PSS-BEM114PS1 (1)

April 18, 2024

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

factors = pd.read_csv('/content/F-F_Research_Data_Factors.CSV')
factors['Date'] = pd.to_datetime(factors['Date'], format='%Y%m')
factors.set_index('Date', inplace=True)
factors['Total-Returns'] = factors['Mkt-RF'] + factors['RF']
avg_monthly_return = factors['Total-Returns'].mean()
volatility = factors['Total-Returns'].std()
sharpe_ratio = factors['Mkt-RF'].mean() / volatility

strategies = pd.read_csv('/content/ps1_strategies.csv')
strategies['date'] = pd.to_datetime(strategies['date'], format='%Y%m')
strategies.set_index('date', inplace=True)

joined = strategies.join(factors, how='inner')
CA_avg_monthly_return = (joined['CA'] + joined['RF']).mean()
CA_volatility = (joined['CA'] + joined['RF']).std()
CA_sharpe_ratio = joined['CA'].mean() / CA_volatility
```

1A) 0.9498208191126281 5.331034595353004 0.12790229056673896 **1B)** 0.9424588948216268 2.6183379914062206 0.2803191307780822

[ ]:
```
print(avg_monthly_return, volatility, sharpe_ratio)
print(CA_avg_monthly_return, CA_volatility, CA_sharpe_ratio)
```

```
0.9498208191126281 5.331034595353004 0.12790229056673896
0.9424588948216268 2.6183379914062206 0.2803191307780822
```

C) Code for part C below:

[ ]:
```
def CAPM_estimator(factor_data, strategy):
  model = sm.OLS(factor_data[strategy], sm.add_constant(factor_data['Mkt-RF']))
  results = model.fit()
  conf_interval = results.conf_int(0.05)
  print("Alpha confidence interval: ")
```

```
    print(conf_interval)
    return np.array([results.params[1], results.params[0]])
```

```
[ ]: output = CAPM_estimator(joined, 'CA')
     print(output)
```

```
Alpha confidence interval:
                0         1
const    0.250935  0.545057
Mkt-RF   0.456086  0.521355
[0.48872063 0.39799634]
```

D) Beta and Alpha values for the CA strategy are: (0.48872063, 0.39799634)

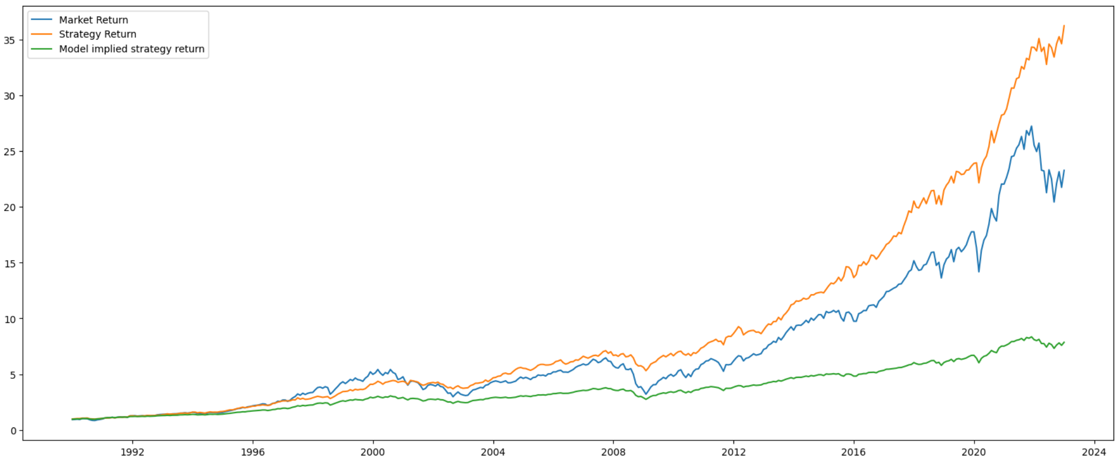
E) CAPM implied returns calculated and printed and cumulative returns calculated below:

