『1%の人が知っている99%勝てる株が見つかる本』を検証する

検証方法

p67記載のステップ①~⑤のうち、機械的に抽出できるステップ①~③に該当する銘柄を抽出する

抽出期間：2009～2013年

株価リターン計測期間：2013年12月30日～2018年12月2日

In [108]:

```
# 必要なライブラリをインポートする
import numpy as np
import pandas as pd
from IPython.display import HTML

%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['font.family'] = 'Yu Mincho'


from pandas import Series, DataFrame
pd.options.display.max_columns = 999
from datetime import datetime

# from scipy.ndimage.interpolation import shift

from IPython.display import display

FONT_NAME = 'TakaoPGothic'
mpl.rcParams["font.family"] = FONT_NAME

df = pd.read_pickle("findata.pickle")
# df["securityCode"] = pd.to_numeric(df["securityCode"])
df["year"] = df["fiscalYearEndDate"].dt.year
```

In [109]:

```

# 営業費用の系列をつくる
df['営業費用'] = df['共通_サマリー_売上高'] - df['共通_サマリー_営業利益']
# 費用あたり売上の系列（売上/営業費用）をつくる
df['費用あたり売上'] = df['共通_サマリー_売上高'] / df['営業費用']
```

===== ステップ1　費用あたり売上が1.15倍以上 =====

In [110]:

```
df2013 = df.query("year=='2013'")
df2013_1 = df2013.query('費用あたり売上 > 1.15')
df2013_1['securityCode'] = df2013_1['securityCode'].astype(str)
df2013_1['securityCode'] = df2013_1['securityCode'].str.strip()
rieki_list = df2013_1['securityCode']
len(rieki_list)
```

```
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/in
dexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/i
ndexing.html#indexing-view-versus-copy)
  after removing the cwd from sys.path.
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:5: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/in
dexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/i
ndexing.html#indexing-view-versus-copy)
  """
```

Out[110]:

```
346
```

ステップ1では346企業を抽出

===== ステップ2　配当性向5割以上 =====

In [111]:

```
df_toyo = pd.read_pickle("toyokeizai_fin_data.pickle")
```

In [113]:

```
df_toyo2 = df_toyo[['securityCode', 'fiscalYearEndDate', '共通_サマリー_潜在１株益', '共通_サマリー_
df_toyo2['securityCode'] = df_toyo2['securityCode'].astype(str)
df_toyo2['securityCode'] = df_toyo2['securityCode'].str.strip()
df_toyo2['fiscalYearEndDate'] = pd.to_datetime(df_toyo2['fiscalYearEndDate'])
df_toyo2['配当性向'] = df_toyo2['共通_サマリー_ＤＰＳ'] / df_toyo2['共通_サマリー_潜在１株益']
df_toyo3 = df_toyo2[df_toyo2['fiscalYearEndDate'].dt.year == 2013]
df_toyo4 = df_toyo3[df_toyo3['配当性向'] > 0.5]
df_toyo5 = df_toyo4[df_toyo4['securityCode'].isin(rieki_list)]
df_toyo5
dps_list = df_toyo5['securityCode'].tolist()
len(dps_list)
```

```
indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy)

C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:3: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports until
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:4: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/
indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy)
```

ステップ2では63企業を抽出

===== ステップ3　4年連続増収 =====

In [114]:

```
df['securityCode'] = df['securityCode'].astype(str)
df['securityCode'] = df['securityCode'].str.strip()
#df.set_index('securityCode')
df2 = df[['securityCode', 'fiscalYearEndDate', '共通_サマリー_売上高', 'year']]
yearlist = ['2009', '2010', '2011', '2012', '2013']
df3 = df2[df2['year'].isin(yearlist)]
df4 = df3[df3['securityCode'].isin(rieki_list)]
#df4.set_index('securityCode')

df5 = df4.groupby(by='securityCode')['共通_サマリー_売上高'].rolling(2).apply(lambda x:x[1] / x[0]).
df6 = df5[df5 > 1]
df7 = df6.dropna()
zenlist = df7['securityCode'].value_counts()
zenlist2 = zenlist[zenlist ==4]
zenlist3 = zenlist2.index.tolist()
len(zenlist3)
```

```
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:10: FutureWarning:
Currently, 'apply' passes the values as ndarrays to the applied function. In the fut
ure, this will change to passing it as Series objects. You need to specify 'raw=Tru
e' to keep the current behaviour, and you can pass 'raw=False' to silence this warni
ng
  # Remove the CWD from sys.path while we load stuff.
```

Out[114]:

```
79
```

ステップ3では79企業を抽出

ステップ2と3で重複している企業を除外し、該当企業をまとめる

In [115]:

```
zenlist3.extend(dps_list)
len(zenlist3)
kateru_list = list(set(zenlist3))
len(kateru_list)
```

Out[115]:

```
107
```

107社を抽出（63+79-107=35社が重複していた）

市場全体と条件該当の107社のリターンを比較する

In [116]:

```
kabu = pd.read_pickle('kabuka.pickle')
kabu['CALENDAR_DATE'] = pd.to_datetime(kabu['CALENDAR_DATE'])
kabu['SECURITY_CODE'] = kabu['SECURITY_CODE'].astype(str)
kabu['SECURITY_CODE'] = kabu['SECURITY_CODE'].str.strip()
date_list = ['2013-12-30', '2018-12-04']
kabu1 = kabu[kabu['CALENDAR_DATE'].isin(date_list)]
kabu1.tail()
```

