the bottom thresholds
    for threshold in bottom_thresholds:
        if end_price is None:
            state = 'None'
        elif first_price < end_price:
            state = 1
        else:
            state = 0

        # 观察是否触发
        if pb_percentile < threshold:
            bottom_signals.setdefault(threshold, []).append((end, threshold,pb_percentile, today_pb
            # break
        if pe_percentile < threshold:
            bottom_signals.setdefault(threshold, []).append((end, threshold,pe_percentile, today_pe
            # break
        if pbpe_percentile < threshold:
            bottom_signals.setdefault(threshold, []).append((end, threshold,pbpe_percentile, today_p
            # break
        if risk_premiere_percentile < threshold:
            bottom_signals.setdefault(threshold, []).append((end, threshold,risk_premiere_percentile


combined_signals = {}
combined_signals.update(top_signals)
combined_signals.update(bottom_signals)
```

In [ ]:

```
rows = []
for threshold, signals in combined_signals.items():
    for signal in signals:
        # try:
        row = [signal[0], signal[1], signal[2], signal[3], signal[4],signal[5]]
        # except:
        #     row = [signal[0], signal[1], signal[2], signal[3], signal[4]]
        rows.append(row)

# Define the header row
header = ['Date', 'Threshold', 'Percentile', 'Value', 'Factor','20daystate']
df_result = pd.DataFrame(rows,columns=header)
```

In [ ]:

```
df_result
```

Out[ ]:

	Date	Threshold	Percentile	Value	Factor	20daystate
0	20190110	99	100.0	-3.048929	RISK_PREMIERE	0
1	20190111	99	100.0	-3.044560	RISK_PREMIERE	0
2	20190116	99	100.0	-3.040421	RISK_PREMIERE	0
3	20190117	99	100.0	-3.001369	RISK_PREMIERE	0
4	20190214	99	100.0	1.679125	PB	0
...	...	...	...	...	...	...
2096	20201029	8	6.0	31.225431	PE	1
2097	20201029	8	2.0	8.070517	PBPE	1
2098	20201030	8	2.0	2.030671	PB	1
2099	20201030	8	2.0	30.282798	PE	1
2100	20201030	8	2.0	7.841837	PBPE	1

2101 rows × 6 columns

In [ ]:

```
results = []

for group_name, group_data in df_result.groupby(['Threshold','Factor']):
    percentage3 = group_data['20daystate'].astype(bool).mean() * 100
    count = len(group_data)
    results.append({'indicator': group_name, 'winning ratio': percentage3,'time': count})

percentage_df = pd.DataFrame(results)
percentage_df = percentage_df.reset_index(drop=True)
percentage_df = percentage_df.sort_values(by= 'winning ratio', ascending= False)
```

In [ ]:

```
percentage_df
```

Out[ ]:

	indicator	winning ratio	time
4	(5, PB)	88.000000	25
0	(3, PB)	86.666667	15
8	(8, PB)	83.333333	36
12	(10, PB)	82.608696	46
6	(5, PE)	81.818182	11
1	(3, PBPE)	81.250000	16
2	(3, PE)	80.000000	10
5	(5, PBPE)	78.260870	23
14	(10, PE)	76.923077	26
9	(8, PBPE)	76.666667	30
10	(8, PE)	76.470588	17
13	(10, PBPE)	76.190476	42
18	(90, PE)	54.716981	106
30	(97, PE)	54.545455	77
22	(92, PE)	54.081633	98
26	(95, PE)	53.846154	91
21	(92, PBPE)	52.884615	104
34	(99, PE)	52.830189	53
17	(90, PBPE)	52.173913	115
25	(95, PBPE)	51.612903	93
29	(97, PBPE)	49.333333	75
33	(99, PBPE)	48.148148	54
16	(90, PB)	48.076923	104
7	(5, RISK_PREMIERE)	46.666667	60
15	(10, RISK_PREMIERE)	46.341463	82
11	(8, RISK_PREMIERE)	46.268657	67
20	(92, PB)	45.833333	96
3	(3, RISK_PREMIERE)	45.714286	35
24	(95, PB)	44.047619	84
28	(97, PB)	41.791045	67
32	(99, PB)	36.734694	49
19	(90, RISK_PREMIERE)	36.250000	80
23	(92, RISK_PREMIERE)	36.111111	72
27	(95, RISK_PREMIERE)	35.483871	62
35	(99, RISK_PREMIERE)	33.333333	33
31	(97, RISK_PREMIERE)	29.787234	47

In [ ]:

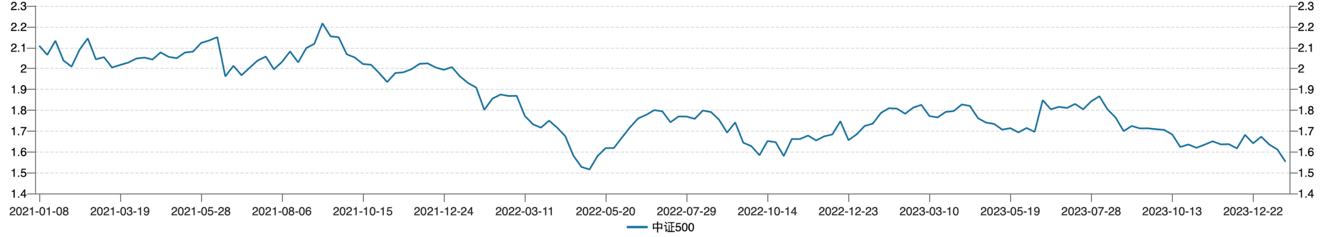
```
df_pd_nowadays = pd.read_excel('市场整体估值.xlsx',index_col=0)
df_pd_nowadays
```

In [ ]:

```
np.percentile(df_pd_nowadays.iloc[1:-3,-1], 5)
```

In [ ]:

```
np.percentile(df_pd_nowadays.iloc[:-3,-1], 10)
```



数据来源：Wind

太低了！可以买了！

问题:

你敢跟吗?

结论：

这个世界上有两种投资人：第一种是不知道股市往哪里走的，第二种是不知道他们自己不知道股市的走向的。但是事实上还有第三种人：他们靠假装可以预测股市的走向来骗吃骗喝。

In this world, there are two types of investors: the first type doesn't know where the stock market is headed, and the second type doesn't know they don't know where the stock market is headed. But in reality, there is a third type of person: those who deceive and manipulate by pretending they can predict the direction of the stock market to benefit themselves.