Group Project 2 - Project Group 2 (Prasun Surana, Joanna Oghoore, Hongjiang Li)

Introduction

Price to Earnings, or P/E ratio is a common stock metric often used to gauge whether a stock is overvalued or undervalued. A low P/E ratio is commonly thought of as analogous to undervalued stocks, since the market is willing to pay a relatively low amount for the stock of a company based on it's earnings. Conversely, a high P/E ratio indicates taht investors are willing to pay a lot for a portion of the company's earnings. High P/E ratios are commonly seen in stocks that experience a lot of growth. For instance, the P/E ratio of Amazon in 2012 was 3633! However, it is important to note that just because a stock may have a large P/E ratio, it does not necessarily mean that it is overvalued.

A common investment strategy is to invest in stocks with a low P/E ratio; undervalued stocks trade at a discount to what would be considered a 'fair' value, and thus investors may be able to get that stock for a relatively cheap price. However, the strategy we will be exploring is a portfolio is high P/E stocks. We conduct this by taking the 10 stocks with the highest P/E ratios in each year, and rebalancing our portfolio every year with the new set of 10 stocks with the lowest P/E ratios for that year, all throughout the investment period of 39 years from 1980 to 2018.

Our first assumption is that high P/E is synonymos with growth stocks, so essentially our strategy investigates whether it is profitable to invest purely in growth stocks, despite their risk from possible overvaluation. Secondly, we also assume that there are no transaction costs. A portfolio that buys and sells stocks every year repeatedly would most certainly incur transaction costs which would eat into investment profits. For the purposes of our project, we have left these transaction costs out of consideration as there is no readily available data for it. Similarly, we assume there are no taxes. Our third assumption is that there is no issue in buying or selling a stock due to illiquidity. Some stocks may have a very small market cap, or a very small number of shares outstanding, which would make them hard to trade at the exact moment that we may want. For instance, if we wanted to buy a stock at a certain price, but that stock was particularly illiquid, we may have to wait for some time before we are able to make the purchase. Hence, we assume that there is no time lag in investment due to illiquidity, and that the stocks are bought exactly when we want: on the first day of every year.

To evaluate whether our portfolio has performed well, the benchmark we used is essentially the market returns. Specifically, these are the excess returns produced by the market portfolio over the risk-free rate. The reason we choose this as a benchmark is because we should not give credit to the market for the portion of returns it generates that we could get for free. If we invested in Treasury Bonds, which are issued by a very creditworthy government, these returns are almost guaranteed, i.e. risk-free. Therefore, the portion of the market returns that should be seriously considered are those that exist above the risk-free rate. Furthermore, it is very

common for an average investor to simply buy an ETF or a fund that tracks the broad market, or a broad subset of the market. Hence, it makes sense to use this as our benchmark and explore if our strategy can outperform it.

We believe our strategy should outperform the benchmark, because growth stocks tend to generate robust returns. Additionally, the structure of our portfolio is such that no stock remains in the portfolio for more than a year at a time (unless it consistently has the highest P/E ratios year after year). If an investor holds a growth stock for a long time horizon and the stock keeps increasing in price, becoming further overvalued, there may be more percieved risk in holding the stock, as it may be due for a correction. By rebalancing the portfolio every year, we avoid this, and instead aim to capture returns without holding a single stock for too long and allowing for the possibility of further overvaluation.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
```

In [2]:
```
# Import the CRSP Dataset
crsp = pd.read_csv('CRSP_Data.csv', parse_dates = ['date'], na_values=['A', 'B',

# Select only those PERMNOS which are common stock
crsp = crsp[crsp['SHRCD'].isin([10, 11])]

crsp.sort_values(['PERMNO', 'date', 'RET'], inplace=True)
crsp.drop_duplicates(subset=['PERMNO', 'date'], inplace=True)
crsp['date'] += pd.offsets.MonthEnd(0)
```

```
/Users/prasunsurana/opt/anaconda3/lib/python3.8/site-packages/IPython/core/inter
activeshell.py:3146: DtypeWarning: Columns (5) have mixed types.Specify dtype op
tion on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [3]:
```
# Using the rolling method to find annualized returns, stored in the column 'RET
crsp.loc[crsp.groupby('PERMNO').head(1).index, 'RET'] = np.nan
crsp['RET12'] = np.exp(np.log(1 + crsp['RET']).rolling(12).sum()) - 1
crsp1 = crsp[['PERMNO', 'date','PRC','RET','SHROUT','RET12']]
```

In [4]:
```
# Import Compustat
compustat = pd.read_csv('compustat-2021-01-14.csv', parse_dates=['datadate'])
compustat.sort_values(['LPERMNO', 'datadate', 'fyear'], inplace=True)
compustat.drop_duplicates(subset=['LPERMNO', 'datadate'], inplace=True, keep='la
compustat['datadate'] += pd.offsets.YearEnd(0)
```

In [5]:
```
compustat = compustat.rename(columns={'LPERMNO':'PERMNO', 'datadate':'date'})
compustat1 = compustat[['date','PERMNO','ni']].set_index(['date','PERMNO'])
```

In [6]:
```
# Import Fama-French Factors:
ff = pd.read_csv(
    'F-F_Research_Data_Factors.CSV',
    index_col=0,
    skiprows=3,
    nrows=12*(2020 - 1927 + 1) + 6 + 1
)
```

```
ff.index = pd.to_datetime(ff.index, format='%Y%m') + pd.offsets.MonthEnd(0)
ff /= 100
```

In [7]:
```
crsp1 = crsp[['PERMNO', 'date','PRC','RET','SHROUT','RET12']].dropna()
crsp1 = crsp1.set_index(['date'])
crsp1['PRC'] = crsp1['PRC'].abs()
crsp1 = crsp1[crsp1.index.year >= 1980]
```

In [8]:
```
cc = crsp1.merge(compustat1, how='right', on=['date','PERMNO'])
cc
```

Out[8]:

date	PERMNO	PRC	RET	SHROUT	RET12	ni
1986-12-31	10000	NaN	NaN	NaN	NaN	-0.730
1986-12-31	10001	NaN	NaN	NaN	NaN	0.669
1987-12-31	10001	5.87500	-0.033535	992.0	-0.101275	0.312
1988-12-31	10001	6.37500	-0.021132	998.0	0.163160	0.564
1989-12-31	10001	10.12500	0.037975	1022.0	0.687926	1.208
...	...	...	...	...	...	...
2015-12-31	93436	240.00999	0.042343	131425.0	0.079132	-888.663
2016-12-31	93436	213.69000	0.128247	161561.0	-0.109664	-674.914
2017-12-31	93436	311.35001	0.008095	168797.0	0.457017	-1961.400
2018-12-31	93436	332.79999	-0.050445	172602.0	0.068893	-976.091
2019-12-31	93436	418.32999	0.267897	181062.0	0.257003	-862.000

307514 rows × 6 columns

In [9]:
```
# Some of the prices in cc are also negative. Since prices cannot be negative, w
cc['PRC'] = cc['PRC'].abs()
```

In [10]:
```
# We now create a column to calculate the P/E ratio:
cc['p_e'] = cc['PRC']/(cc['ni']*1000000/(cc['SHROUT']*1000))
cc
```

