Report of Maximum Return

Group: SMRMJ

Hengyu Shen 120090633 Kexuan Ma 120090651 Jun Gao 120090240

Xinyu Liu 120020128 Haoshen Zhang 120090798

1. Suggestion Message

Dear Rui,

It's our pleasure to offer you an evaluation report of a potential factor for investors. The factor maximum return (maxret) could provide a higher return than putting money into the market with high volatility straightly forward before the subprime crisis. We use Pandas in Python to analyze data for more than ninety thousand stocks since 1980 and use our maximum daily return (maxret) as a factor to conduct the CAPM and Fama-French three factors (FF3) model. Both models' alpha has a significant monotonic expression. However, the factor failed to catch up with the market after the subprime mortgage crisis, especially during the 2010s American economic expansion. Other limitations are also shown below.

Best regards,

Your team: SMRMJ

2. Background Introduction

2.1. Definitions

The definition of maximum return is relatively straightforward. In the original literature by Bali, Cakici, and Whitelaw (2011), the authors select the maximum return stocks using two methods. One is the single maximum daily return, which can choose the stock with the highest return in one day of a month; the other is to select the stock with the average highest return in a few days of one month. The result is robust in both methods, so for simplicity, we use the single maximum daily return to conduct our study as they do. In addition, they describe assets that have a relatively small probability of a significant pay-off as lottery-like assets. Investing in these assets is like gambling. Investors who get used to buying these assets are usually not well-diversified or rational. Error in the probability weighting of investors causes them to over-value stocks with a small probability of a significant positive return (Barberis and Huang, 2008).

2.2. Literature results

In Bali and his partners' paper, the authors add maxret into the ordinary CAPM model and Fama-French three-factor model (FF3) and find a negative and significant relationship between the maximum daily returns and expected stock returns. Their explanation for this phenomenon is that investors may be willing to pay more for stocks that exhibit extreme positive returns, and this explanation is consistent with two theories. The first is the cumulative theory, which says that errors in the probability weighting of investors cause them to over-value stocks with a small probability of a significant positive return. The second is the framework of the optimal belief, which means that agents optimally choose to distort their views about future possibilities to maximize their current utility. We can take the Chinese A-share market as an example. Many investors fall into this trap and become "jiucai" – being trapped in the market and suffering massive losses – successively. To explore the practicality of this factor in the present, we raise the study.

3. Data and Methodology

3.1. Database

We choose data from 93437 stocks since 1980, including their PERMNO, date, monthly return, market capitalization, and Maximum daily return. In particular, all maxret data are multiplied by -1 in advance. We also reference data of the original Fama-French three factors model from January 1980 to August 2021. The risk-free rate is provided in FF3's database. Therefore, our time frame to search is also from January 1980 to August 2021.

3.2. CAPM model

The CAPM model defines the ratio of the covariance between an asset's return and the market portfolio return to the variance of the market portfolio return as the asset's systematic risk. It is shown as:

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i (R_{m,t} - R_f) + \varepsilon_t$$

where R_i is the return on investment i, r_f is the risk-free rate, β_i is the systematic risk and R_m is the market return. We usually call $R_i - r_f$ as risk premium and $R_m - r_f$ as the market factor.

3.3. Fama-French three-factor model

To improve the accuracy of the CAPM model, the Fama-French three-factor model introduces two new factors: *SMB* shows the difference between small- and big- capitalization companies and HML shows the difference between high- and low- book value / market capitalization ratio firms.

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i (R_{m,t} - R_{f,t}) + s_i SMB_t + h_i HML_t + \varepsilon_{i,t}$$

3.4. Methodology

a) Divide American stocks into ten portfolios according to their past month's maximum return from small to large. Since there is a minus sign in front of the data, portfolio 0 is the group with the highest maximum return, and portfolio 9 has the lowest maximum return.

	PERMNO	ym	RET	mcap	maxret	allo
0	10000	Apr-86	-0.098592	1.63E+07	-0.145161	0
1	10000	May-86	-0.222656	1.52E+07	-0.022727	8
2	10000	Jun-86	-0.005025	1.18E+07	-0.115702	1
3	10000	Jul-86	-0.080808	1.17E+07	-0.042553	5
4	10000	Aug-86	-0.615385	1.08E+07	-0.116667	1
...	...	...	...	...	...	...
2381625	93436	Nov-21	0.027612	1.12E+12	-0.126616	0
2381626	93436	Dec-21	-0.076855	1.15E+12	-0.08491	2
2381627	93436	Jan-22	-0.113609	1.09E+12	-0.074947	3
2381628	93436	Feb-22	-0.070768	9.68E+11	-0.135317	0
2381629	93436	Mar-22	0.238009	9.00E+11	-0.074777	3

b) Calculate value-weighted returns of each portfolio.

	PERMNO	ym	RET	mcap	maxret	allo	totalmcap	weight	vw_ret
0	10000	Apr-86	-0.098592	1.63E+07	-0.145161	0	1.43E+10	0.001143	-0.000113
1	10000	May-86	-0.222656	1.52E+07	-0.022727	8	4.20E+11	0.000036	-0.000008
2	10000	Jun-86	-0.005025	1.18E+07	-0.115702	1	3.64E+10	0.000324	-0.000002
3	10000	Jul-86	-0.080808	1.17E+07	-0.042553	5	2.66E+11	0.000044	-0.000004
4	10000	Aug-86	-0.615385	1.08E+07	-0.116667	1	4.06E+10	0.000265	-0.000163
...	...	...	...	...	...	...	...	...	...
2381625	93436	Nov-21	0.027612	1.12E+12	-0.126616	0	1.68E+12	0.667175	0.018422
2381626	93436	Dec-21	-0.076855	1.15E+12	-0.08491	2	2.19E+12	0.524546	-0.040314
2381627	93436	Jan-22	-0.113609	1.09E+12	-0.074947	3	3.11E+12	0.351332	-0.039915
2381628	93436	Feb-22	-0.070768	9.68E+11	-0.135317	0	1.56E+12	0.619297	-0.043827
2381629	93436	Mar-22	0.238009	9.00E+11	-0.074777	3	3.94E+12	0.228422	0.054367

c) Get FF3 factors and do linear regressions in CAPM and FF3 models.

	allo	ym	vw_ret	mktrf	smb	hml	rf	ri-rf
0	0	Jan-80	0.125937	0.0551	0.0162	0.0175	0.008	0.117937
1	1	Jan-80	0.084405	0.0551	0.0162	0.0175	0.008	0.076405
2	2	Jan-80	0.082535	0.0551	0.0162	0.0175	0.008	0.074535
3	3	Jan-80	0.097458	0.0551	0.0162	0.0175	0.008	0.089458
4	4	Jan-80	0.115045	0.0551	0.0162	0.0175	0.008	0.107045
...	...	...	...	...	...	...	...	...
4995	5	Aug-21	0.045068	0.0291	-0.0045	-0.0007	0	0.045068
4996	6	Aug-21	0.040469	0.0291	-0.0045	-0.0007	0	0.040469
4997	7	Aug-21	0.024415	0.0291	-0.0045	-0.0007	0	0.024415
4998	8	Aug-21	0.034322	0.0291	-0.0045	-0.0007	0	0.034322
4999	9	Aug-21	0.012547	0.0291	-0.0045	-0.0007	0	0.012547

d) Create maxret and plot the cumulative return time series.