```
[ ]: joined['CA_implied'] = joined['RF'] + output[0] * joined['Mkt-RF']
     print(joined['CA_implied'])

     joined['CA_percent'] = [(joined['RF'][i] + joined['CA'][i] + 100) / 100 for i␣
      ↪in range(len(joined['CA']))]
     joined['CA_cumulative'] = [np.prod(joined['CA_percent'][0:i+1]) for i in␣
      ↪range(len(joined['CA_percent']))]

     joined['CA_implied_percent'] = [(i + 100) / 100 for i in joined['CA_implied']]
     joined['CA_implied_cumulative'] = [np.prod(joined['CA_implied_percent'][0:i+1])␣
      ↪for i in range(len(joined['CA_implied_percent']))]

     joined['Total_percent'] = [(i + 100) / 100 for i in joined['Total-Returns']]
     joined['Total_cumulative'] = [np.prod(joined['Total_percent'][0:i+1]) for i in␣
      ↪range(len(joined['Total_percent']))]
```

```
1990-01-01   -3.266457
1990-02-01    1.112480
1990-03-01    1.534359
1990-04-01   -0.952101
1990-05-01    4.795028
                ...
2022-09-01   -4.379538
2022-10-01    4.056683
2022-11-01    2.538115
2022-12-01   -2.802699
2023-01-01    3.599992
Name: CA_implied, Length: 397, dtype: float64
```

f) Plot of Market, Strategy, and Model-implied strategy cumulative returns below:

```
[ ]: plt.figure(figsize = (20, 8))
     plt.plot(joined.index, joined['Total_cumulative'], label = 'Market Return')
     plt.plot(joined.index, joined['CA_cumulative'], label = 'Strategy Return')
     plt.plot(joined.index, joined['CA_implied_cumulative'], label = 'Model implied␣
     ↪strategy return')
     plt.legend()

     plt.show()
```



G) Yes, we believe the constant alpha strategy would make a good hedge fund strategy as its cumulative returns over the historical data are greater than the returns of the market. This is more impressive when considering that the market beta is ∼.5. This means that the strategy outperforms the market considering despite its low market exposure. The low beta value reduces downside risk of the CA strategy, and the alpha value ∼.4 ensures that our returns are still substantial.

0.1 Generalize functions for all strategies (Question 2)

```
[ ]: def calculate_stats(data, strategy):
       avg_return = (data[strategy] + data['RF']).mean()
       volatility = (data[strategy] + data['RF']).std()
       sharpe_ratio = data[strategy].mean() / volatility

       print("Average return: {:.3f} \t Volatility: {:.3f}\t Sharpe ratio: {:.3f}".
      ↪format(avg_return, volatility, sharpe_ratio))

     def calculate_percentage_returns(data, strategy):
       beta, alpha = CAPM_estimator(data, strategy)
       print("Beta: ", beta, "Alpha: ", alpha)
       data[strategy + '_implied'] = data['RF'] + beta * data['Mkt-RF']
       data[strategy + '_implied_percent'] = [(100 + i) / 100 for i in data[strategy␣
      ↪+ '_implied']]
```

```
  data[strategy + '_implied_cumulative'] = [np.prod(data[strategy +␣
↪'_implied_percent'][0:i+1]) for i in range(len(data[strategy +␣
↪'_implied_percent']))]

  data[strategy + '_percent'] = [(data[strategy][i] + joined['RF'][i] + 100) /␣
↪100 for i in range(len(data[strategy]))]
  data[strategy + '_cumulative'] = [np.prod(data[strategy + '_percent'][0:i+1])␣
↪for i in range(len(data[strategy + '_percent']))]