Out[116]:

	SECURITY_CODE	CALENDAR_DATE	adjusted_price	adjusted_dps	DIV_PROFIT_GAIN
1121422	9995	2018-12-04	442.000000	12.0	2.714932
1121654	9996	2013-12-30	818.747191	24.0	2.605863
1121714	9996	2018-12-04	1720.000000	30.0	1.744186
1121946	9997	2013-12-30	465.539872	7.5	1.473477
1122006	9997	2018-12-04	1171.000000	12.5	1.067464

In [117]:

```
## 現在東証に上場している企業に絞る
df_jpx = pd.read_excel('jpx_data.xls')
df_jpx['SECURITY_CODE'] = df_jpx['SECURITY_CODE'].astype(str)
df_jpx['SECURITY_CODE'] = df_jpx['SECURITY_CODE'].str.strip()
joujo = df_jpx['SECURITY_CODE']
kabu2 = kabu1[kabu1['SECURITY_CODE'].isin(joujo)]
## 比較対象がある企業（新規上場等は除外）に絞る
kisuu = kabu2['SECURITY_CODE'].value_counts()
kisuu2 = kisuu[kisuu == 2].index.tolist()
kabu3 = kabu2[kabu2['SECURITY_CODE'].isin(kisuu2)]
```

In [100]:

```
kabu4 = kabu3.groupby(by='SECURITY_CODE')['adjusted_price'].rolling(2).apply(lambda x:x[1] / x[0]).
kabu5 = kabu4[kabu4['SECURITY_CODE'].isin(kateru_list)]
```

```
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:1: FutureWarning: C
urrently, 'apply' passes the values as ndarrays to the applied function. In the futu
re, this will change to passing it as Series objects. You need to specify 'raw=True'
to keep the current behaviour, and you can pass 'raw=False' to silence this warning
  """Entry point for launching an IPython kernel.
```

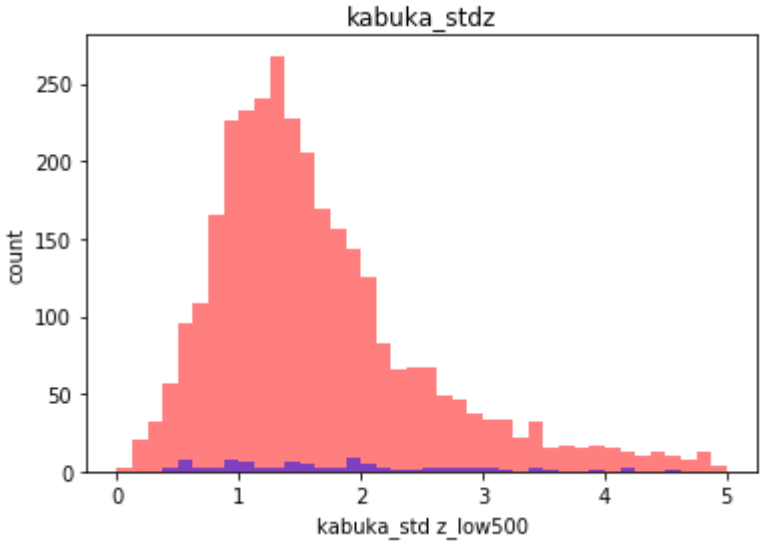
In [101]:

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)

ax.hist(kabu4['adjusted_price'], bins=40, range=(0, 5), color='red', alpha=0.5)
ax.hist(kabu5['adjusted_price'], bins=40, range=(0, 5), color='blue', alpha=0.5)
ax.set_title('kabuka_stdz')
ax.set_xlabel('kabuka_std z_low500')
ax.set_ylabel('count')

fig.show()
```

```
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥matplotlib¥figure.py:445: UserWarning: Ma
tplotlib is currently using module://ipykernel.pylab.backend_inline, which is a non-
GUI backend, so cannot show the figure.
  % get_backend())
C:¥Users¥legen¥Anaconda3¥lib¥site-packages¥matplotlib¥font_manager.py:1241: UserWarn
ing: findfont: Font family ['TakaoPGothic'] not found. Falling back to DejaVu Sans.
  (prop.get_family(), self.defaultFamily[fontext]))
```



In [102]:

```
kabu4['adjusted_price'].describe()
```

Out[102]:

```
count    3260.000000
mean        1.828724
std         1.546171
min         0.103187
25%         1.053175
50%         1.477861
75%         2.087419
max        28.908354
Name: adjusted_price, dtype: float64
```

In [103]:

```
kabu5['adjusted_price'].describe()
```

Out[103]:

```
count    96.000000
mean      2.247856
std       2.293881
min       0.388203
25%       1.037306
50%       1.747803
75%       2.568907
max      17.164143
Name: adjusted_price, dtype: float64
```

市場全体（該当3260社）の平均リターンは182%、中央値は148%に対し、条件抽出の96社の平均リターンは225%、中央値は175%

In [ ]:

```

```