Dep. Variable:		ri-rf	**R-squared:**		0.563
Model:		OLS	**Adj. R-squared:**		0.562
Method:		Least Squares	**F-statistic:**		641.7
Date:		Tue, 21 Jun 2022	**Prob (F-statistic):**		1.40e-91
Time:		10:27:21	**Log-Likelihood:**		704.52
No. Observations:		500	**AIC:**		-1405.
Df Residuals:		498	**BIC:**		-1397.
Df Model:		1			
Covariance Type:		nonrobust			
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0126	0.003	-4.687	0.000	-0.018	-0.007
mktrf	1.4985	0.059	25.332	0.000	1.382	1.615
Omnibus:		84.076	**Durbin-Watson:**		1.997
Prob(Omnibus):		0.000	**Jarque-Bera (JB):**		621.511
Skew:		0.478	**Prob(JB):**		1.10e-135
Kurtosis:		8.377	**Cond. No.**		22.3

Regression table for CAPM model (using Portfolio 0 as an example)

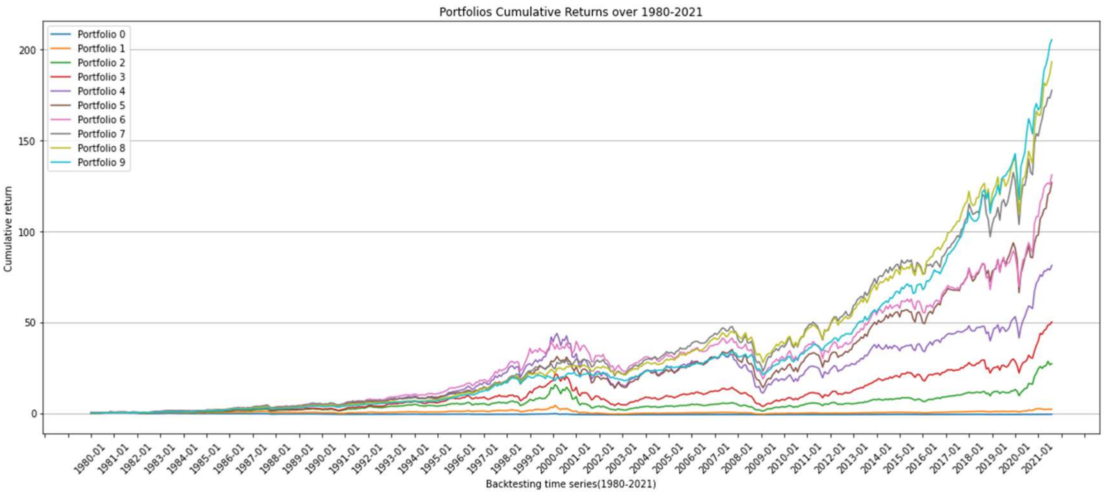
Dep. Variable:		ri-rf	**R-squared:**		0.697
Model:		OLS	**Adj. R-squared:**		0.695
Method:		Least Squares	**F-statistic:**		380.3
Date:		Tue, 21 Jun 2022	**Prob (F-statistic):**		3.76e-128
Time:		10:27:22	**Log-Likelihood:**		796.03
No. Observations:		500	**AIC:**		-1584.
Df Residuals:		496	**BIC:**		-1567.
Df Model:		3			
Covariance Type:		nonrobust			
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-0.0118	0.002	-5.242	0.000	-0.016	-0.007
mktrf	1.2990	0.052	25.215	0.000	1.198	1.400
smb	1.0602	0.077	13.735	0.000	0.909	1.212
hml	-0.2271	0.075	-3.021	0.003	-0.375	-0.079
Omnibus:		99.467	**Durbin-Watson:**		1.786
Prob(Omnibus):		0.000	**Jarque-Bera (JB):**		625.998
Skew:		0.689	**Prob(JB):**		1.16e-136
Kurtosis:		8.305	**Cond. No.**		37.4

Regression table for FF3 model (using Portfolio 0 as an example)

4. Findings

The factor can help investors gain well pay-off until the beginning of this century. Unfortunately, it is not suitable for all investors in nowadays market.

4.1. Portfolio performance



We backtracked the data from 1980 to 2021 for 41 years and got the cumulative return curve of the ten portfolios. We find an interesting phenomenon, the group with the lower maximum return in the last month will get the higher return.

4.2. Monotonic relation of alpha

In the CAPM and Fama-French models, the alpha for group 0 through group 9 are monotonic. Moreover, both models' alphas and betas are mostly statistically significant. This means our strategy has the potential to be profitable

CAPM	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
α	-0.0125	-0.0089	**-0.0046**	-0.003	-0.0018	**-0.0006**	-0.0001	0.001	**0.002**	0.0031
	(-4.640)	(-4.325)	**(-2.787)**	(-2.224)	(-1.878)	**(-0.705)**	(-0.127)	-1.671	**-2.905**	-2.905
β	1.4972	1.527	**1.4551**	1.3307	1.2451	**1.1513**	1.0608	0.9525	**0.8179**	0.6594
	-25.239	-33.829	**-40.462**	-44.768	-57.722	**-64.463**	-68.457	-66.576	**-53.382**	-53.382

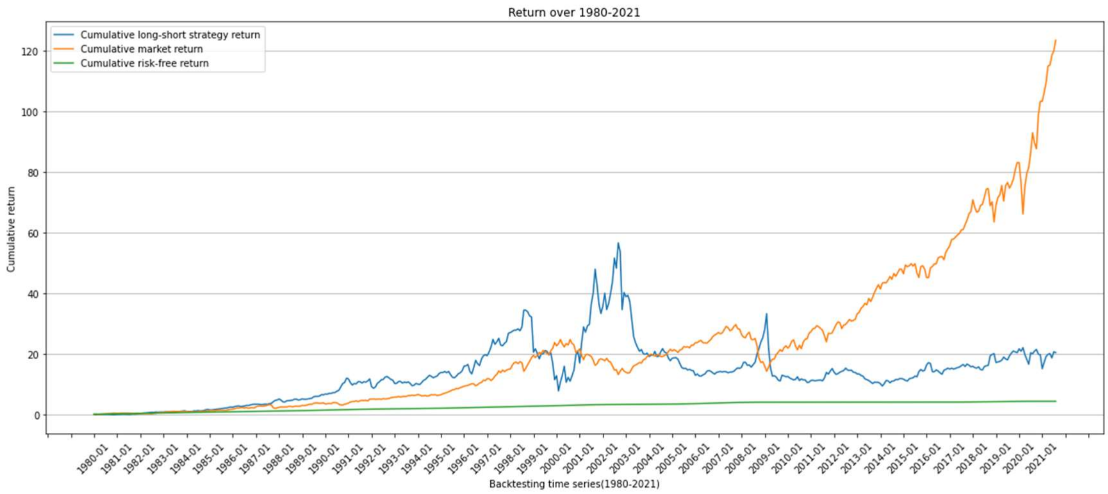
Alpha & beta for ten portfolios using CAPM

FF3	0	1	2	3	4	5	6	7	8	9
α	-0.0117	-0.0077	**-0.0037**	-0.0025	-0.0016	**-0.0007**	-0.0003	0.0007	**0.0015**	0.0026
	(-5.172)	(-4.792)	**(-2.837)**	(-2.215)	(-1.765)	**(-0.883)**	(-0.444)	-1.154	**-2.475**	-3.375
β	1.2983	1.3446	**1.3192**	1.234	1.1959	**1.1377**	1.0693	0.9847	**0.868**	0.7135
	-25.062	-36.44	**-43.415**	-46.205	-57.026	**-62.436**	-66.716	-70.162	**-62.256**	-40.154
S	1.0562	0.8312	**0.6405**	0.4674	0.253	**0.1337**	0.017	-0.0882	**-0.1619**	-0.1889
	-13.607	-15.034	**-14.068**	-11.682	-8.051	**-4.897**	-0.706	(-4.198)	**(-7.749)**	(-7.095)
H	-0.2276	-0.3595	**-0.2445**	-0.1607	-0.066	**0.0517**	0.0779	0.1276	**0.1716**	0.1698
	(-3.010)	(-6.675)	**(-5.513)**	(-4.213)	(-2.157)	**-1.944**	-3.329	-6.228	**-8.431**	-6.546

All coefficients for ten portfolios using FF3

4.3. Strategy performance

We plot the net change of one dollar invested in the ETF market index, the bond market, and our strategy since 1981. The graph is shown below.



Our strategy has a maximum return of 60 times if operating correctly. If investors put one dollar in our strategy in 1980, they would get nearly 60 dollars in 2002. However, the strategy failed to outperform the ETF for a period of time. It did not catch up with the American market after the subprime mortgage crisis.

4.4. Sharpe ratio and maximum drawdown

Sharpe ratio is used to evaluate one asset's return and risk. The Sharpe ratio of our strategy can be calculated as approximately 0.08. This number is not that high to prove the strategy is safe enough.

```
[32]: 0.006118065377444524
[33]: sharp.ls_ret.std()
[33]: 0.07958716614837075
[34]: sharp_ratio = (sharp.ls_ret.mean() - sharp.rf.mean())/sharp.ls_ret.std()
      sharp_ratio
[34]: 0.07687251190774769
```

A maximum drawdown is used to describe the worst situation after a specific date. Our strategy's highest return date is September 2002, and the lowest point after this is February 2014.

```
[37]: maxdraw
[37]: 47.24769132778143
[38]: maxrate
[38]: 0.8355405441668534
```

After calculation, the maximum drawdown is shown as about 0.836. It is an extremely high number compared to most of the private funds. If our investors put in 100 dollars in September 2002, the maximum potential loss they would suffer is 83.6 dollars. These two parameters prove that our strategy is unfriendly to normal individual investors.

