ps1

April 18, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
```

1 Problem 1

1.1 (a)

[ ]:
```
ff = pd.read_csv('data/F-F_Research_Data_Factors.csv')
avg_monthly_ff = ff['Mkt-RF'].mean() # unsure if this means relative returns or␣
 ↪average rate of return
volatility_ff = ff['Mkt-RF'].std()
sharpe_ratio_ff = (ff['Mkt-RF']/volatility_ff).mean()
print("The average monthly return of the market is ")
print(avg_monthly_ff)
print("the volatility is")
print(volatility_ff)
print("The average monthly Sharpe Ratio is")
print(sharpe_ratio_ff)
```

```
The average monthly return of the market is
0.6701034482758605
the volatility is
5.352554226265553
The average monthly Sharpe Ratio is
0.12519321055872606
```

1.2 (b)

[ ]:
```
sdf = pd.read_csv('data/ps1_strategies.csv')
avg_monthly_sdf = sdf.CA.mean()
volatility_sdf = sdf.CA.std() # Not sure if this works for excess returns or if␣
 ↪I need to subtract the risk free rate
sharpe_ratio_sdf = (sdf.CA/volatility_sdf).mean()
print("The average monthly return of the CA is")
```

```
print(avg_monthly_sdf)
print("the volatility is")
print(volatility_sdf)
print("The average monthly Sharpe Ratio is")
print(sharpe_ratio_sdf)
```

```
The average monthly return of the CA is
0.7339702298342209
the volatility is
2.6293585544149796
The average monthly Sharpe Ratio is
0.27914421507930354
```

1.3 (c)

[ ]:
```
def generate_capm(strategy_return, market_return, risk_free_rate):
    # Prepare data for regression
    x = market_return - risk_free_rate  # Independent variable (market excess return)
    y = strategy_return  # Dependent variable (strategy excess return)
    x = sm.add_constant(x)  # Add a constant term to the independent variable

    # Perform ordinary least squares (OLS) regression
    model = sm.OLS(y, x).fit()

    # Extract alpha and beta from the model's parameters
    alpha = model.params[0]
    beta = model.params[1]

    # Extract p-value for significance of alpha
    alpha_p_value = model.pvalues[0]

    return alpha, beta, alpha_p_value

def generate_cumulative_returns(df):
    return (((df/100)+1).cumprod() - 1)*100

def test_alpha(alpha, p):
    if alpha > 0 and p < 0.05:
        return True, alpha, p
    else:
        return False, alpha, p
```

1.4 (d)

[ ]:
```
ff_trim = ff[ff['Month'] >= sdf['date'].min()]
ff_trim = ff_trim[ff_trim['Month'] <= sdf['date'].max()]
ff_trim = ff_trim.reset_index(drop=True)
```

[ ]:
```
alpha, beta, p = generate_capm(sdf.CA.to_numpy(), (ff_trim['Mkt-RF'] +␣
↪ff_trim['RF']).to_numpy(), ff_trim.RF.to_numpy())
print("The beta of the CA is")
print(beta)
print("The alpha of the CA is")
print(alpha)
print("The significance of the alpha is")
print(p)
```