Out[10]:

date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
1986-12-31	10000	NaN	NaN	NaN	NaN	-0.730	NaN
1986-12-31	10001	NaN	NaN	NaN	NaN	0.669	NaN
1987-12-31	10001	5.87500	-0.033535	992.0	-0.101275	0.312	18.679487
1988-12-31	10001	6.37500	-0.021132	998.0	0.163160	0.564	11.280585
1989-12-31	10001	10.12500	0.037975	1022.0	0.687926	1.208	8.566018
...	...	...	...	...	...	...	...
2015-12-31	93436	240.00999	0.042343	131425.0	0.079132	-888.663	-35.495247

date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
2016-12-31	93436	213.69000	0.128247	161561.0	-0.109664	-674.914	-51.153140
2017-12-31	93436	311.35001	0.008095	168797.0	0.457017	-1961.400	-26.794610
2018-12-31	93436	332.79999	-0.050445	172602.0	0.068893	-976.091	-58.848964
2019-12-31	93436	418.32999	0.267897	181062.0	0.257003	-862.000	-87.869681

307514 rows × 7 columns

In [11]:
```
# The P/E ratios shown in the dataframe above are calculated using the data for
# a high P/E stock can only be bought at the start of this year using LAST year'
# P/E column values down by 1.
```

In [12]:
```
cc['p_e'] = cc['p_e'].shift(1)
cc.dropna(inplace=True)
cc
```

Out[12]:

date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
1988-12-31	10001	6.37500	-0.021132	998.0	0.163160	0.564	18.679487
1989-12-31	10001	10.12500	0.037975	1022.0	0.687926	1.208	11.280585
1990-12-31	10001	9.50000	0.001299	1054.0	-0.008721	1.131	8.566018
1991-12-31	10001	14.50000	-0.006780	1075.0	0.607471	1.073	8.853227
1992-12-31	10001	14.00000	-0.015130	1080.0	0.012621	0.834	14.527027
...	...	...	...	...	...	...	...
2015-12-31	93436	240.00999	0.042343	131425.0	0.079132	-888.663	-95.069610
2016-12-31	93436	213.69000	0.128247	161561.0	-0.109664	-674.914	-35.495247
2017-12-31	93436	311.35001	0.008095	168797.0	0.457017	-1961.400	-51.153140
2018-12-31	93436	332.79999	-0.050445	172602.0	0.068893	-976.091	-26.794610
2019-12-31	93436	418.32999	0.267897	181062.0	0.257003	-862.000	-58.848964

162800 rows × 7 columns

In [13]:
```
# We discard the P/E ratios that are negative. Negative P/E ratios mean that the
# we focus on profitable businesses, and out of those, we will select the busine
cc = cc[cc['p_e'] > 0]
cc.replace([np.inf, -np.inf], np.nan, inplace=True)
cc = cc.dropna()
cc
```

```
/Users/prasunsurana/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.
py:4379: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
```

```
le/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().replace(
```

Out[13]:

date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
1988-12-31	10001	6.375	-0.021132	998.0	0.163160	0.564	18.679487
1989-12-31	10001	10.125	0.037975	1022.0	0.687926	1.208	11.280585
1990-12-31	10001	9.500	0.001299	1054.0	-0.008721	1.131	8.566018
1991-12-31	10001	14.500	-0.006780	1075.0	0.607471	1.073	8.853227
1992-12-31	10001	14.000	-0.015130	1080.0	0.012621	0.834	14.527027
...	...	...	...	...	...	...	...
2018-12-31	93429	97.830	-0.090968	112202.0	-0.206040	425.200	35.063408
2019-12-31	93429	120.000	0.009251	110777.0	0.241574	374.400	25.815432
2013-12-31	93434	6.950	0.210801	11598.0	-0.099742	-2.516	162.078827
2015-12-31	93434	4.220	-0.040909	14673.0	0.055000	-3.163	138.906166
2017-12-31	93434	3.900	0.147059	24328.0	-0.152175	-11.822	226.193973

115552 rows × 7 columns

In [14]:
```
cc.sort_values(['date','p_e'], ascending = [1,1], inplace=True)
```

In [15]:
```
df = pd.DataFrame(columns = ['date','PERMNO','PRC','RET','SHROUT','RET12','ni','
df.set_index('date')
```

Out[15]:

date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e

In [16]:
```
# Creating the portfolio by selecting the 10 stocks every year with the highest
curr_year = 1980
for i in range(0,len(cc)):
    if curr_year != cc.index.year[i]:
        df = pd.concat([df,cc.iloc[i-10:i,:]])
        curr_year += 1

df
```

Out[16]:

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
1980-12-31	NaN	64565	5.06250	-0.065368	3281.0	0.684942	2.328	23.623766
1980-12-31	NaN	64195	27.62500	-0.071429	3734.0	0.524137	3.137	31.975344
1980-12-31	NaN	12045	0.29688	-0.050000	27778.0	0.187499	-0.484	37.260183
1980-12-31	NaN	60150	17.25000	0.041509	1198.0	-0.026593	3.621	38.627103

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e
1980-12-31	NaN	80013	17.75000	-0.134146	1498.0	0.302754	2.110	48.885205
...	...	...	...	...	...	...	...	...
2018-12-31	NaN	24272	70.96000	-0.133472	62487.0	0.009061	114.000	4430.420000
2018-12-31	NaN	76672	10.17000	-0.211016	48977.0	0.006931	32.243	9094.573585
2018-12-31	NaN	12089	48.19000	-0.065723	93650.0	0.087811	34.725	9774.201592
2018-12-31	NaN	42286	31.89000	-0.076012	94840.0	0.009977	200.100	10167.700933
2018-12-31	NaN	15351	14.34000	-0.098680	84051.0	-0.222126	5.882	11269.671343

390 rows × 8 columns

In [17]:
```
# Creating the investment values for each stock in each year. We start with 1000
# after a year in the column 'Value'. At the end of every year, we sum our retur
# For our investments next year, we divide this portfolio value by 10, and inves
# next year. We continue this process until the end of 2018.
df['Investment'] = np.nan
df.iloc[0:10,8] = 1000

df['Value'] = np.nan

d = 1980
inv = 1000
s = 0
for i in range(0,len(df)):
    if d == df.index.year[i]:
        df.iloc[i,9] = inv*(1+df.iloc[i,5])
        df.iloc[i,8] = inv
        s = s+inv*(1+df.iloc[i,5])
    else:
        inv = s/10
        df.iloc[i,9] = inv*(1+df.iloc[i,5])
        df.iloc[i,8] = inv
        s=inv*(1+df.iloc[i,5])
        d = df.index.year[i]

df.head(10)
```

Out[17]:

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e	Investme
1980-12-31	NaN	64565	5.06250	-0.065368	3281.0	0.684942	2.328	23.623766	100
1980-12-31	NaN	64195	27.62500	-0.071429	3734.0	0.524137	3.137	31.975344	100
1980-12-31	NaN	12045	0.29688	-0.050000	27778.0	0.187499	-0.484	37.260183	100
1980-12-31	NaN	60150	17.25000	0.041509	1198.0	-0.026593	3.621	38.627103	100

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e	Investme
1980-12-31	NaN	80013	17.75000	-0.134146	1498.0	0.302754	2.110	48.885205	100
1980-12-31	NaN	75791	8.56250	0.045802	5320.0	1.795914	-0.269	61.712593	100
1980-12-31	NaN	18060	12.06250	0.026596	1620.0	0.787035	0.249	66.918648	100
1980-12-31	NaN	58473	10.37500	-0.194175	1832.0	-0.284483	-0.593	115.660037	100
1980-12-31	NaN	69228	1.10938	0.126984	31468.0	10.833329	-0.213	3776.250000	100
1980-12-31	NaN	63693	63.00000	-0.030769	1846.0	0.826085	3.205	4422.686786	100