5. Discussions

5.1. Hedging strategy analysis

From the monotonic alpha pattern for both models, we can see in the data period from Jan 1980 to Aug 2021, with monthly rebalanced value-weighted portfolios, there is statistically significant alpha for each portfolio 0 to 9. Since our maxret has a minus sign before the data, we should long the past month's lowest maxret stocks (portfolio 9) and short the highest maxret stocks (portfolio 0) to generate the profit.

The motivation behind the actions is that we can somehow hedge away the market risk in this way. Of course, we cannot guarantee ultimately diversifying the market risk. However, since we choose stocks by maxret in the whole market from the beta value shown before (can be described as APT equations below), if we do a minus operation between the equations, we can diversify a lot of market risk away. Therefore, our alpha mostly comes from the idiosyncratic risk (the difference between alpha).

$$r_0 - r_f = \alpha_0 + \beta_0 * \left(r_m - r_f\right) + \epsilon_0$$

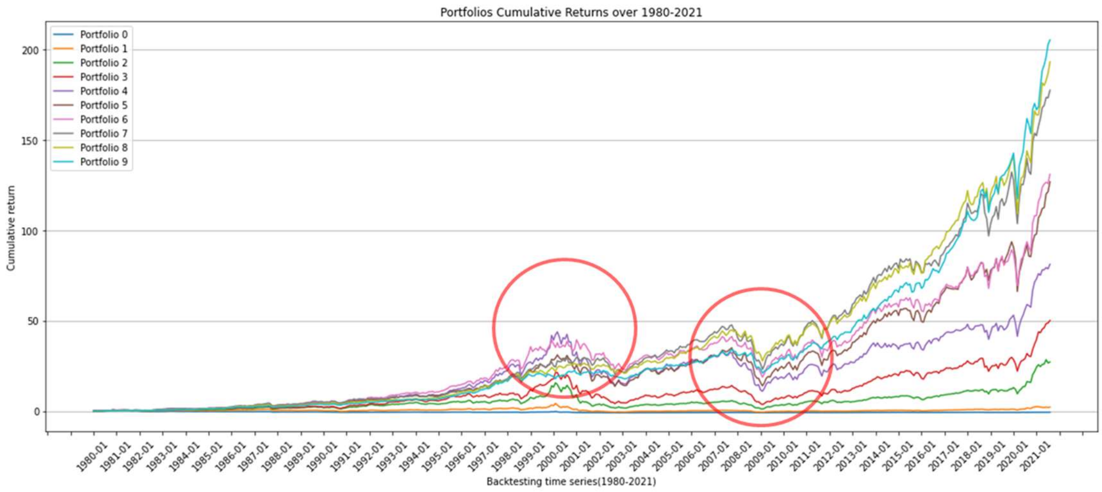
$$r_9 - r_f = \alpha_9 + \beta_9 * \left(r_m - r_f\right) + \epsilon_9$$

APT equations showing stock selection by maxret

We often use a long-short strategy to earn a huge amount of money when the market is volatile. Each section of the industry has different trends in the stock market at the same time. If we are smart enough to choose the time to buy the correct stocks, we can earn a profit from buying some stocks that are rising in some industries, and generate another profit when we sell stocks that are going to fall. However, when the market is expanding, we cannot take advantage of the market's good performance and suffer more idiosyncratic risks. Finding an appropriate chance for long or short stocks is difficult. Thus, we cannot have a considerable profit at those times.

5.2. Market volatility and portfolio returns

Suppose we solve the earning and losing periods of our strategy and map the periods into the cumulative return table. In that case, we can find out that the cumulative return of the ten portfolios intersects with each other. Why are there intersections, and why do the intersections affect the performance of our long-short portfolio?



Intersections of portfolios (circled part)

The volatility of the market causes intersections of 10 portfolios. The more volatile the market, the greater the market risk. In this period, there are opportunities to generate profits.

The portfolio we short is the one that has the most significant maximum return in the previous period. Following Bali and his partners' work, more investors tend to invest in these stocks, so the demand increases, then the price increases. Ideally, the stock is persistent and still performs well, and the return compared with the increasing price has less space for increasing. However, those stocks have more risk and volatility (Bali et al., 2011), so it's more likely to perform not that well in the waving market. Therefore, we short this kind of stock.

The portfolio we buy includes stocks with the lowest maximum return in the previous period, so not many investors trust these stocks. When the market is waving, it has a larger space to increase and generate huge profit, which will easily cover the loss in the shorted portfolio and gain us lots of profit.

6. Recommendation

Our strategy may be helpful for:

a) Wealthy individuals who are willing to take high risks for a high pay-off.

b) Institutions that can evaluate proper entries for a short-term pay-off.

References

Bali, T. G., Cakici, N., & Whitelaw, R. F. (2011). Maxing out: Stocks as lotteries and the cross-section of expected returns. *Journal of Financial Economics*, 99(2), 427-446. https://doi.org/10.1016/j.jfineco.2010.08.014.

Barberis, N., & Huang, M. (2008). Stocks as Lotteries: The Implications of Probability Weighting for Security Prices. *The American Economic Review*, 98(5), 2066–2100. http://www.jstor.org/stable/29730162.

Maximum_Return

June 24, 2022

1 Group Assignment 1: Construct the factor (Maxret)

[1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import matplotlib.pyplot as plt
from matplotlib.pyplot import MultipleLocator
```

1.1 Import the data

[2]:
```
df = pd.read_sas(r'F:\ \3380\project_1.sas7bdat')
df
```

[2]:
```
          PERMNO   PERMCO        DATE       RET          mcap  momentum12  \
0        10000.0   7952.0 1986-04-30 -0.098592  1.633000e+07         NaN
1        10000.0   7952.0 1986-05-30 -0.222656  1.517200e+07         NaN
2        10000.0   7952.0 1986-06-30 -0.005025  1.179386e+07         NaN
3        10000.0   7952.0 1986-07-31 -0.080808  1.173459e+07         NaN
4        10000.0   7952.0 1986-08-29 -0.615385  1.078634e+07         NaN
…            …        …          …         …             …           …
2381625  93436.0  53453.0 2021-11-30  0.027612  1.118751e+12    0.807603
2381626  93436.0  53453.0 2021-12-31 -0.076855  1.149642e+12    1.870838
2381627  93436.0  53453.0 2022-01-31 -0.113609  1.092218e+12    1.016843
2381628  93436.0  53453.0 2022-02-28 -0.070768  9.681319e+11    0.497556
2381629  93436.0  53453.0 2022-03-31  0.238009  8.996190e+11    0.180447

          Accrual     GProf     Oprof       Noa  …     FCF_P   CAPXLTG  \
0        0.051385 -0.032242 -0.434257 -0.284131  … -0.044091       NaN
1        0.051385 -0.032242 -0.434257 -0.284131  … -0.047456       NaN
2        0.051385 -0.032242 -0.434257 -0.284131  … -0.061049       NaN
3        0.067911  0.009177 -0.752524 -0.557969  … -0.105841       NaN
4        0.067911  0.009177 -0.752524 -0.557969  … -0.115146       NaN
…             …         …         …         …  …         …         …
2381625  0.143450  0.227853  0.151546 -0.404512  …  0.003686 -2.403436
2381626  0.122502  0.247703  0.174489 -0.407116  …  0.003538 -3.386263
2381627  0.122502  0.247703  0.174489 -0.407116  …  0.003724 -3.386263
```

```
2381628  0.122502  0.247703  0.174489 -0.407116  …  0.004201 -3.386263
2381629  0.104621  0.265842  0.193140 -0.429623  …  0.005539 -3.548883

             CEQG  FIRMTANG      ivol    maxret  seasonality       CEI  \
0        3.008746  0.737513 -0.047159 -0.145161          NaN       NaN
1        3.008746  0.737513 -0.032645 -0.022727          NaN       NaN
2        3.008746  0.737513 -0.034740 -0.115702          NaN       NaN
3        0.161103  0.546046 -0.026370 -0.042553          NaN       NaN
4        0.161103  0.546046 -0.037241 -0.116667          NaN       NaN
…             …         …         …         …           …         …
2381625 -0.077638  0.637375 -0.018568 -0.126616     0.087781 -0.292665
2381626 -0.090671  0.628411 -0.028291 -0.084910     0.119409 -0.292178
2381627 -0.090671  0.628411 -0.028752 -0.074947     0.183826 -0.246419
2381628 -0.090671  0.628411 -0.036362 -0.135317    -0.023906 -0.245744
2381629 -0.358335  0.632665 -0.030830 -0.074777    -0.092575 -0.245744

              nsi  prcdelay1
0             NaN        NaN
1             NaN        NaN
2             NaN        NaN
3             NaN        NaN
4             NaN        NaN
…             …         …
2381625 -0.060591   0.180243
2381626 -0.072393   0.159999
2381627 -0.057393   0.162654
2381628 -0.057761   0.231407
2381629 -0.057761   0.306750