def compare_returns(data, strategy):
  plt.figure(figsize = (20, 8))
  plt.plot(joined.index, joined['Total_cumulative'], label = 'Market Return')
  plt.plot(joined.index, joined[strategy + '_cumulative'], label = '{} Strategy␣
↪Return'.format(strategy))
  plt.plot(joined.index, joined[strategy + '_implied_cumulative'],  label =␣
↪'Model implied {} strategy return'.format(strategy))
  plt.legend()
  plt.show()
```

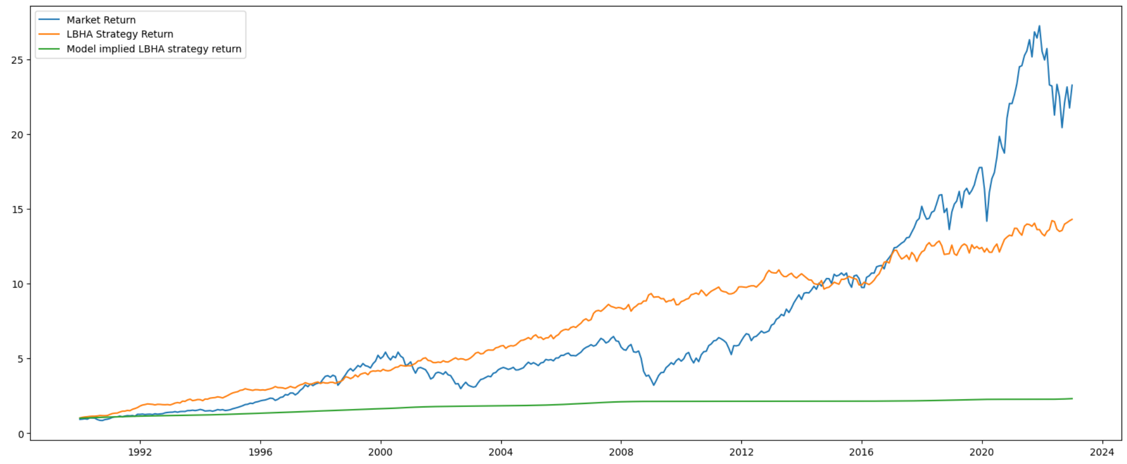
```
[ ]: calculate_stats(joined, 'Mkt-RF')
```

```
Average return: 0.896    Volatility: 4.453       Sharpe ratio: 0.154
```

```
[ ]: calculate_percentage_returns(joined, 'LBHA')
     calculate_stats(joined, 'LBHA')
```

```
Alpha confidence interval:
               0         1
const   0.274364  0.691330
Mkt-RF -0.041756  0.050774
Beta:  0.004508928165361608 Alpha:  0.4828467313945996
Average return: 0.694    Volatility: 2.110       Sharpe ratio: 0.230
```

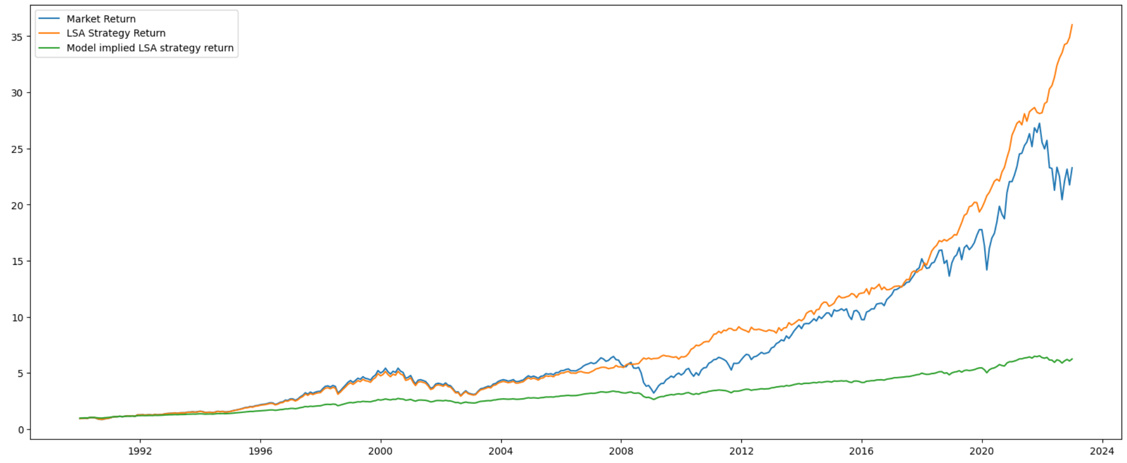
```
[ ]: compare_returns(joined, 'LBHA')
```



2A) In terms of total net return over time the LBHA strategy does not beat the market as seen by the graph above. However as calculated above as well, the sharpe ratio of the LBHA strategy is higher, so while it does not beat the market, there is a promising signal in the stability of its returns, and over time we do see there are periods where the strategy does outperform the market, it just currently does not have the same level of returns (it could outperform in the future).