```
The beta of the CA is
0.48871999679803935
The alpha of the CA is
0.39800908543585123
The significance of the alpha is
1.7352942016039679e-07
```

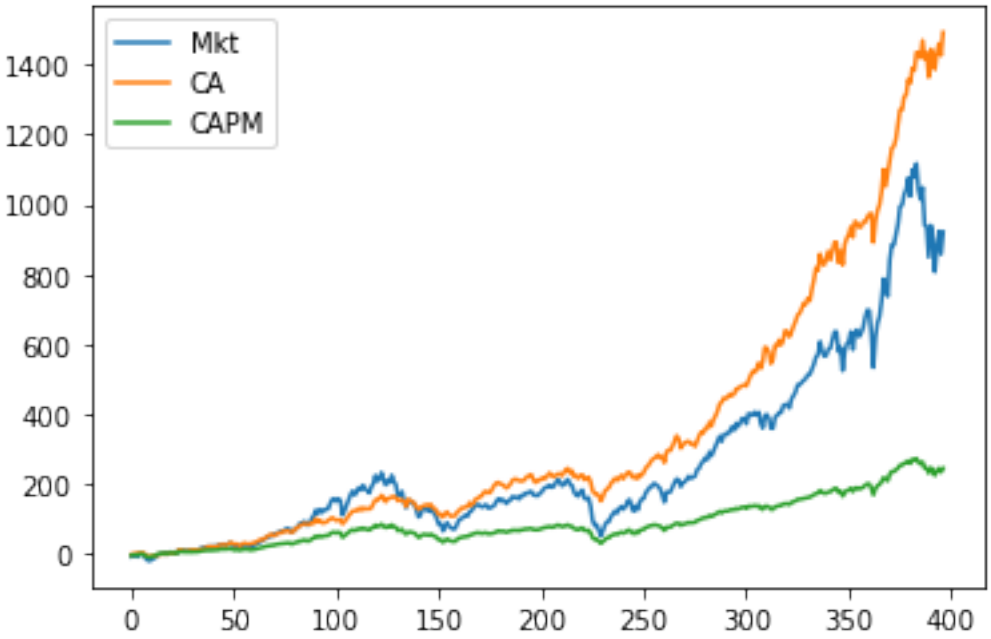
[ ]:
```
returns = ff_trim.RF + beta*ff_trim['Mkt-RF']
```

1.5 (f)

does he want excess or total returns?

[ ]:
```
#plot cumulative returns, market cumulative returns, and CA cumulative returns
plt.plot(generate_cumulative_returns(ff_trim['Mkt-RF']), label='Mkt')
plt.plot(generate_cumulative_returns(sdf.CA), label='CA')
plt.plot(generate_cumulative_returns(returns - ff_trim.RF), label='CAPM')
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x168c107f0>
```



1.6 (g)

This strategy seems to generate a good amount of alpha, so it would seem like a good strategy from that point of view. There could be some downside, though, as it seems to follow the market trends fairly closely and could lead to higher losses in a market downturn.

2 Problem 2

[ ]:
```
strats = pd.read_csv("data/ps1_strategies.csv")
strats
```

[ ]:

	date	CA	LBHA	LSA	TA	HV	LV	\
0	199001	-1.771984	1.498262	-7.457500	1.679061	-7.271919	0.022091	
1	199002	1.418966	3.642659	1.054500	0.205289	-0.986167	0.062055	
2	199003	1.375007	1.737180	1.738500	-1.572688	-0.018665	0.341639	
3	199004	-0.395588	0.734520	-3.192000	2.474704	-3.294381	0.253568	
4	199005	2.588010	1.298923	7.999000	0.754379	8.038877	-0.113650	
..	…	…	…	…	…	…	…	
392	202209	-2.640759	0.341477	1.206628	-8.882500	-8.396274	0.138919	
393	202210	3.290022	2.849030	1.965639	7.438500	7.728801	0.099102	
394	202211	1.615024	0.474610	0.054055	4.370000	4.132530	0.025099	
395	202212	-2.144610	0.503661	1.172292	-6.089500	-3.276646	0.290945	
396	202301	4.303679	0.332465	2.888904	6.317500	6.697050	0.243330	

NA	LB	HB

```
0   -5.392944 -1.353457 -22.772632
1   -1.768405 -2.118514   5.151408
2   -0.333926  1.452434   4.480134
3   -2.578905  2.123740 -10.101798
4    1.337511 -1.555230  26.259080
..        …       …         …
392 -8.208565  4.911723 -27.800254
393  3.743379  2.064744  24.367165
394  4.448278  0.730237  14.882494
395 -2.127425 -2.965015 -18.772379
396  3.496876 -1.065103  20.550014

[397 rows x 10 columns]
```

[ ]:
```python
def analyze_strategy(strat="CA", ff=pd.read_csv('data/F-F_Research_Data_Factors.csv')):
    '''
    strategy: dataframe with columns 'date' and 'CA'
    ff: dataframe with columns 'Month', 'Mkt-RF', and 'RF'
    '''
    # Trim ff to match strategy
    sdf = pd.read_csv('data/ps1_strategies.csv')
    strategy = sdf[strat]
    ff_trim = ff[ff['Month'] >= sdf['date'].min()]
    ff_trim = ff_trim[ff_trim['Month'] <= sdf['date'].max()]
    ff_trim = ff_trim.reset_index(drop=True)