[2381630 rows x 25 columns]
```

[3]:
```python
ff3 = pd.read_sas(r'F:\ \3380\factors_monthly.sas7bdat')
ff3
```

[3]:
```
           date  mktrf     smb     hml      rf    year  month  umd     dateff
0    1926-07-01  0.0296 -0.0238 -0.0273  0.0022  1926.0    7.0  NaN 1926-07-31
1    1926-08-01  0.0264 -0.0147  0.0414  0.0025  1926.0    8.0  NaN 1926-08-31
2    1926-09-01  0.0036 -0.0139  0.0012  0.0023  1926.0    9.0  NaN 1926-09-30
3    1926-10-01 -0.0324 -0.0013  0.0065  0.0032  1926.0   10.0  NaN 1926-10-30
4    1926-11-01  0.0253 -0.0016 -0.0038  0.0031  1926.0   11.0  NaN 1926-11-30
…           …      …       …       …       …       …      …    …        …
1137 2021-04-01  0.0493 -0.0311 -0.0093  0.0000  2021.0    4.0  NaN 2021-04-30
1138 2021-05-01  0.0029 -0.0028  0.0704  0.0000  2021.0    5.0  NaN 2021-05-28
1139 2021-06-01  0.0279  0.0180 -0.0776  0.0000  2021.0    6.0  NaN 2021-06-30
1140 2021-07-01  0.0119 -0.0396 -0.0175  0.0000  2021.0    7.0  NaN 2021-07-30
1141 2021-08-01  0.0291 -0.0045 -0.0007  0.0000  2021.0    8.0  NaN 2021-08-31
```

[1142 rows x 9 columns]

1.2 Data cleaning

[4]:
```
mr = df[['PERMNO','DATE','RET','mcap','maxret']] # Select the needed columns
mr['ym'] = pd.to_datetime(mr.DATE).dt.to_period('M') # Transfer the data type
mr
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_896\2140906520.py:2:
SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  mr['ym'] = pd.to_datetime(mr.DATE).dt.to_period('M') # Transfer the data type
```

[4]:

	PERMNO	DATE	RET	mcap	maxret	ym
0	10000.0	1986-04-30	-0.098592	1.633000e+07	-0.145161	1986-04
1	10000.0	1986-05-30	-0.222656	1.517200e+07	-0.022727	1986-05
2	10000.0	1986-06-30	-0.005025	1.179386e+07	-0.115702	1986-06
3	10000.0	1986-07-31	-0.080808	1.173459e+07	-0.042553	1986-07
4	10000.0	1986-08-29	-0.615385	1.078634e+07	-0.116667	1986-08
…	…	…	…	…	…	…
2381625	93436.0	2021-11-30	0.027612	1.118751e+12	-0.126616	2021-11
2381626	93436.0	2021-12-31	-0.076855	1.149642e+12	-0.084910	2021-12
2381627	93436.0	2022-01-31	-0.113609	1.092218e+12	-0.074947	2022-01
2381628	93436.0	2022-02-28	-0.070768	9.681319e+11	-0.135317	2022-02
2381629	93436.0	2022-03-31	0.238009	8.996190e+11	-0.074777	2022-03

[2381630 rows x 6 columns]

[5]:
```
mr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381630 entries, 0 to 2381629
Data columns (total 6 columns):
 #   Column  Dtype
---  ------  -----
 0   PERMNO  float64
 1   DATE    datetime64[ns]
 2   RET     float64
 3   mcap    float64
 4   maxret  float64
 5   ym      period[M]
dtypes: datetime64[ns](1), float64(4), period[M](1)
memory usage: 109.0 MB
```

[6]:
```
mr = mr.sort_values(['PERMNO','ym']).reset_index(drop = True) # Sort the␣
↪dataframe by PERMNO and year-month
mr = mr[['PERMNO','ym','RET','mcap','maxret']]
mr = mr.loc[mr.maxret.isnull() == False] # Drop the columns with no maximum␣
↪return value
mr['RET'] = mr.RET.fillna(0) # Fill the empty return value with 0
mr
```

[6]:
```
          PERMNO       ym       RET          mcap    maxret
0        10000.0  1986-04 -0.098592  1.633000e+07 -0.145161
1        10000.0  1986-05 -0.222656  1.517200e+07 -0.022727
2        10000.0  1986-06 -0.005025  1.179386e+07 -0.115702
3        10000.0  1986-07 -0.080808  1.173459e+07 -0.042553
4        10000.0  1986-08 -0.615385  1.078634e+07 -0.116667
…            …        …        …             …         …
2381625  93436.0  2021-11  0.027612  1.118751e+12 -0.126616
2381626  93436.0  2021-12 -0.076855  1.149642e+12 -0.084910
2381627  93436.0  2022-01 -0.113609  1.092218e+12 -0.074947
2381628  93436.0  2022-02 -0.070768  9.681319e+11 -0.135317
2381629  93436.0  2022-03  0.238009  8.996190e+11 -0.074777

[2381503 rows x 5 columns]
```

1.3 Allocation into 10 groups by Past 1 month Maximum Return

[7]:
```
perc = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9] # Set the percentiles
mr['allo'] = 1 # Create a new column to store the portfolio index
```

[8]:
```
def allocation(df): # Doing allocation by sorting maximum returns(Actually␣
↪sorted by the negative of the maximum returns)
    quan = df['maxret'].describe(percentiles = perc)
    quan = quan.iloc[4:13]
    df.loc[df.maxret < quan[0],'allo'] = 0
    df.loc[(df.maxret >= quan[0])&(df.maxret < quan[1]),'allo'] = 1
    df.loc[(df.maxret >= quan[1])&(df.maxret < quan[2]),'allo'] = 2
    df.loc[(df.maxret >= quan[2])&(df.maxret < quan[3]),'allo'] = 3
    df.loc[(df.maxret >= quan[3])&(df.maxret < quan[4]),'allo'] = 4
    df.loc[(df.maxret >= quan[4])&(df.maxret < quan[5]),'allo'] = 5
    df.loc[(df.maxret >= quan[5])&(df.maxret < quan[6]),'allo'] = 6
    df.loc[(df.maxret >= quan[6])&(df.maxret < quan[7]),'allo'] = 7
    df.loc[(df.maxret >= quan[7])&(df.maxret < quan[8]),'allo'] = 8
    df.loc[df.maxret >= quan[8],'allo'] = 9
    return df

mr = mr.groupby(['ym']).apply(allocation) # Groupby function applied to each␣
↪stock at each year-month
```

```
mr
```

[8]:
```
         PERMNO       ym       RET          mcap    maxret  allo
0        10000.0  1986-04 -0.098592  1.633000e+07 -0.145161     0
1        10000.0  1986-05 -0.222656  1.517200e+07 -0.022727     8
2        10000.0  1986-06 -0.005025  1.179386e+07 -0.115702     1
3        10000.0  1986-07 -0.080808  1.173459e+07 -0.042553     5
4        10000.0  1986-08 -0.615385  1.078634e+07 -0.116667     1
…          …        …         …             …         …     …
2381625  93436.0  2021-11  0.027612  1.118751e+12 -0.126616     0
2381626  93436.0  2021-12 -0.076855  1.149642e+12 -0.084910     2
2381627  93436.0  2022-01 -0.113609  1.092218e+12 -0.074947     3
2381628  93436.0  2022-02 -0.070768  9.681319e+11 -0.135317     0
2381629  93436.0  2022-03  0.238009  8.996190e+11 -0.074777     3