```
[ ]: calculate_percentage_returns(joined, 'LSA')
     calculate_stats(joined, 'LSA')
     compare_returns(joined, 'LSA')
```

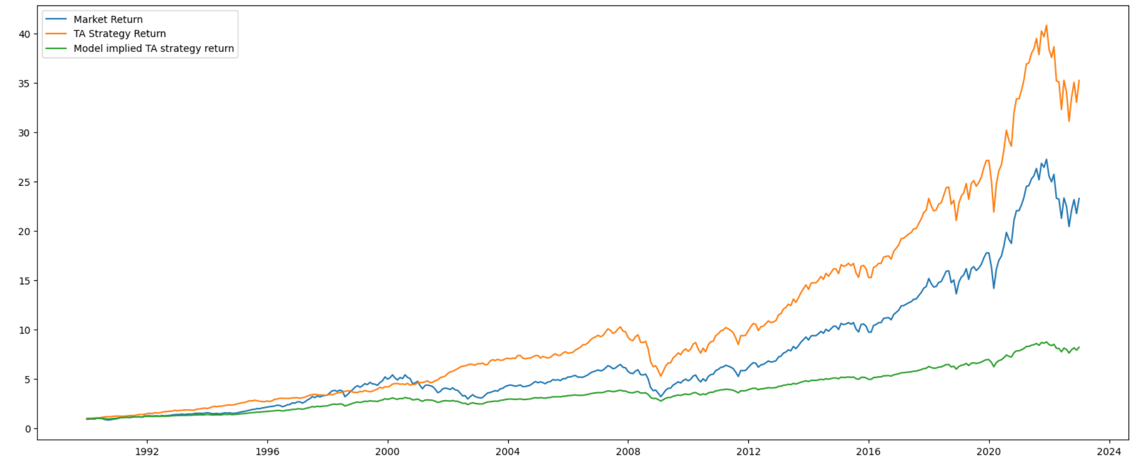
```
Alpha confidence interval:
              0         1
const   0.214679  0.744208
Mkt-RF  0.332810  0.450319
Beta:  0.3915644928792887 Alpha:  0.47944385516804483
Average return: 0.957    Volatility: 3.167      Sharpe ratio: 0.236
```



2B) The 95% confidence interval on the alpha constant for the LSA strategy is given by [0.214679 0.744208] which suggests that the alpha value is positive and signficiant at the 5% level. Some concerns the client might have over this strategy is that historically it had a strong correlation with the early 2000's financial crisis (could indicate possible high correlation with tech industry given .com bubble). These concerns can be allayed by noticing the lack of correlation during 2008 and 2022, as the strategy continues to go up. Additionally, the higher sharpe ratio of the strategy compared to the market suggesting the stability of the strategies returns are better leading to the strategy outperforming the market later overtime, as its consistent returns cumulatively are better than the higher-on-average but more volatile market returns.