In [18]:
```
# Creating a column 'Value2' to store the cumulative portfolio value at the end
df['Value2'] = np.nan
df2 = df.reset_index()
df2 = df2.drop(columns='date')
df2 = df2.rename(columns = {'index':'date'})
```

In [19]:
```
# Summing up the stock returns to obtain the portfolio value at the end of each
curr_year1 = 1980
for i in range(0,len(df2)):
    if df2['date'].dt.year[i] >= curr_year1:
        df2.iloc[i-1,10] = df2.iloc[i-10:i,9].sum()
        curr_year1 += 1
    else:
         df2.iloc[i-1,10] = np.nan

df2.head(30)
```

Out[19]:

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e	Investme
0	1980-12-31	64565	5.06250	-0.065368	3281.0	0.684942	2.328	23.623766	1000.0000
1	1980-12-31	64195	27.62500	-0.071429	3734.0	0.524137	3.137	31.975344	1000.0000
2	1980-12-31	12045	0.29688	-0.050000	27778.0	0.187499	-0.484	37.260183	1000.0000
3	1980-12-31	60150	17.25000	0.041509	1198.0	-0.026593	3.621	38.627103	1000.0000
4	1980-12-31	80013	17.75000	-0.134146	1498.0	0.302754	2.110	48.885205	1000.0000
5	1980-12-31	75791	8.56250	0.045802	5320.0	1.795914	-0.269	61.712593	1000.0000
6	1980-12-31	18060	12.06250	0.026596	1620.0	0.787035	0.249	66.918648	1000.0000
7	1980-12-31	58473	10.37500	-0.194175	1832.0	-0.284483	-0.593	115.660037	1000.0000

	date	PERMNO	PRC	RET	SHROUT	RET12	ni	p_e	Investme
8	1980-12-31	69228	1.10938	0.126984	31468.0	10.833329	-0.213	3776.250000	1000.0000
9	1980-12-31	63693	63.00000	-0.030769	1846.0	0.826085	3.205	4422.686786	1000.0000
10	1981-12-31	11105	11.62500	0.120482	13510.0	0.015805	-10.916	593.846154	2563.0617
11	1981-12-31	38392	36.50000	-0.087500	4753.0	0.327899	-0.394	599.897177	2563.0617
12	1981-12-31	63088	22.37500	0.104938	2039.0	0.787314	3.770	808.888889	2563.0617
13	1981-12-31	83679	0.40625	-0.235294	16903.0	-0.723404	-0.505	883.243304	2563.0617
14	1981-12-31	12264	2.96875	0.055556	7516.0	-0.525000	-0.060	1159.218750	2563.0617
15	1981-12-31	46384	6.25000	-0.152542	5098.0	-0.180328	0.049	1525.000000	2563.0617
16	1981-12-31	63853	2.93750	-0.113208	15595.0	-0.661871	0.903	1874.192383	2563.0617
17	1981-12-31	19933	4.06250	-0.244186	10031.0	-0.380952	0.602	2148.505435	2563.0617
18	1981-12-31	29963	0.65625	0.000000	26677.0	-0.743902	0.077	4748.312500	2563.0617
19	1981-12-31	27677	0.35938	-0.148148	32623.0	-0.460290	0.461	6721.875000	2563.0617
20	1982-12-31	46384	1.68750	0.173913	5098.0	-0.730000	-0.634	650.255102	1910.8318
21	1982-12-31	63175	2.25000	-0.142857	7744.0	-0.718750	-2.265	710.476190	1910.8318
22	1982-12-31	33196	0.68750	-0.083333	2665.0	-0.717949	-0.773	808.031250	1910.8318
23	1982-12-31	10218	6.75000	0.102041	3721.0	1.400000	0.193	872.642045	1910.8318
24	1982-12-31	68495	7.87500	0.125000	625.0	0.016129	-0.070	968.750000	1910.8318
25	1982-12-31	69201	2.50000	0.000000	801.0	-0.090910	0.197	1101.375000	1910.8318
26	1982-12-31	79987	0.81250	-0.187500	16642.0	-0.653333	0.358	1128.338068	1910.8318
27	1982-12-31	47985	0.29688	-0.095238	13135.0	-0.869863	-1.952	1866.237981	1910.8318
28	1982-12-31	50842	3.81250	-0.164384	7761.0	-0.175677	-0.333	2239.945312	1910.8318
29	1982-12-31	74414	4.37500	0.206897	4816.0	1.500002	-0.294	8115.463415	1910.8318

In [20]:
```
df2.iloc[389,10] = 324236.7044
df3 = df2[['date','Value2']].dropna()

# Finding the annual return for our portfolio year-on-year:
df3['RET_pe'] = df3['Value2'].pct_change()
df3
```

Out[20]:

	date	Value2	RET_pe
9	1980-12-31	25630.617037	NaN
19	1981-12-31	19108.318083	-0.254473
29	1982-12-31	17120.380946	-0.104035
39	1983-12-31	59328.130222	2.465351
49	1984-12-31	44398.191774	-0.251650
59	1985-12-31	45025.537276	0.014130
69	1986-12-31	39318.328247	-0.126755
79	1987-12-31	32058.698818	-0.184637
89	1988-12-31	39254.612437	0.224461
99	1989-12-31	49382.075243	0.257994
109	1990-12-31	41340.533010	-0.162843
119	1991-12-31	64849.926019	0.568677
129	1992-12-31	86357.541024	0.331652
139	1993-12-31	107538.491359	0.245270
149	1994-12-31	88519.795488	-0.176855
159	1995-12-31	120835.290615	0.365065
169	1996-12-31	132978.599626	0.100495
179	1997-12-31	160832.013014	0.209458
189	1998-12-31	172072.580814	0.069890
199	1999-12-31	312208.373035	0.814399
209	2000-12-31	290879.863945	-0.068315
219	2001-12-31	190595.321307	-0.344763
229	2002-12-31	131100.138672	-0.312154
239	2003-12-31	241801.957777	0.844407
249	2004-12-31	296527.419462	0.226323
259	2005-12-31	256448.341442	-0.135161
269	2006-12-31	271541.513647	0.058855
279	2007-12-31	225384.223912	-0.169982
289	2008-12-31	144161.743605	-0.360373
299	2009-12-31	193863.396361	0.344763
309	2010-12-31	231008.749344	0.191606

	date	Value2	RET_pe
319	2011-12-31	246785.303138	0.068294
329	2012-12-31	252315.557323	0.022409
339	2013-12-31	365258.220877	0.447625
349	2014-12-31	278460.363534	-0.237634
359	2015-12-31	275231.826260	-0.011594
369	2016-12-31	262460.082929	-0.046404
379	2017-12-31	285918.597404	0.089379
389	2018-12-31	324236.704400	0.134018

In [21]:
```
ff2 = ff.copy()
ff2 = ff2[ff2.index.year >= 1980]
ff2 = ff2[ff2.index.year < 2019]
ff2
```

Out[21]:

	Mkt-RF	SMB	HML	RF
1980-01-31	0.0551	0.0165	0.0172	0.0080
1980-02-29	-0.0122	-0.0183	0.0066	0.0089
1980-03-31	-0.1290	-0.0665	-0.0102	0.0121
1980-04-30	0.0397	0.0096	0.0101	0.0126
1980-05-31	0.0526	0.0216	0.0038	0.0081
...	...	...	...	...
2018-08-31	0.0344	0.0115	-0.0392	0.0016
2018-09-30	0.0006	-0.0228	-0.0163	0.0015
2018-10-31	-0.0768	-0.0478	0.0346	0.0019
2018-11-30	0.0169	-0.0071	0.0034	0.0018
2018-12-31	-0.0955	-0.0248	-0.0189	0.0019

468 rows × 4 columns

In [22]:
```
# Creating rolling annual returns for market return and SMB:
pd.options.mode.chained_assignment = None
ff2['MktRET12'] = np.exp(np.log(1 + ff2['Mkt-RF']).rolling(12, min_periods = 1).
ff2['SMB12'] = np.exp(np.log(1 + ff2['SMB']).rolling(12, min_periods = 1).sum())
ff2 = ff2[ff2.index.month == 12]
ff2['Value'] = np.nan

# Finding the portfolio value for MktRET12 at the end of each year over the inve
ff2.iloc[0,6] = 10000*(1+ff2.iloc[0,4])
for i in range(1,len(ff2)):
    ff2.iloc[i,6] = ff2.iloc[i-1,6]*(1 + ff2.iloc[i,4])

ff2
```

Out[22]:

	Mkt-RF	SMB	HML	RF	MktRET12	SMB12	Value
1980-12-31	-0.0452	-0.0027	0.0273	0.0131	0.199416	0.053049	11994.164006
1981-12-31	-0.0365	0.0117	0.0075	0.0087	-0.159860	0.073159	10076.774501
1982-12-31	0.0055	-0.0019	-0.0003	0.0067	0.096447	0.070677	11048.651621
1983-12-31	-0.0178	-0.0029	0.0168	0.0073	0.127092	0.112223	12452.843158
1984-12-31	0.0184	-0.0060	-0.0020	0.0064	-0.055464	-0.080020	11762.159534
1985-12-31	0.0388	-0.0049	-0.0152	0.0065	0.232819	0.002139	14500.610087
1986-12-31	-0.0327	0.0009	0.0027	0.0049	0.095802	-0.088336	15889.794472
1987-12-31	0.0681	0.0013	-0.0445	0.0039	-0.037475	-0.088500	15294.329016
1988-12-31	0.0149	0.0194	-0.0154	0.0063	0.108976	0.048250	16961.049248
1989-12-31	0.0116	-0.0240	0.0021	0.0061	0.190490	-0.101446	20191.966217
1990-12-31	0.0246	0.0075	-0.0172	0.0060	-0.130431	-0.144784	17558.313680
1991-12-31	0.1084	-0.0226	-0.0411	0.0038	0.277838	0.127686	22436.685919
1992-12-31	0.0153	0.0163	0.0263	0.0028	0.060218	0.063769	23787.788810
1993-12-31	0.0165	0.0122	0.0036	0.0023	0.079950	0.058153	25689.610833
1994-12-31	0.0086	0.0030	-0.0020	0.0044	-0.039530	-0.005730	24674.106109
1995-12-31	0.0103	0.0064	0.0033	0.0049	0.297179	-0.075036	32006.743744
1996-12-31	-0.0170	0.0308	0.0133	0.0046	0.152322	-0.037611	36882.086035
1997-12-31	0.0132	-0.0266	0.0403	0.0048	0.247828	-0.070513	46022.492453
1998-12-31	0.0616	-0.0054	-0.0420	0.0038	0.186208	-0.198608	54592.234772
1999-12-31	0.0772	0.0707	-0.0840	0.0044	0.197141	0.133903	65354.598285
2000-12-31	0.0119	0.0073	0.0767	0.0050	-0.167004	-0.075810	54440.097186
2001-12-31	0.0161	0.0479	0.0090	0.0015	-0.146904	0.199921	46442.609230
2002-12-31	-0.0576	-0.0001	0.0228	0.0011	-0.224183	0.044091	36030.972370
2003-12-31	0.0429	-0.0276	0.0158	0.0008	0.304712	0.206211	47010.027187
2004-12-31	0.0343	-0.0007	-0.0018	0.0016	0.106307	0.041953	52007.544177
2005-12-31	-0.0025	-0.0046	0.0024	0.0032	0.029984	-0.019089	53566.960387
2006-12-31	0.0087	-0.0114	0.0272	0.0040	0.101593	0.003808	59009.002857
2007-12-31	-0.0087	0.0014	-0.0059	0.0027	0.009762	-0.067438	59585.069147
2008-12-31	0.0174	0.0351	0.0014	0.0000	-0.377760	0.051329	37076.225149
2009-12-31	0.0275	0.0613	-0.0011	0.0001	0.282348	0.081426	47544.631761
2010-12-31	0.0682	0.0068	0.0364	0.0001	0.173538	0.132364	55795.425396
2011-12-31	0.0074	-0.0066	0.0168	0.0000	0.004352	-0.048285	56038.244883
2012-12-31	0.0118	0.0145	0.0353	0.0001	0.162529	-0.007374	65146.102594
2013-12-31	0.0281	-0.0059	-0.0005	0.0000	0.351792	0.052186	88063.992027
2014-12-31	-0.0006	0.0248	0.0223	0.0000	0.117080	-0.068573	98374.559791

	Mkt-RF	SMB	HML	RF	MktRET12	SMB12	Value
2015-12-31	-0.0217	-0.0286	-0.0245	0.0001	0.000794	-0.039956	98452.698399
2016-12-31	0.0181	0.0011	0.0351	0.0003	0.132656	0.060723	111513.073021
2017-12-31	0.0106	-0.0129	0.0017	0.0009	0.213586	-0.042314	135330.684432
2018-12-31	-0.0955	-0.0248	-0.0189	0.0019	-0.068096	-0.028940	126115.176250

In [23]:
```
ff3 = ff2.reset_index()
ff3.rename({'index':'date'}, inplace=True, axis=1)
```