[2381503 rows x 6 columns]
```

1.4 Get the value-weighted returns

[9]:
```
mr['totalmcap'] = mr.groupby(['allo','ym']).mcap.transform('sum') # To get the␣
↪total marketcap in the same portfolio at each year-month
mr['weight'] = mr.mcap / mr.totalmcap # Get the weight for each stock in each␣
↪portfolio at each period
mr['vw_ret'] = mr.weight * mr.RET # Get the value-weighted return for each␣
↪stock in the portfolio
mr
```

[9]:
```
         PERMNO       ym       RET          mcap    maxret  allo  \
0        10000.0  1986-04 -0.098592  1.633000e+07 -0.145161     0
1        10000.0  1986-05 -0.222656  1.517200e+07 -0.022727     8
2        10000.0  1986-06 -0.005025  1.179386e+07 -0.115702     1
3        10000.0  1986-07 -0.080808  1.173459e+07 -0.042553     5
4        10000.0  1986-08 -0.615385  1.078634e+07 -0.116667     1
…          …        …         …             …         …     …
2381625  93436.0  2021-11  0.027612  1.118751e+12 -0.126616     0
2381626  93436.0  2021-12 -0.076855  1.149642e+12 -0.084910     2
2381627  93436.0  2022-01 -0.113609  1.092218e+12 -0.074947     3
2381628  93436.0  2022-02 -0.070768  9.681319e+11 -0.135317     0
2381629  93436.0  2022-03  0.238009  8.996190e+11 -0.074777     3

            totalmcap    weight    vw_ret
0        1.429314e+10  0.001143 -0.000113
1        4.198753e+11  0.000036 -0.000008
2        3.642032e+10  0.000324 -0.000002
3        2.660435e+11  0.000044 -0.000004
4        4.063298e+10  0.000265 -0.000163
```

```
…                …          …          …
2381625  1.676849e+12  0.667175  0.018422
2381626  2.191692e+12  0.524546 -0.040314
2381627  3.108792e+12  0.351332 -0.039915
2381628  1.563276e+12  0.619297 -0.043827
2381629  3.938403e+12  0.228422  0.054367

[2381503 rows x 9 columns]
```

1.5 Get the number of observations in each period

[10]:
```
num_pf = mr.groupby(['allo','ym']).agg({'PERMNO':'count'}).reset_index()
num_pf
```

[10]:
```
      allo       ym  PERMNO
0        0  1980-01     374
1        0  1980-02     376
2        0  1980-03     367
3        0  1980-04     383
4        0  1980-05     381
…     …        …       …
5065     9  2021-11     373
5066     9  2021-12     383
5067     9  2022-01     383
5068     9  2022-02     382
5069     9  2022-03     385

[5070 rows x 3 columns]
```

[11]:
```
# num_pf.to_csv(r'F:\ \3380\Portofolio stocks number.csv') # Customize the path␣
↪when other users run the code
```

[12]:
```
vw_pf = mr.groupby(['allo','ym']).agg({'vw_ret':'sum'}).reset_index() # Sum the␣
↪stock return into the portfolio at each month
vw_pf = vw_pf.sort_values(['ym','allo']).reset_index(drop = True) # Sort the␣
↪dataframe
vw_pf
```

[12]:
```
      allo       ym    vw_ret
0        0  1980-01  0.125937
1        1  1980-01  0.084405
2        2  1980-01  0.082535
3        3  1980-01  0.097458
4        4  1980-01  0.115045
…     …        …        …
5065     5  2022-03  0.026769
```

```
5066      6  2022-03  0.024341
5067      7  2022-03  0.041449
5068      8  2022-03  0.011741
5069      9  2022-03  0.033091

[5070 rows x 3 columns]
```

1.6 Get the ff3 factors at the same time period

[13]:
```python
ff3['ym'] = pd.to_datetime(ff3.date).dt.to_period('M') # Transfer the data type
ff3 = ff3[['ym','mktrf','smb','hml','rf']] # Select the columns
ff3 = ff3[ff3.ym >= '1980-01'].reset_index(drop = True) # Set the time period␣
↪of the factors data
ff3
```

[13]:
```
          ym   mktrf     smb     hml      rf
0    1980-01  0.0551  0.0162  0.0175  0.0080
1    1980-02 -0.0122 -0.0185  0.0061  0.0089
2    1980-03 -0.1290 -0.0664 -0.0101  0.0121
3    1980-04  0.0397  0.0105  0.0108  0.0126
4    1980-05  0.0526  0.0213  0.0038  0.0081
..       …       …       …       …       …
495  2021-04  0.0493 -0.0311 -0.0093  0.0000
496  2021-05  0.0029 -0.0028  0.0704  0.0000
497  2021-06  0.0279  0.0180 -0.0776  0.0000
498  2021-07  0.0119 -0.0396 -0.0175  0.0000
499  2021-08  0.0291 -0.0045 -0.0007  0.0000

[500 rows x 5 columns]
```

1.7 Merge the table

[14]:
```python
vw_pf = pd.merge(vw_pf, ff3, how = 'left', on = 'ym') # Merge the table
vw_pf['ri-rf'] = vw_pf['vw_ret'] - vw_pf['rf'] # Get the risk premium of the␣
↪portfolio
vw_pf = vw_pf.dropna()
vw_pf
```

[14]:
```
      allo       ym    vw_ret   mktrf     smb     hml     rf     ri-rf
0        0  1980-01  0.125937  0.0551  0.0162  0.0175  0.008  0.117937
1        1  1980-01  0.084405  0.0551  0.0162  0.0175  0.008  0.076405
2        2  1980-01  0.082535  0.0551  0.0162  0.0175  0.008  0.074535
3        3  1980-01  0.097458  0.0551  0.0162  0.0175  0.008  0.089458
4        4  1980-01  0.115045  0.0551  0.0162  0.0175  0.008  0.107045
…     …        …        …       …       …       …      …         …
4995     5  2021-08  0.045068  0.0291 -0.0045 -0.0007  0.000  0.045068
```

```
4996      6  2021-08  0.040469  0.0291 -0.0045 -0.0007  0.000  0.040469
4997      7  2021-08  0.024415  0.0291 -0.0045 -0.0007  0.000  0.024415
4998      8  2021-08  0.034322  0.0291 -0.0045 -0.0007  0.000  0.034322
4999      9  2021-08  0.012547  0.0291 -0.0045 -0.0007  0.000  0.012547

[5000 rows x 8 columns]
```

1.8 Use sklearn to do linear regression

[15]:
```
def capm_alpha(df): # Get CAPM alpha
    y = df[['ri-rf']]
    X = df[['mktrf']]
    return np.squeeze(LinearRegression().fit(X,y).intercept_)

capm_a = vw_pf.groupby('allo').apply(capm_alpha).reset_index() # Apply the␣
 ↪linear regression function onto each group
capm_a = capm_a.rename(columns = {0:'alpha'}) # Rename the column
capm_a
```

[15]:
```
   allo                  alpha
0     0   -0.012495042292301237
1     1   -0.008863696728677605
2     2   -0.004550742756413477
3     3  -0.0030008100510776356
4     4  -0.0018387787222079056
5     5  -0.0005716188134089282
6     6  -8.961470574735727e-05
7     7   0.0010852679491717273
8     8   0.0020209725508139936
9     9   0.0031317738167843156
```

[16]:
```
def ff3_alpha(df): # Get Fama-French 3 Factors alpha
    y = df[['ri-rf']]
    X = df[['mktrf','smb','hml']]
    return np.squeeze(LinearRegression().fit(X,y).intercept_)

ff3_a = vw_pf.groupby('allo').apply(ff3_alpha).reset_index()
ff3_a = ff3_a.rename(columns = {0:'alpha'})
ff3_a
```

[16]:
```
   allo                  alpha
0     0   -0.011706225499884884
1     1   -0.007725406095218336
2     2  -0.0037669032689515106
3     3   -0.002479527662928635
4     4  -0.0016168069822782676
5     5  -0.0007027576528644639
```