```
[ ]: calculate_percentage_returns(joined, 'TA')
     calculate_stats(joined, 'TA')
     compare_returns(joined, 'TA')
```

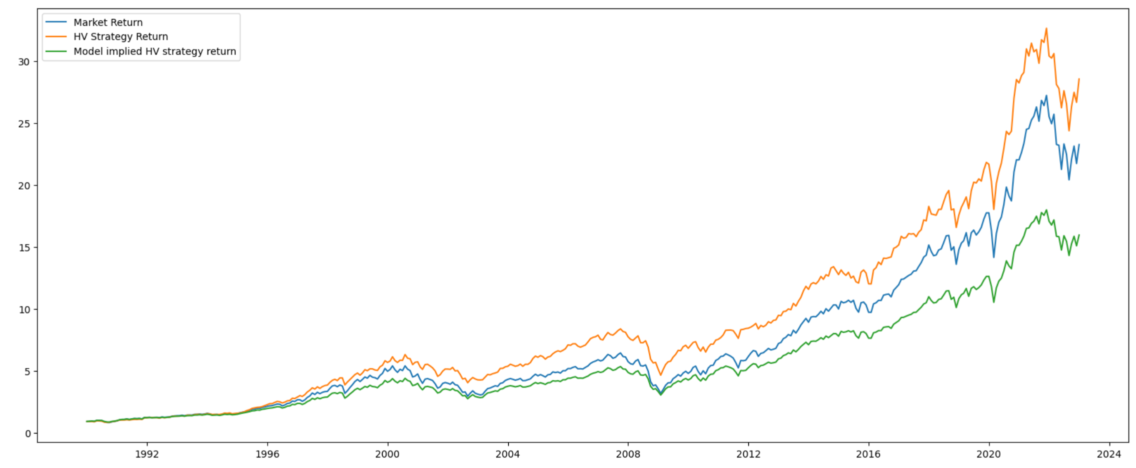
```
Alpha confidence interval:
               0         1
const   0.142531  0.664814
Mkt-RF  0.449817  0.565718
Beta:  0.5077675637624897 Alpha:  0.40367260082652157
Average return: 0.961    Volatility: 3.456       Sharpe ratio: 0.218
```



2C) The 95% confidence interval on the alpha constant for the LSA strategy is given by [0.142531 0.664814] which suggests that the alpha value is positive and signficiant at the 5% level. Investors may be interested in this strategy for the higher sharpe ratio it has versus the market, which would suggest to them that this strategy will bring in more stable returns compared to the market.

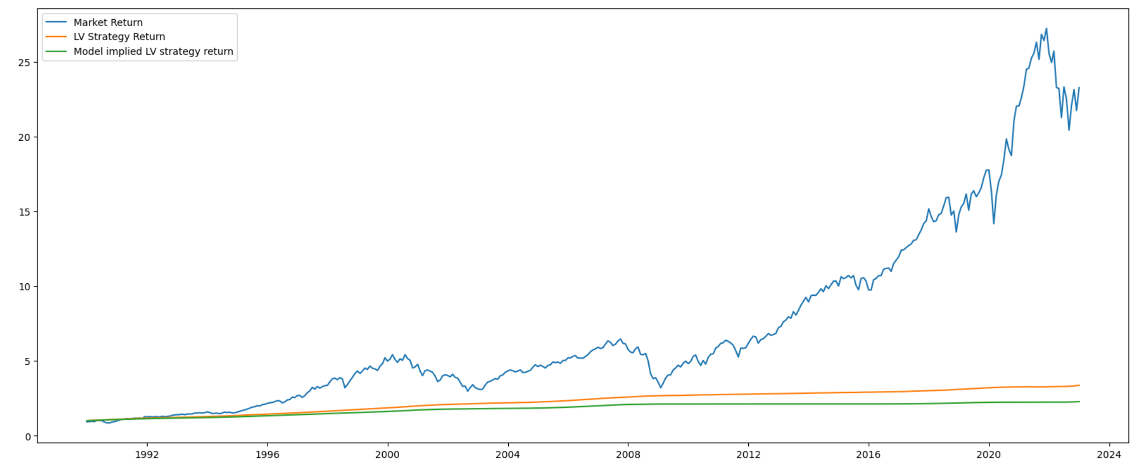
[ ]:
```
calculate_percentage_returns(joined, 'HV')
calculate_stats(joined, 'HV')
compare_returns(joined, 'HV')
```

```
Alpha confidence interval:
               0         1
const   0.027988  0.283488
Mkt-RF  0.782669  0.839367
Beta:  0.8110183123957878 Alpha:  0.15573809771489316
Average return: 0.922    Volatility: 3.835       Sharpe ratio: 0.186
```



```
[ ]: calculate_percentage_returns(joined, 'LV')
     calculate_stats(joined, 'LV')
     compare_returns(joined, 'LV')
```

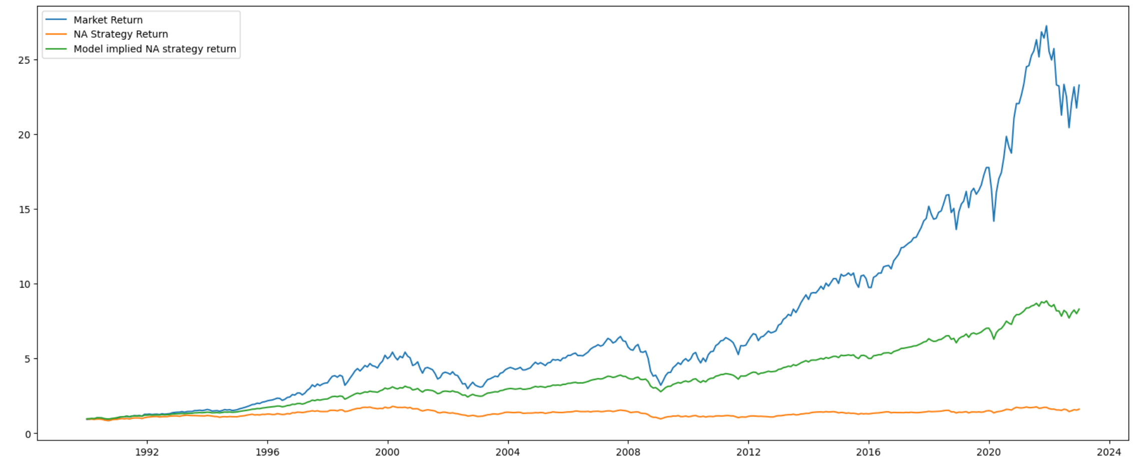
```
Alpha confidence interval:
               0         1
const   0.089254  0.108125
Mkt-RF -0.002722  0.001465
Beta:  -0.0006285054365238572 Alpha:  0.09868942537270566
Average return: 0.307    Volatility: 0.215       Sharpe ratio: 0.458
```



2D) We would prefer the high volatility (HV) strategy as even though the sharpe ratio of the Low Volatility strategy is much higher, the net returns are far too small over the period of historical data tested for, including a very low alpha value. Thus, while the HV strategy isn't necessarily the best option, it is better than the LV strategy which only returns ~150% on over 30 years due to both its low beta and low alpha.