In [24]:
```
merged = df3.merge(ff3, on='date')
merged
```

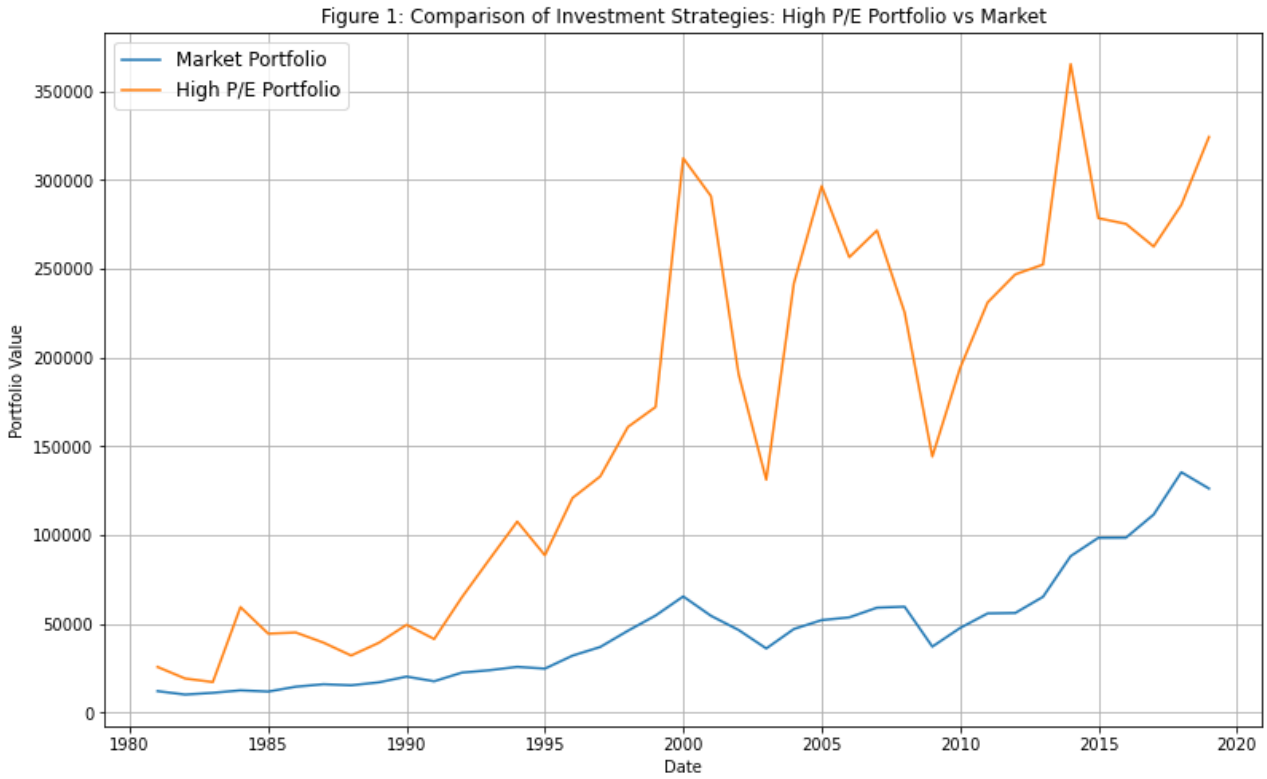
Out[24]:

	date	Value2	RET_pe	Mkt-RF	SMB	HML	RF	MktRET12	SMB12
0	1980-12-31	25630.617037	NaN	-0.0452	-0.0027	0.0273	0.0131	0.199416	0.053049
1	1981-12-31	19108.318083	-0.254473	-0.0365	0.0117	0.0075	0.0087	-0.159860	0.073159
2	1982-12-31	17120.380946	-0.104035	0.0055	-0.0019	-0.0003	0.0067	0.096447	0.070677
3	1983-12-31	59328.130222	2.465351	-0.0178	-0.0029	0.0168	0.0073	0.127092	0.112223
4	1984-12-31	44398.191774	-0.251650	0.0184	-0.0060	-0.0020	0.0064	-0.055464	-0.080020
5	1985-12-31	45025.537276	0.014130	0.0388	-0.0049	-0.0152	0.0065	0.232819	0.002139
6	1986-12-31	39318.328247	-0.126755	-0.0327	0.0009	0.0027	0.0049	0.095802	-0.088336
7	1987-12-31	32058.698818	-0.184637	0.0681	0.0013	-0.0445	0.0039	-0.037475	-0.088500
8	1988-12-31	39254.612437	0.224461	0.0149	0.0194	-0.0154	0.0063	0.108976	0.048250
9	1989-12-31	49382.075243	0.257994	0.0116	-0.0240	0.0021	0.0061	0.190490	-0.101446
10	1990-12-31	41340.533010	-0.162843	0.0246	0.0075	-0.0172	0.0060	-0.130431	-0.144784
11	1991-12-31	64849.926019	0.568677	0.1084	-0.0226	-0.0411	0.0038	0.277838	0.127686
12	1992-12-31	86357.541024	0.331652	0.0153	0.0163	0.0263	0.0028	0.060218	0.063769
13	1993-12-31	107538.491359	0.245270	0.0165	0.0122	0.0036	0.0023	0.079950	0.058153
14	1994-12-31	88519.795488	-0.176855	0.0086	0.0030	-0.0020	0.0044	-0.039530	-0.005730
15	1995-12-31	120835.290615	0.365065	0.0103	0.0064	0.0033	0.0049	0.297179	-0.075036

	date	Value2	RET_pe	Mkt-RF	SMB	HML	RF	MktRET12	SMB12
16	1996-12-31	132978.599626	0.100495	-0.0170	0.0308	0.0133	0.0046	0.152322	-0.037611
17	1997-12-31	160832.013014	0.209458	0.0132	-0.0266	0.0403	0.0048	0.247828	-0.070513
18	1998-12-31	172072.580814	0.069890	0.0616	-0.0054	-0.0420	0.0038	0.186208	-0.198608
19	1999-12-31	312208.373035	0.814399	0.0772	0.0707	-0.0840	0.0044	0.197141	0.133903
20	2000-12-31	290879.863945	-0.068315	0.0119	0.0073	0.0767	0.0050	-0.167004	-0.075810
21	2001-12-31	190595.321307	-0.344763	0.0161	0.0479	0.0090	0.0015	-0.146904	0.199921
22	2002-12-31	131100.138672	-0.312154	-0.0576	-0.0001	0.0228	0.0011	-0.224183	0.044091
23	2003-12-31	241801.957777	0.844407	0.0429	-0.0276	0.0158	0.0008	0.304712	0.206211
24	2004-12-31	296527.419462	0.226323	0.0343	-0.0007	-0.0018	0.0016	0.106307	0.041953
25	2005-12-31	256448.341442	-0.135161	-0.0025	-0.0046	0.0024	0.0032	0.029984	-0.019089
26	2006-12-31	271541.513647	0.058855	0.0087	-0.0114	0.0272	0.0040	0.101593	0.003808
27	2007-12-31	225384.223912	-0.169982	-0.0087	0.0014	-0.0059	0.0027	0.009762	-0.067438
28	2008-12-31	144161.743605	-0.360373	0.0174	0.0351	0.0014	0.0000	-0.377760	0.051329
29	2009-12-31	193863.396361	0.344763	0.0275	0.0613	-0.0011	0.0001	0.282348	0.081426
30	2010-12-31	231008.749344	0.191606	0.0682	0.0068	0.0364	0.0001	0.173538	0.132364
31	2011-12-31	246785.303138	0.068294	0.0074	-0.0066	0.0168	0.0000	0.004352	-0.048285
32	2012-12-31	252315.557323	0.022409	0.0118	0.0145	0.0353	0.0001	0.162529	-0.007374
33	2013-12-31	365258.220877	0.447625	0.0281	-0.0059	-0.0005	0.0000	0.351792	0.052186
34	2014-12-31	278460.363534	-0.237634	-0.0006	0.0248	0.0223	0.0000	0.117080	-0.068573
35	2015-12-31	275231.826260	-0.011594	-0.0217	-0.0286	-0.0245	0.0001	0.000794	-0.039956
36	2016-12-31	262460.082929	-0.046404	0.0181	0.0011	0.0351	0.0003	0.132656	0.060723
37	2017-12-31	285918.597404	0.089379	0.0106	-0.0129	0.0017	0.0009	0.213586	-0.042314

	date	Value2	RET_pe	Mkt-RF	SMB	HML	RF	MktRET12	SMB12
38	2018-12-31	324236.704400	0.134018	-0.0955	-0.0248	-0.0189	0.0019	-0.068096	-0.028940

In [25]:
```
plt.figure(figsize = (13,8))
plt.plot(merged['date'], merged['Value'], label = "Market Portfolio")
plt.plot(merged['date'], merged['Value2'], label = 'High P/E Portfolio')
plt.title("Figure 1: Comparison of Investment Strategies: High P/E Portfolio vs
plt.xlabel('Date')
plt.ylabel('Portfolio Value')
plt.legend(fontsize = 12)
plt.grid()
plt.show()
```



From Figure 1 above, we can see that at all times, the high P/E portfolio generates greater returns than the market portfolio. It is important to note that the 2 portfolio values generally rise and fall in unison, but to differing degrees. The downturns in portfolio values seem to happen around the time of major financial crises; the large downswing in 2002-2003 could be the aftermath of the 2001 financial crash following the dotcom bubble, and the downswing in 2007 - 2008 is ostensibly the Great Recession. In both cases, the portfolio value of the high P/E stocks falls far more precipitously than the market portfolio. Broadly speaking, value stocks tend to outperform growth stocks in recessions, which may explain why our portfolio suffers greater declines in times of economic distress than the market portfolio. However, our portfolio does indeed exceed the benchmark in performance over the investment period.