```
6     6  -0.0003108391933360549
7     7   0.0007078950031855569
8     8   0.0015079043851672401
9     9    0.002620428352202414
```

1.9 Use statsmodels to do linear regression

[17]:
```
Y = vw_pf.loc[vw_pf.allo == 0,'ri-rf'] # Get the more statistical data of the␣
 ↪regression of each group
X = vw_pf.loc[vw_pf.allo == 0,'mktrf'] # Can adjust from portfolio 0 to␣
 ↪portfolio 9
X = sm.add_constant(X)
model1 = sm.OLS(Y,X)
capm_res = model1.fit()
capm_res.summary() # Get the CAPM regression results
```

[17]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  ri-rf   R-squared:                       0.561
Model:                            OLS   Adj. R-squared:                  0.560
Method:                 Least Squares   F-statistic:                     637.0
Date:                Thu, 23 Jun 2022   Prob (F-statistic):           3.94e-91
Time:                        10:27:22   Log-Likelihood:                 703.11
No. Observations:                 500   AIC:                            -1402.
Df Residuals:                     498   BIC:                            -1394.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0125      0.003     -4.640      0.000      -0.018      -0.007
mktrf          1.4972      0.059     25.239      0.000       1.381       1.614
==============================================================================
Omnibus:                       85.545   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              637.631
Skew:                           0.491   Prob(JB):                    3.47e-139
Kurtosis:                       8.444   Cond. No.                         22.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[18]:
```
Y = vw_pf.loc[vw_pf.allo == 0,'ri-rf']
X = vw_pf.loc[vw_pf.allo == 0,['mktrf','smb','hml']]
X = sm.add_constant(X)
model2 = sm.OLS(Y,X)
ff3_res = model2.fit()
ff3_res.summary() # Get FF3 regression results
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  ri-rf   R-squared:                       0.694
Model:                            OLS   Adj. R-squared:                  0.692
Method:                 Least Squares   F-statistic:                     375.1
Date:                Thu, 23 Jun 2022   Prob (F-statistic):          4.04e-127
Time:                        10:27:23   Log-Likelihood:                 793.27
No. Observations:                 500   AIC:                            -1579.
Df Residuals:                     496   BIC:                            -1562.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0117      0.002     -5.172      0.000      -0.016      -0.007
mktrf          1.2983      0.052     25.062      0.000       1.197       1.400
smb            1.0562      0.078     13.607      0.000       0.904       1.209
hml           -0.2276      0.076     -3.010      0.003      -0.376      -0.079
==============================================================================
Omnibus:                      102.488   Durbin-Watson:                   1.785
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              654.711
Skew:                           0.713   Prob(JB):                    6.78e-143
Kurtosis:                       8.422   Cond. No.                         37.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

1.10 Create the Maxret factor (long-short portfolio)

1.10.1 Basic Strategy is to long No.9 portfolio and short No.0 portfolio at each period

[19]:
```
ls_pf = vw_pf[(vw_pf.allo == 0)|(vw_pf.allo == 9)].reset_index(drop = True)
ls_pf
```

[19]:
```
     allo       ym    vw_ret   mktrf     smb     hml      rf     ri-rf
0       0  1980-01  0.125937  0.0551  0.0162  0.0175  0.0080  0.117937
1       9  1980-01  0.025887  0.0551  0.0162  0.0175  0.0080  0.017887
2       0  1980-02 -0.011719 -0.0122 -0.0185  0.0061  0.0089 -0.020619
3       9  1980-02 -0.034281 -0.0122 -0.0185  0.0061  0.0089 -0.043181
4       0  1980-03 -0.181539 -0.1290 -0.0664 -0.0101  0.0121 -0.193639
..    …        …       …       …       …       …       …        …
995     9  2021-06  0.023031  0.0279  0.0180 -0.0776  0.0000  0.023031
996     0  2021-07 -0.068221  0.0119 -0.0396 -0.0175  0.0000 -0.068221
997     9  2021-07  0.034776  0.0119 -0.0396 -0.0175  0.0000  0.034776
998     0  2021-08  0.024619  0.0291 -0.0045 -0.0007  0.0000  0.024619
999     9  2021-08  0.012547  0.0291 -0.0045 -0.0007  0.0000  0.012547

[1000 rows x 8 columns]
```

[20]:
```python
ls_pf['sft_ret'] = ls_pf.vw_ret.shift(1) # Shift the return and get the␣
 ↪long-short portfolio return in each month
ls_pf['ls_ret'] = ls_pf.vw_ret - ls_pf.sft_ret
ls_pf
```

[20]:
```
     allo       ym    vw_ret   mktrf     smb     hml      rf     ri-rf  \
0       0  1980-01  0.125937  0.0551  0.0162  0.0175  0.0080  0.117937
1       9  1980-01  0.025887  0.0551  0.0162  0.0175  0.0080  0.017887
2       0  1980-02 -0.011719 -0.0122 -0.0185  0.0061  0.0089 -0.020619
3       9  1980-02 -0.034281 -0.0122 -0.0185  0.0061  0.0089 -0.043181
4       0  1980-03 -0.181539 -0.1290 -0.0664 -0.0101  0.0121 -0.193639
..    …        …       …       …       …       …       …        …
995     9  2021-06  0.023031  0.0279  0.0180 -0.0776  0.0000  0.023031
996     0  2021-07 -0.068221  0.0119 -0.0396 -0.0175  0.0000 -0.068221
997     9  2021-07  0.034776  0.0119 -0.0396 -0.0175  0.0000  0.034776
998     0  2021-08  0.024619  0.0291 -0.0045 -0.0007  0.0000  0.024619
999     9  2021-08  0.012547  0.0291 -0.0045 -0.0007  0.0000  0.012547

      sft_ret    ls_ret
0         NaN       NaN
1    0.125937 -0.100051
2    0.025887 -0.037606
3   -0.011719 -0.022561
4   -0.034281 -0.147259
..        …         …
995  0.090957 -0.067926
996  0.023031 -0.091252
997 -0.068221  0.102997
998  0.034776 -0.010157
999  0.024619 -0.012072

[1000 rows x 10 columns]
```

[21]:
```
ls_pf = ls_pf[ls_pf.allo == 9]
ls_pf = ls_pf[['ym','ls_ret','mktrf','rf']].reset_index(drop = True)
ls_pf
```

[21]:
```
          ym    ls_ret   mktrf      rf
0    1980-01 -0.100051  0.0551  0.0080
1    1980-02 -0.022561 -0.0122  0.0089
2    1980-03  0.114231 -0.1290  0.0121
3    1980-04 -0.025424  0.0397  0.0126
4    1980-05  0.001162  0.0526  0.0081
..       …         …       …       …
495  2021-04  0.035703  0.0493  0.0000
496  2021-05  0.011366  0.0029  0.0000
497  2021-06 -0.067926  0.0279  0.0000
498  2021-07  0.102997  0.0119  0.0000
499  2021-08 -0.012072  0.0291  0.0000

[500 rows x 4 columns]
```

[22]:
```
vis = ls_pf # Get the gross return of each component
vis['grs_ls_ret'] = vis.ls_ret + 1
vis['grs_mkt'] = vis.mktrf + vis.rf + 1
vis['grs_rf'] = vis.rf + 1
vis
```

[22]:
```
          ym    ls_ret   mktrf      rf  grs_ls_ret  grs_mkt  grs_rf
0    1980-01 -0.100051  0.0551  0.0080    0.899949   1.0631  1.0080
1    1980-02 -0.022561 -0.0122  0.0089    0.977439   0.9967  1.0089
2    1980-03  0.114231 -0.1290  0.0121    1.114231   0.8831  1.0121
3    1980-04 -0.025424  0.0397  0.0126    0.974576   1.0523  1.0126
4    1980-05  0.001162  0.0526  0.0081    1.001162   1.0607  1.0081
..       …         …       …       …           …        …       …
495  2021-04  0.035703  0.0493  0.0000    1.035703   1.0493  1.0000
496  2021-05  0.011366  0.0029  0.0000    1.011366   1.0029  1.0000
497  2021-06 -0.067926  0.0279  0.0000    0.932074   1.0279  1.0000
498  2021-07  0.102997  0.0119  0.0000    1.102997   1.0119  1.0000
499  2021-08 -0.012072  0.0291  0.0000    0.987928   1.0291  1.0000

[500 rows x 7 columns]
```

[23]:
```
vis['cum_ls_ret'] = vis['grs_ls_ret'].cumprod()-1 # Get the cumulative return␣
↪of each component
vis['cum_mkt'] = vis['grs_mkt'].cumprod()-1
vis['cum_rf'] = vis['grs_rf'].cumprod()-1
vis
```

[23]:
```
          ym    ls_ret  mktrf      rf  grs_ls_ret  grs_mkt  grs_rf  \
0    1980-01 -0.100051  0.0551  0.0080    0.899949   1.0631  1.0080
1    1980-02 -0.022561 -0.0122  0.0089    0.977439   0.9967  1.0089
2    1980-03  0.114231 -0.1290  0.0121    1.114231   0.8831  1.0121
3    1980-04 -0.025424  0.0397  0.0126    0.974576   1.0523  1.0126
4    1980-05  0.001162  0.0526  0.0081    1.001162   1.0607  1.0081
..       …         …       …       …          …        …       …
495  2021-04  0.035703  0.0493  0.0000    1.035703   1.0493  1.0000
496  2021-05  0.011366  0.0029  0.0000    1.011366   1.0029  1.0000
497  2021-06 -0.067926  0.0279  0.0000    0.932074   1.0279  1.0000
498  2021-07  0.102997  0.0119  0.0000    1.102997   1.0119  1.0000
499  2021-08 -0.012072  0.0291  0.0000    0.987928   1.0291  1.0000

     cum_ls_ret     cum_mkt    cum_rf
0     -0.100051    0.063100  0.008000
1     -0.120355    0.059592  0.016971
2     -0.019872   -0.064275  0.029277
3     -0.044791   -0.015336  0.042245
4     -0.043681    0.044433  0.050688
..          …           …         …
495   19.764335  114.865760  4.257767
496   20.000346  115.201770  4.257767
497   18.573868  118.443800  4.257767
498   20.589915  119.865181  4.257767
499   20.329281  123.382358  4.257767

[500 rows x 10 columns]
```