```
[ ]: calculate_percentage_returns(joined, 'NA')
     calculate_stats(joined, 'NA')
     compare_returns(joined, 'NA')
```

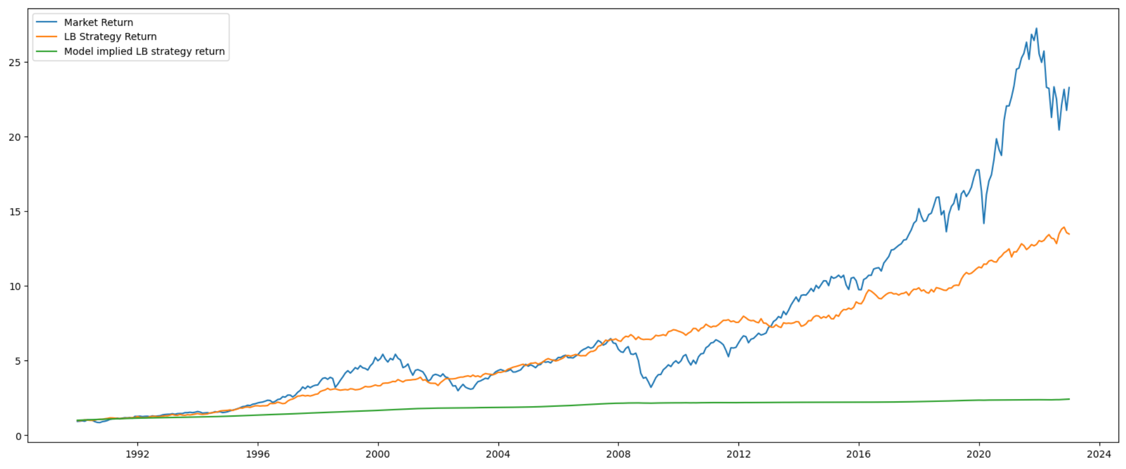
```
Alpha confidence interval:
               0         1
const  -0.547923 -0.261357
Mkt-RF  0.480180  0.543772
Beta:  0.5119756458422365 Alpha:  -0.40463995104401723
Average return: 0.156    Volatility: 2.691      Sharpe ratio: -0.020
```



2E) The interval produced by our tests for the alpha constant of the Negative Alpha strategy was: [-0.547923 -0.261357]. Additionally the returns were negative as well, resulting in a negative sharpe ratio and a loss of money over time / insignificant gains (returns appear to be below inflation rate so effectively losing money). This provides convincing evidence that this is not a good strategy to follow or invest in, which basically makes the efforts of this strategy useless.

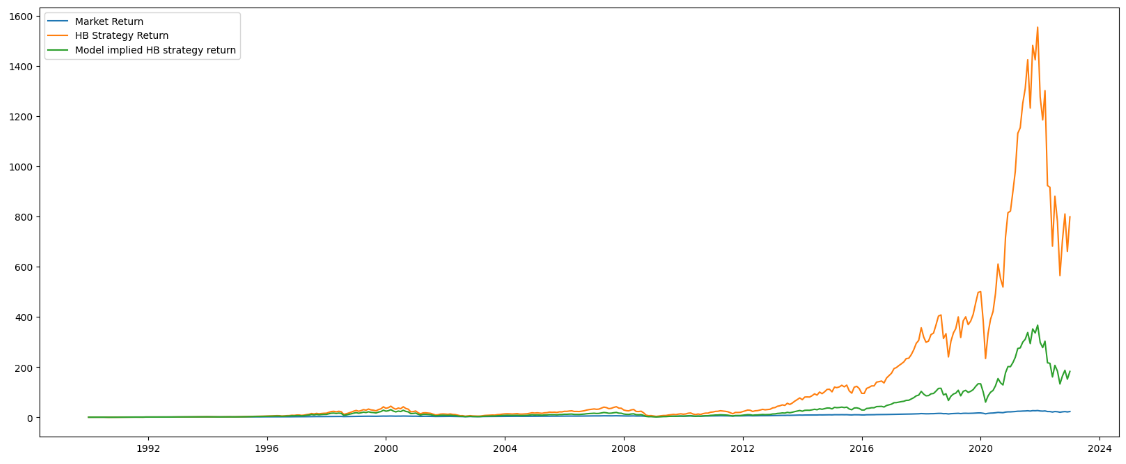
```
[ ]: calculate_percentage_returns(joined, 'LB')
     calculate_stats(joined, 'LB')
     compare_returns(joined, 'LB')
```