In [26]:
```
# Final Value for High P/E Portfolio:
merged.iloc[38,1]
```

Out[26]:
```
324236.7044
```

In [27]:
```
# Final Value for High P/E Portfolio:
merged.iloc[38,9]
```

Out[27]:
```
126115.17625041348
```

In [28]:
```
# How much more money would the high P/E portfolio generate over the market port
merged.iloc[38,1] - merged.iloc[38,9]
```

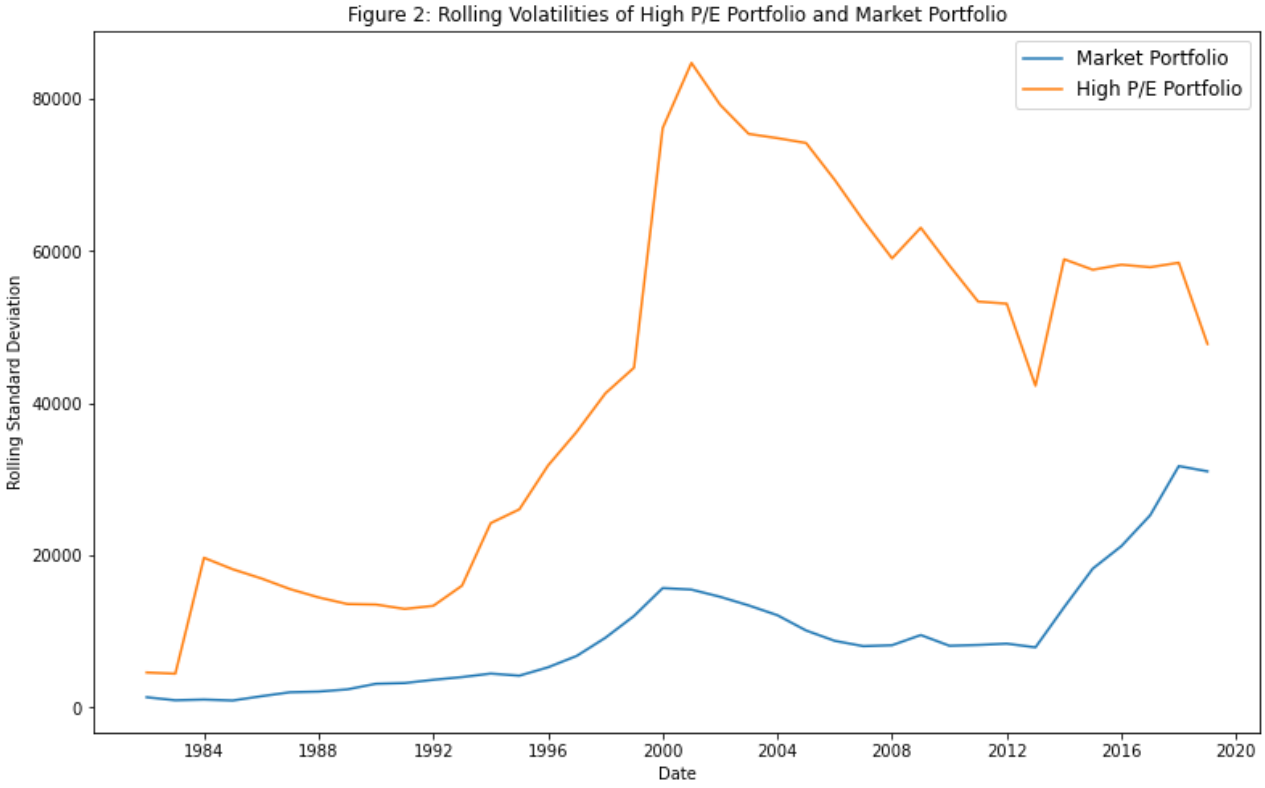
Out[28]:
```
198121.52814958652
```

Rolling Volatilities

In [29]:
```
merged['StDev_pe'] = merged['Value2'].rolling(window=10, min_periods=1).std()
merged['StDev_market'] = merged['Value'].rolling(window=10, min_periods=1).std()
```

In [30]:
```
plt.figure(figsize = (13,8))
plt.plot(merged['date'], merged['StDev_market'], label = "Market Portfolio")
plt.plot(merged['date'], merged['StDev_pe'], label = 'High P/E Portfolio')
plt.title("Figure 2: Rolling Volatilities of High P/E Portfolio and Market Portf
plt.xlabel('Date')
plt.ylabel('Rolling Standard Deviation')
plt.legend(fontsize = 12)
```

Out[30]:
```
<matplotlib.legend.Legend at 0x7fe9c1376be0>
```



In Figure 1, we discussed that the downswings in our portfolio reacts disproportionately to economic downturns. We verify this by plotting the rolling volatilites. Again, the volatilities for both portfolios seem to move in tandem, but in absolute terms, the high P/E portfolio always has greater volatilities in all years as compared to the benchmark. The volatility for the high P/E portfolio also changes by a greater amount every time there is a volatility swing as compared to the benchmark. So while our portfolio generates far superior returns, by almost a factor of 3, it

also has greater risk. This is unsurprising given that growth stocks tend to be more risky than stable value stocks. Therefore, an investor in a portfolio of exclusively high P/E stocks must be aware that althought they will enjoy higher returns, there is far greater risk and that their returns will suffer by a greater proportion than if they had invested in a total market ETF or a similar security that gave them more exposure to the broader market.