[24]:
```python
vis = vis[['ym','cum_ls_ret','cum_mkt','cum_rf']] # Reorganize the data
vis.loc[vis.ym == '1980-01',['cum_ls_ret','cum_mkt','cum_rf']] = 0
vis['ym'] = vis.ym.astype('str')
vis
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_896\879734901.py:3:
SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  vis['ym'] = vis.ym.astype('str')
```

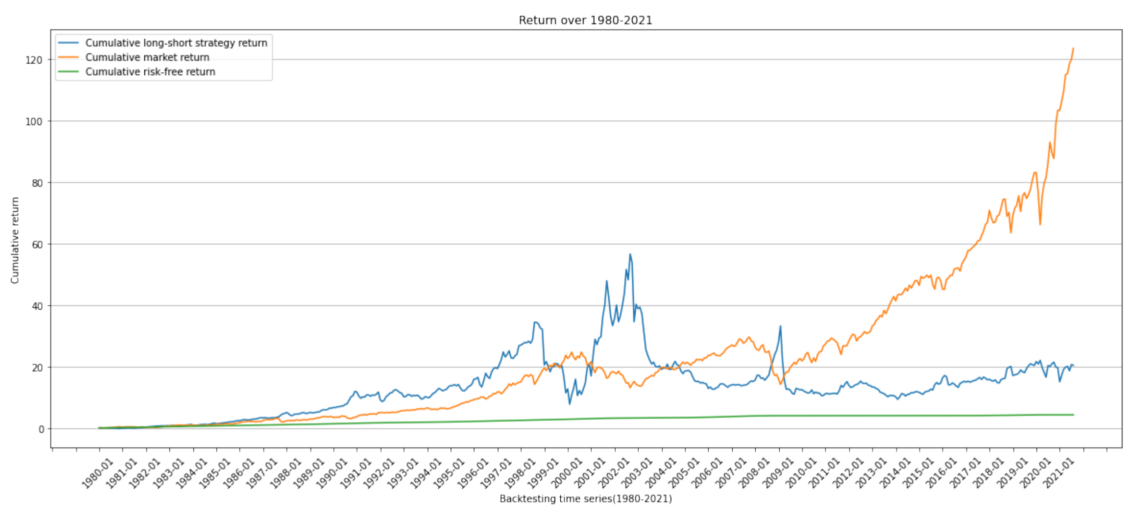
[24]:
```
          ym  cum_ls_ret    cum_mkt    cum_rf
0    1980-01    0.000000   0.000000  0.000000
1    1980-02   -0.120355   0.059592  0.016971
2    1980-03   -0.019872  -0.064275  0.029277
3    1980-04   -0.044791  -0.015336  0.042245
```

```
4    1980-05   -0.043681    0.044433  0.050688
..       …          …          …         …
495  2021-04   19.764335  114.865760  4.257767
496  2021-05   20.000346  115.201770  4.257767
497  2021-06   18.573868  118.443800  4.257767
498  2021-07   20.589915  119.865181  4.257767
499  2021-08   20.329281  123.382358  4.257767

[500 rows x 4 columns]
```

1.11 Plot the cumulative return time series

[25]:
```
plt.figure(figsize = [20,8])
plt.plot(vis.ym, vis.cum_ls_ret, label = 'Cumulative long-short strategy␣
 ↪return')
plt.plot(vis.ym, vis.cum_mkt, label = 'Cumulative market return')
plt.plot(vis.ym, vis.cum_rf, label = 'Cumulative risk-free return')
plt.xlabel('Backtesting time series(1980-2021)')
plt.ylabel('Cumulative return')
plt.xticks(rotation = 45)
plt.title('Return over 1980-2021')
x_major_locator=MultipleLocator(12)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.grid(axis = 'y')
plt.legend()
plt.show()
```



1.12 Plot the Portfolio cumulative return time series

[26]:
```
vw_pf
```

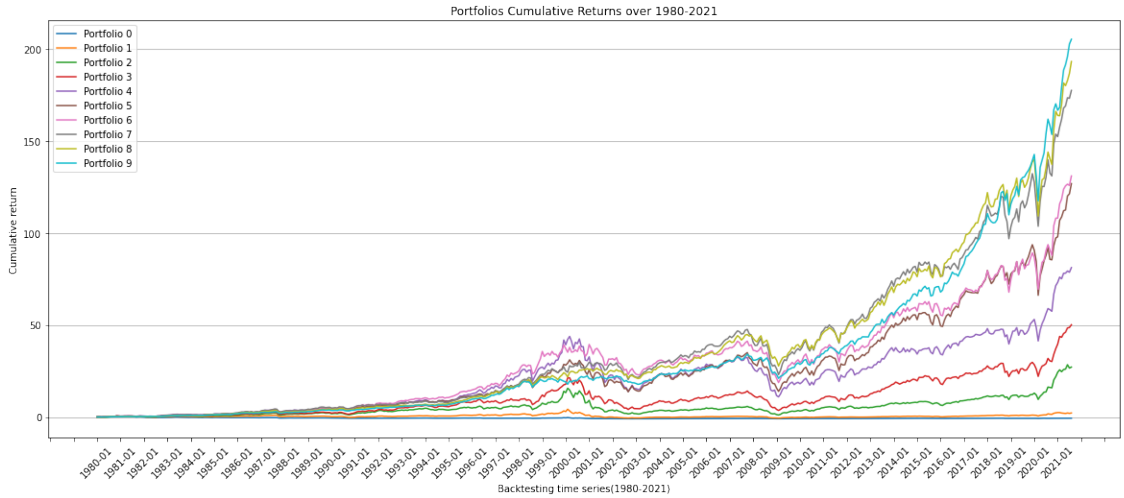
[26]:
```
      allo       ym    vw_ret  mktrf     smb     hml     rf     ri-rf
0        0  1980-01  0.125937  0.0551  0.0162  0.0175  0.008  0.117937
1        1  1980-01  0.084405  0.0551  0.0162  0.0175  0.008  0.076405
2        2  1980-01  0.082535  0.0551  0.0162  0.0175  0.008  0.074535
3        3  1980-01  0.097458  0.0551  0.0162  0.0175  0.008  0.089458
4        4  1980-01  0.115045  0.0551  0.0162  0.0175  0.008  0.107045
…     …       …        …        …       …       …       …       …
4995     5  2021-08  0.045068  0.0291 -0.0045 -0.0007  0.000  0.045068
4996     6  2021-08  0.040469  0.0291 -0.0045 -0.0007  0.000  0.040469
4997     7  2021-08  0.024415  0.0291 -0.0045 -0.0007  0.000  0.024415
4998     8  2021-08  0.034322  0.0291 -0.0045 -0.0007  0.000  0.034322
4999     9  2021-08  0.012547  0.0291 -0.0045 -0.0007  0.000  0.012547

[5000 rows x 8 columns]
```