```
Alpha confidence interval:
               0         1
const   0.260559  0.645326
Mkt-RF -0.021785  0.063599
Beta:  0.020906636547654272 Alpha:  0.45294259298116174
Average return: 0.676    Volatility: 1.941      Sharpe ratio: 0.241
```



```
[ ]: calculate_percentage_returns(joined, 'HB')
     calculate_stats(joined, 'HB')
     compare_returns(joined, 'HB')
```

```
Alpha confidence interval:
               0         1
const   0.305877  0.438987
Mkt-RF  2.983100  3.012639
Beta:  2.9978695765385552 Alpha:  0.37243191058667485
Average return: 2.642    Volatility: 13.376      Sharpe ratio: 0.182
```



2F) We believe the low beta strategy is better compared to the high beta strategy, evaluated based on the sharpe ratio and our observations of the cumulative return plots. As seen by the calculated sharpe ratios, the low beta strategy has one of the higher sharpe ratios compared to all other strategies tested while also maintaining decent return, which are consistently positive as seen by the graph. On the other hand, the high beta strategy, while it does have high gains, has a low sharpe ratio due to extreme volatility which is also seen in some of the massive losses the strategy has taken over time (2021-). Thus, we conclude that the low beta strategy is a better hedge fund strategy as it is less correlated with the market and can produce high uncorrelated returns- a good sign for a hedge fund, basically meaning that the fund has higher alpha (this higher alpha value is estimated and proven in the next question/ part of the code).

[ ]:
```
hb_coef = CAPM_estimator(joined, 'HB')
lb_coef = CAPM_estimator(joined, 'LB')

print("HB")
print("Beta: {:.3f} \t Alpha: {:.3f}".format(hb_coef[0], hb_coef[1]))
print("LB")
print("Beta: {:.3f} \t Alpha: {:.3f}".format(lb_coef[0], lb_coef[1]))
```