Examining Sharpe Ratios of High P/E Portfolio vs Market Portfolio

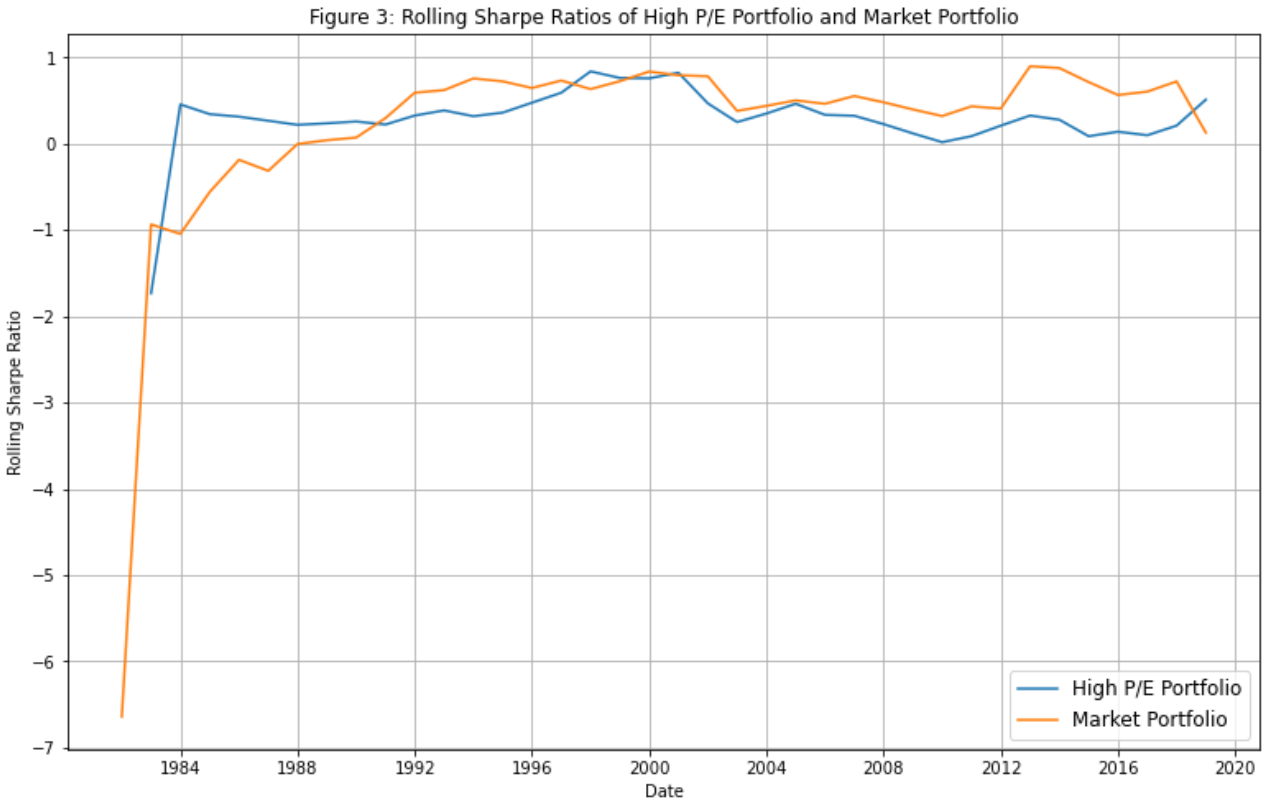
From the rolling volatilities, it seems that our portfolio is much more risky. However, it is more telling to examine how much return we are getting for the risk we take on.

In [31]:
```
# Excess returns, i.e. how much more the return on the portfolio made than the r
# the market portfolio already exists as Mkt-RF.
merged['RET_pe-RF'] = merged['RET_pe'] - merged['RF']

# # Adding columns for Annualized Rolling Sharpe Ratios:
merged['SR252_pe'] = merged['RET_pe-RF'].rolling(10, min_periods=1).mean() / mer
merged['SR252_market'] = merged['Mkt-RF'].rolling(10, min_periods=1).mean() / me

# Plotting the Rolling Sharpe Ratios for both portfolios:
plt.figure(figsize = (13,8))
plt.plot(merged['date'], merged['SR252_pe'], label = "High P/E Portfolio")
plt.plot(merged['date'], merged['SR252_market'], label = 'Market Portfolio')
plt.title("Figure 3: Rolling Sharpe Ratios of High P/E Portfolio and Market Port
plt.xlabel('Date')
plt.ylabel('Rolling Sharpe Ratio')
plt.grid()
plt.legend(fontsize = 12)
```

Out[31]:
```
<matplotlib.legend.Legend at 0x7fe9a27104f0>
```



From the previous section on rolling volatilities, we observed that the volatility for the high P/E portfolio was much higher than the benchmark. However, from Figure 3 above, we observe that the risk endured in the high P/E portfolio is actually commensurate to it's return, as it's rolling Sharpe ratios are very similar to the that of the market portfolio's. While the market portfolio, on average, has slighty higher Sharpe Ratios, the difference is negligible as they remain in a similar range. Hence, our portfolio does not take on an abnormal amount of risk given the returns it generates.

Sortino Ratios

However, not all risk is the same. As investors, we would LIKE to have upside risk, since upside risk essentially means higher returns. The only risk that investors are really wary of is downside risk. In a fashion similar to how we compared Sharpe Ratios for the two portfolios, we can go a step further and also compare Sortino Ratios, which measure the excess return enjoyed by the investor per unit of DOWNSIDE risk.

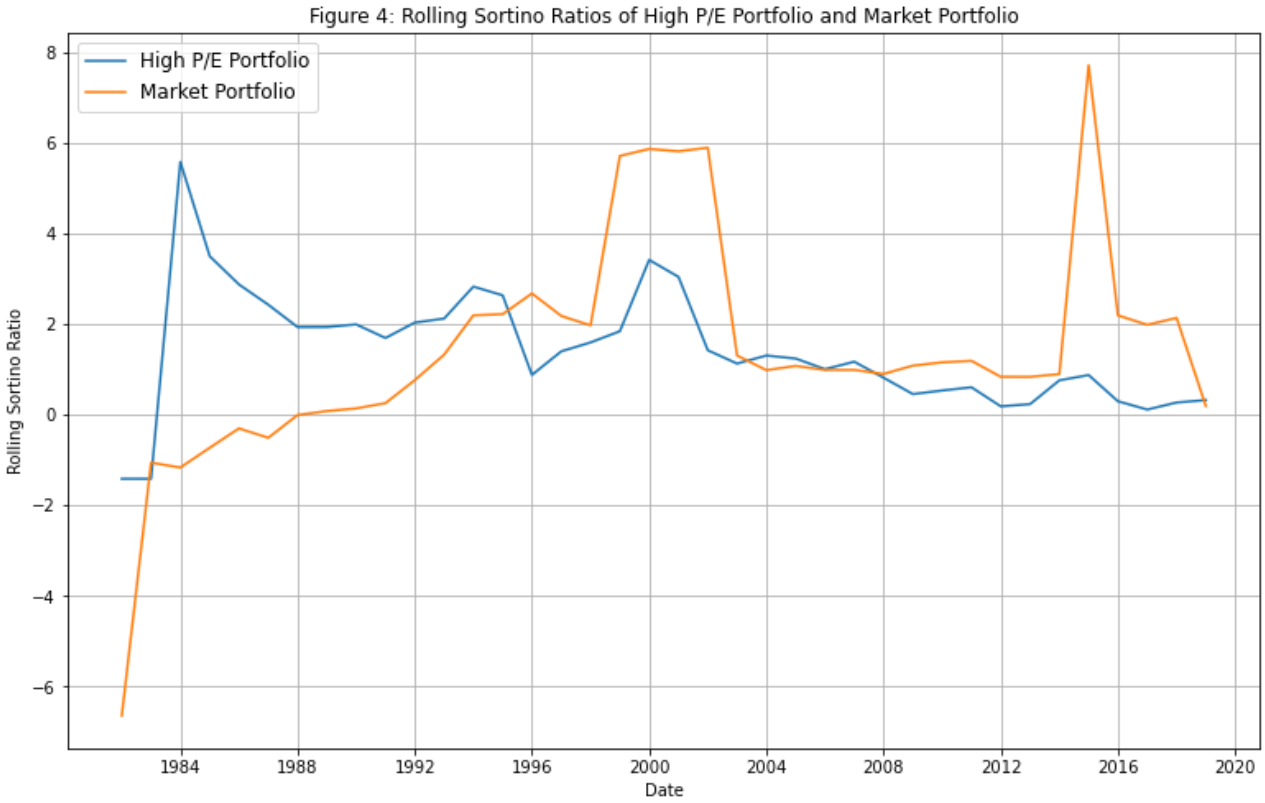
In [32]:
```
# Creating columns for negative excess returns for the portfolios:
merged['RET_pe-RF_neg'] = np.where(merged['RET_pe-RF'] < 0, merged['RET_pe-RF'],
merged['Mkt-RF_neg'] = np.where(merged['Mkt-RF'] < 0, merged['Mkt-RF'], 0)

# Adding columns for Annualized Rolling Sortino Ratios:
merged['SO252_pe'] = merged['RET_pe-RF'].rolling(window=12, min_periods = 1).mea
merged['SO252_market'] = merged['Mkt-RF'].rolling(window=12, min_periods = 1).me

# Plotting the Rolling Sortino Ratios for the portfolios:
plt.figure(figsize = (13,8))
plt.plot(merged['date'], merged['SO252_pe'], label = "High P/E Portfolio")
plt.plot(merged['date'], merged['SO252_market'], label = 'Market Portfolio')
plt.title("Figure 4: Rolling Sortino Ratios of High P/E Portfolio and Market Por
```

```
plt.xlabel('Date')
plt.ylabel('Rolling Sortino Ratio')
plt.grid()
plt.legend(fontsize = 12)
```