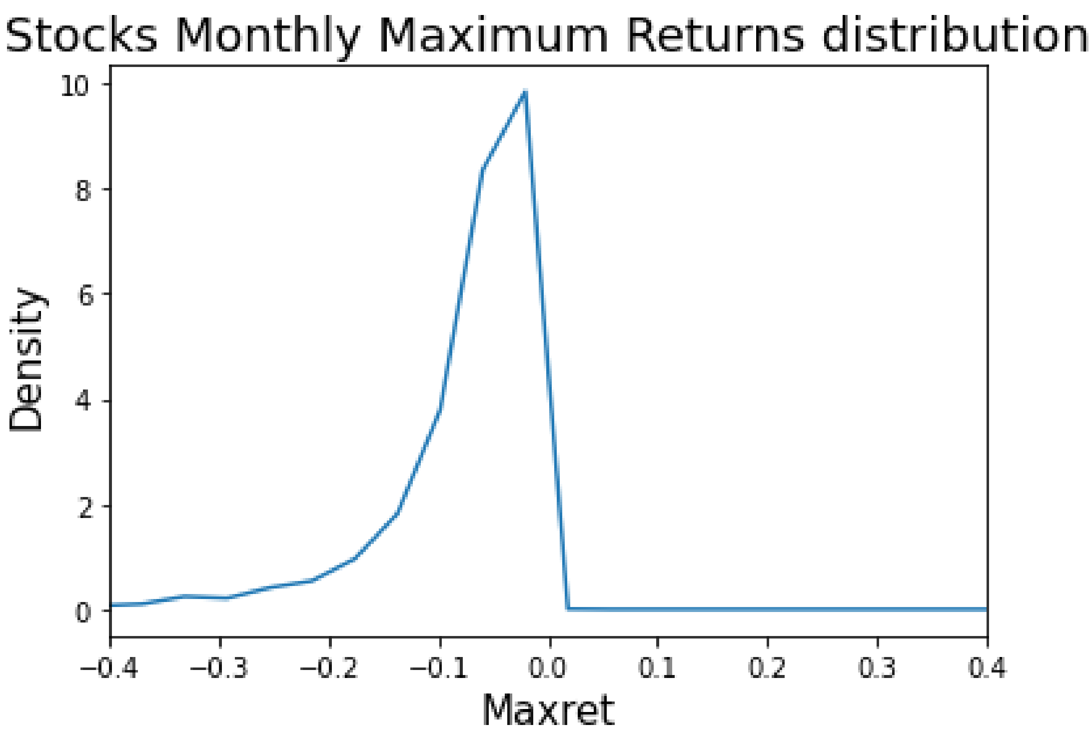
[27]:
```
# Get the gross return and cumulative return of each Portfolio
def Pf_allo_ret(df,a):
    Pf = df[df.allo == a]
    Pf['grs_vw_ret'] = Pf['vw_ret'] + 1
    Pf['cum_vw_ret'] = Pf['grs_vw_ret'].cumprod()-1
    Pf.loc[Pf.ym == '1980-01',['cum_vw_ret']] = 0
    return Pf.cum_vw_ret
```

[28]:
```
plt.figure(figsize = [20,8])
plt.plot(vis.ym, Pf_allo_ret(vw_pf,0), label = 'Portfolio 0')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,1), label = 'Portfolio 1')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,2), label = 'Portfolio 2')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,3), label = 'Portfolio 3')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,4), label = 'Portfolio 4')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,5), label = 'Portfolio 5')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,6), label = 'Portfolio 6')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,7), label = 'Portfolio 7')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,8), label = 'Portfolio 8')
plt.plot(vis.ym, Pf_allo_ret(vw_pf,9), label = 'Portfolio 9')
plt.xlabel('Backtesting time series(1980-2021)')
plt.ylabel('Cumulative return')
plt.xticks(rotation = 45)
plt.title('Portfolios Cumulative Returns over 1980-2021')
x_major_locator=MultipleLocator(12)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.grid(axis = 'y')
plt.legend()
```

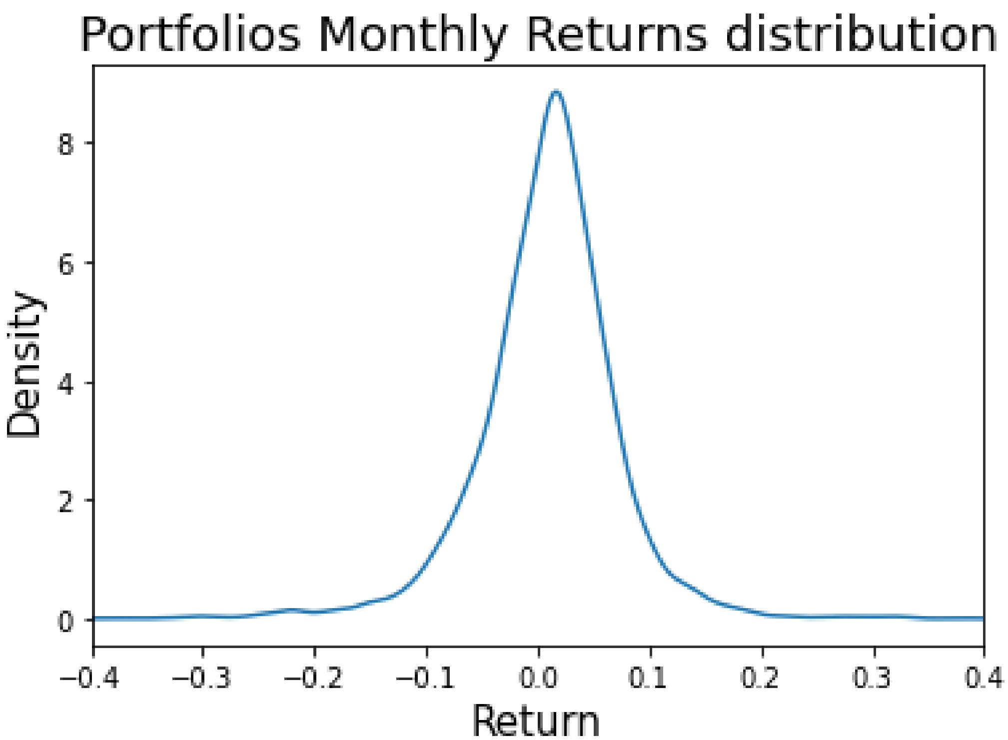
```
plt.show()
```



[29]:
```
mr['maxret'].plot.density()
plt.xlabel("Maxret", fontsize=15)
plt.ylabel("Density", fontsize=15)
plt.title("Stocks Monthly Maximum Returns distribution", fontsize=18)
plt.xlim(-0.4,0.4)
plt.show()
```



[30]:
```
vw_pf['vw_ret'].plot.density()
plt.xlabel("Return", fontsize=15)
plt.ylabel("Density", fontsize=15)
plt.title("Portfolios Monthly Returns distribution", fontsize=18)
plt.xlim(-0.4,0.4)
plt.show()
```



1.13 Sharpe Ratio

[31]:
```
# the sharpe ratio of Long-short Portfolio
sharp = ls_pf[['ym','ls_ret','rf',]]
sharp
```

[31]:

	ym	ls_ret	rf
0	1980-01	-0.100051	0.0080
1	1980-02	-0.022561	0.0089
2	1980-03	0.114231	0.0121
3	1980-04	-0.025424	0.0126
4	1980-05	0.001162	0.0081
..	…	…	…

```
495  2021-04  0.035703  0.0000
496  2021-05  0.011366  0.0000
497  2021-06 -0.067926  0.0000
498  2021-07  0.102997  0.0000
499  2021-08 -0.012072  0.0000

[500 rows x 3 columns]
```

[32]:
```
sharp.ls_ret.mean() - sharp.rf.mean()
```

[32]:
```
0.006118065377444524
```

[33]:
```
sharp.ls_ret.std()
```

[33]:
```
0.07958716614837075
```

[34]:
```
sharp_ratio = (sharp.ls_ret.mean() - sharp.rf.mean())/sharp.ls_ret.std()
sharp_ratio
```

[34]:
```
0.07687251190774769
```

1.14 Maximum Drawdown of the long-short strategy

[35]:
```
ls_nv = vis[['ym','cum_ls_ret']]
ls_nv = ls_nv.rename(columns = {'cum_ls_ret':'Net_value'})
ls_nv
```

[35]:
```
           ym  Net_value
0    1980-01   0.000000
1    1980-02  -0.120355
2    1980-03  -0.019872
3    1980-04  -0.044791
4    1980-05  -0.043681
..       ...        ...
495  2021-04  19.764335
496  2021-05  20.000346
497  2021-06  18.573868
498  2021-07  20.589915
499  2021-08  20.329281

[500 rows x 2 columns]
```

[36]:
```
def MaxDrawdown(ret,df):
    i = np.argmax((np.maximum.accumulate(ret) - ret))
    if i == 0:
        return 0
    j = np.argmax(ret[:i])
```

```
    draw_max = ret[j] - ret[i]
    draw_rate = draw_max / ret[j]
    draw_date_range = i - j
    return draw_max, draw_rate, draw_date_range, df.loc[df.index == i, 'ym'],
 ↪df.loc[df.index == j, 'ym']

maxdraw, maxrate, drawrange, end, start = MaxDrawdown(ls_nv.Net_value, ls_nv)
```

[37]:
```
maxdraw # Maximum Drawdown Net Value of the strategy
```

[37]: 47.24769132778143

[38]:
```
maxrate # Maximum Drawdown return of the strategy
```

[38]: 0.8355405441668534

[39]:
```
drawrange # Months
```

[39]: 137

[40]:
```
start.iloc[0] # Maximum net value day
```

[40]: '2002-09'

[41]:
```
end.iloc[0] # Minimum net value day
```

[41]: '2014-02'