```
Alpha confidence interval:
               0         1
const   0.305877  0.438987
Mkt-RF  2.983100  3.012639
Alpha confidence interval:
               0         1
const   0.260559  0.645326
Mkt-RF -0.021785  0.063599
HB
Beta: 2.998      Alpha: 0.372
LB
Beta: 0.021      Alpha: 0.453
```

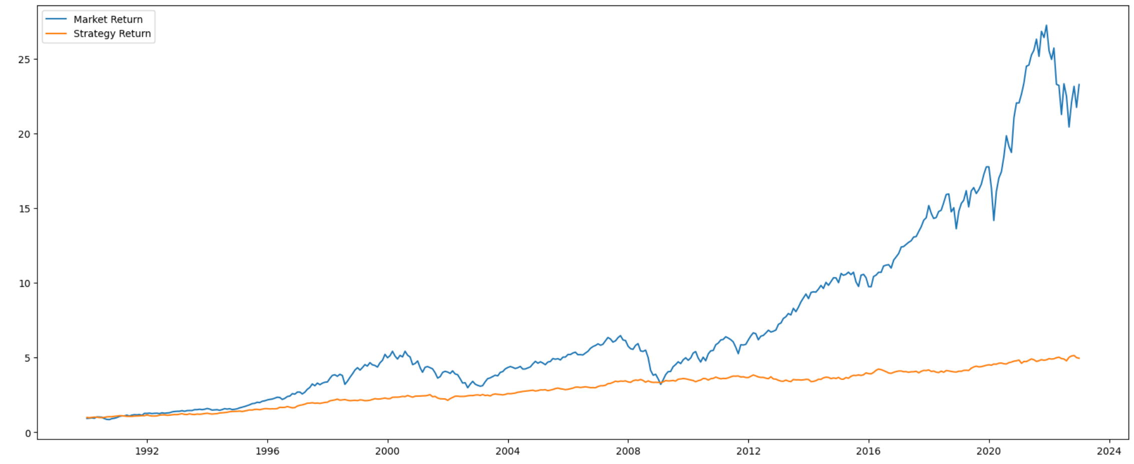
3A) LB Strategy Beta: 0.021 Alpha: 0.453 HB Strategy Beta: 2.998 Alpha: 0.372

[ ]:
```
def calculate_fee_returns(data, strategy, funding):
  fund_val = funding
  percent_list = []
  fee_list = []
  total_fees = 0
  products = []

  fee_tracker = []

  curr_max = fund_val

  management = .0015

  for i in range(len(data[strategy])):
    prev_val = fund_val
    total_fees += .0015 * fund_val
    fund_val -= .0015 * fund_val
    change = (100 + data[strategy][i] + data['RF'][i]) / 100
    fund_val *= change
    total_fees *= change
    if fund_val > curr_max:
      old_max = curr_max
      curr_max = fund_val
      fund_val -= (curr_max - old_max) * .2
      total_fees +=  (curr_max - old_max) * .2
      curr_max = fund_val

    fee_tracker.append(total_fees)

    percent_list.append(((fund_val / prev_val) - 1) * 100)
    products.append(fund_val / prev_val)
    # fee_list.append(fees)

  data[strategy + "fee_return"] = percent_list
  print(strategy)
  print(total_fees)

  output = CAPM_estimator(data, strategy + 'fee_return')

  print("Beta: {:.3f} \t Alpha: {:.3f}".format(output[0], output[1]))

  return fee_tracker, products
```

[ ]:
```
fee_tracker, returns = calculate_fee_returns(joined, 'LB', 100000000)
```

```
LB
851464397.7777592
Alpha confidence interval:
               0         1
const   0.228111  0.591159
Mkt-RF -0.024743  0.055821
Beta: 0.016      Alpha: 0.410
```

[ ]:
```
cumulative = [np.prod(returns[0:i+1]) for i in range(0, len(returns))]

plt.figure(figsize = (20, 8))
plt.plot(joined.index, joined['Total_cumulative'], label = 'Market Return')
plt.plot(joined.index, cumulative, label = 'Strategy Return')
plt.legend()
plt.show()
```



```
[ ]: returns = calculate_fee_returns(joined, 'HB', 100000000)
```

```
HB
67891525040.86178
Alpha confidence interval:
               0         1
const   0.008532  0.214906
Mkt-RF  2.900525  2.946321
Beta: 2.923      Alpha: 0.112
```

3B) LB Strategy Beta: 0.016 Alpha: 0.410 HB Strategy Beta: 2.923 Alpha: 0.112

3C) LB fees: 851464397.7777592 HB fees: 67891525040.86178 **3D)** We can see from the data visualized on the market returns that over time the market does tend to do very well. This idea has been discussed in class such as in the portfolio theory developed by Markowitz where often replicating the market along the efficient frontier can be a very strong investment. Thus, as a hedgefund it is not that bad of an idea to just leverage the returns of the market, since the market itself does fairly well over time and thus a leveraged position on the market portfolio (which is a high beta strategy) will also generate strong returns, which we can see in visualization of the high beta (HB) portfolio versus market returns plot. Considering the fees analysis done above as well, we also see the high beta strategy yielding higher return as well as higher fees, which is good for the hedge fund in that it is appealing in its returns to investors while also making more money through management fees as well, as calculated above. In conclusion, we see that hedge funds with high betas can tend to be successful given the above market performance it has (market generally does well so the leveraged high beta position does even better) and the increased management fees it gains from it.