Out[32]: <matplotlib.legend.Legend at 0x7fe9a504b640>



The difference between the reward-to-risk ratios becomes more apparent from the Sortino Ratios in Figure 4 above. In the Sharpe Ratios, the upside risk was also included, but this is the risk that investors would enjoy. Despite having higher returns, the high P/E portfolio has a proportionally higher level of downside risk, causing it's Sortino Ratio to be smaller than that of the market portfolio. Therefore, in dollar value our investment strategy is far superior, but it must be mentioned that from a risk-assessment standpoint, it is not optimal, since the returns would need to be even higher to make it's Sortino ratios similar to that of the market portofolio's.

We can also find alpha and beta for the portfolio. Alpha indicates how the portfolio performed over the benchmark, and beta measures how volatile the portfolio has been compared to the market as a whole. We regress the portfolio's annualized return on the market's annualized return. We also add the annualized returns for SMB (small minus big) into the regression. This is because we did not consider the size of the companies who's stocks we held in the portfolio. Whatever portion of the our portfolio's return was caused by the size of the companies will now be factored out.

In [33]:
```
reg1 = smf.ols(formula = 'RET_pe ~ MktRET12 + SMB12', data = merged)
results1 = reg1.fit()
results1.summary()
```

Out[33]:

OLS Regression Results

Dep. Variable:	RET_pe	**R-squared:**	0.339

Model:	OLS	**Adj. R-squared:**	0.301
Method:	Least Squares	**F-statistic:**	8.965
Date:	Wed, 28 Apr 2021	**Prob (F-statistic):**	0.000718
Time:	23:59:04	**Log-Likelihood:**	-18.061
No. Observations:	38	**AIC:**	42.12
Df Residuals:	35	**BIC:**	47.04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0197	0.073	0.270	0.788	-0.128	0.168
MktRET12	1.3377	0.409	3.268	0.002	0.507	2.169
SMB12	1.7121	0.731	2.342	0.025	0.228	3.196

Omnibus:	64.804	**Durbin-Watson:**	2.216
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	656.282
Skew:	3.920	**Prob(JB):**	3.09e-143
Kurtosis:	21.789	**Cond. No.**	11.2

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

We can see from the regression above that alpha, which is shown by the intercept, is 0.0197, or 1.97%. Beta is 1.3377, which indicates that our portfolio returns do indeed have greater variance than the market at large. The market return and SMB variables are both statistically significant at the 5% level, since their p-values are less than the critical value of 0.05. We also observe that the positive coefficient on MktRET12 indicates that the high P/E portfolio is positively related to the market portfolio, which explains why both portfolios' values moved in tandem with each other, but the high P/E portfolio had much greater amplitude in it's movements due to it's higher variance (risk).

The R^2 value of the regression above is 0.339 (Adjusted R^2 is 0.301). This means that the model only explains 33.9% of the variation in our portfolio returns, and that it could be improved. We most likely have some omitted variables whose effect is being shown in the MktRET12 and SMB12 variables, such as the other Fama-French Factors.

Limitations and Conclusion

The first limitation of our strategy is that it does not account for size of the company. It is possible that we could have generated even higher returns still had we filtered the stocks for both P/E ratio and their market capitalization. Small cap stocks with high P/E ratios may be the

most likely to provide the highest returns, as they have sizeable scope to grow further. Additionally, because we did not control for market capitalization, our portfolio may have included some large market-cap stocks whose growth is much slower than it's smaller counterparts.

Another limitation is that the industry of the stocks chosen were not controlled. High P/E stocks tend to come from fast growing industries such as technology. Selecting for high P/E stocks may have overweighted our portfolio in technology stocks or other similar industries with high P/E ratios, and greatly reduced our level of diversification. This may help to explain why the portfolio grew in value so dramatically, but also had such drastic changes in value when faced with recessions. However, past performance does not indicate future performance. In the current global economy, there is a plethora of companies whose services encourage a stay-at-home economy, e.g. Netflix for streaming, Amazon for e-commerce, Spotify for music, Facebook for socializing, Salesforce and other enterprise software companies to facilitate working from home, and so on. It is possible that our high P/E portfolio, if it is overweighted in tech, may not respond to future recessions as greatly as it did in the previous decade during the dotcom crisis and the Great Recession.

Another factor left out of our strategy was the effect of momentum. Momentum is the phenomenon where stocks that performed well over one year continue to do so over the next. Given that our portfolio was structured such that it was rebalanced every year, and the stocks from the previous year were replaced by a new basket of high P/E stocks, we never held a stock for longer than a year (unless it's P/E ratios were consistently amongst the highest in back-to-back years), which means our portfolio does not capture the returns that may be generated by the momentum effect. This is not to say that a portfolio that managed to capture the momentum effect would necessarily perform better, but just that this model did not consider it, and may have missed out on potential returns.

In sum, we managed to show that our portfolio successfully would have outperformed the broader market from 1980 to 2018. Specifically, during times of economic growth (when the market portfolio was increasing in value, i.e. stock market performing well), our portfolio outperformed the market, but in times of economic distress, the high P/E portfolio underperformed the benchmark in percentage terms (shown by greater amplitude of downward volatility). Despite carrying more volatility compared to the benchmark, the investor in that portfolio would have been compensated appropriately in returns for the amount of risk they took, shown by the Sharpe Ratios. The high volatility was corroborated by the regression, where the beta of the portfolio was shown to be positive and above 1, indicating that the portfolio returns have greater variance than the market as a whole. We can therefore say that the choice to invest in this portfolio would be in great part dependent on the investor's individual risk appetite. If the strategy were to be amended, it would filter stocks for their market cap and industry as well to provide diversification benefits (e.g. taking the 3 stocks with the highest P/E ratios in each industry for each year), and also hold each stock for longer than a year to capture the effect of momentum.