    # Calculate strategy returns
    avg_monthly = strategy.mean()
    volatility = strategy.std()
    sharpe_ratio = (strategy/volatility).mean()

    # Calculate CAPM
    alpha, beta, p = generate_capm(strategy.to_numpy(), (ff_trim['Mkt-RF'] +␣
↪ff_trim['RF']).to_numpy(), ff_trim.RF.to_numpy())

    # Calculate cumulative returns
    returns = ff_trim.RF + beta*ff_trim['Mkt-RF']

    plt.figure()
    plt.plot(generate_cumulative_returns(ff_trim['Mkt-RF']), label='Mkt')
    plt.plot(generate_cumulative_returns(strategy), label=strat)
    plt.plot(generate_cumulative_returns(returns - ff_trim.RF), label='CAPM for␣
↪' + strat + ' without alpha')
    plt.legend()

    print("------- For strategy " + strat + ": -------")
```

```
    print("The average monthly return of the strategy is ")
    print(avg_monthly)
    print("the volatility is")
    print(volatility)
    print("The average monthly Sharpe Ratio is")
    print(sharpe_ratio)
    print("The beta of the strategy is")
    print(beta)
    print("The alpha of the strategy is")
    print(alpha)
    is_significant, alpha_value, p_value = test_alpha(alpha, p)

    if is_significant:
        print(f"Alpha is positive ({alpha_value}) and significant at the 5%␣
↪level (p-value: {p_value}).")
    else:
        print(f"Alpha is not positive and significant at the 5% level (alpha:␣
↪{alpha_value}, p-value: {p_value}).")

    print()

    return avg_monthly, volatility, sharpe_ratio, beta, alpha
```

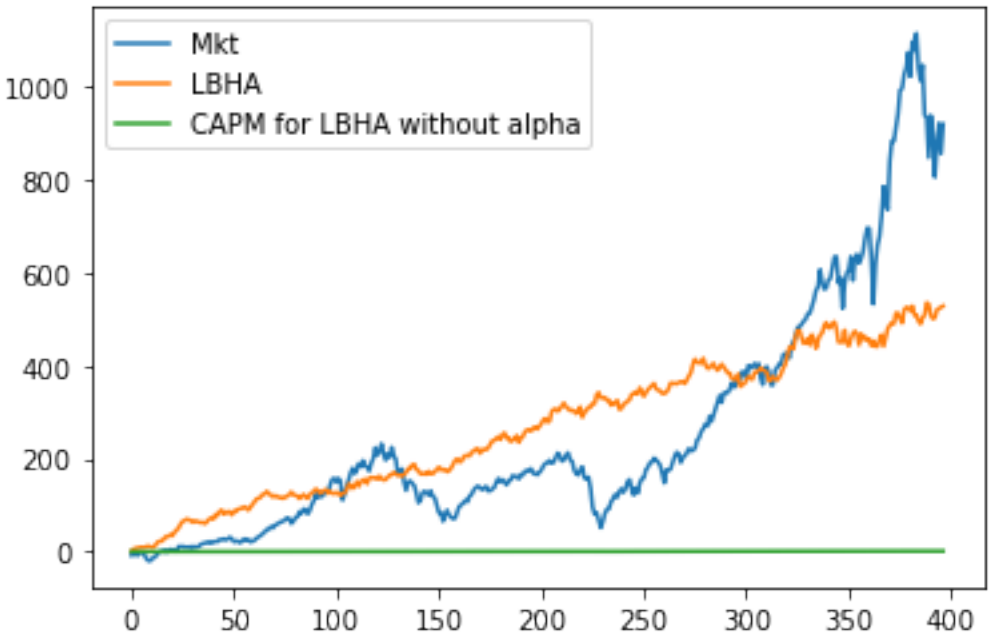
[ ]:
```
analyze_strategy("LBHA")
analyze_strategy("LSA")
analyze_strategy("TA")
analyze_strategy("HV")
analyze_strategy("LV")
analyze_strategy("NA")
analyze_strategy("HB")
analyze_strategy("LB")
```

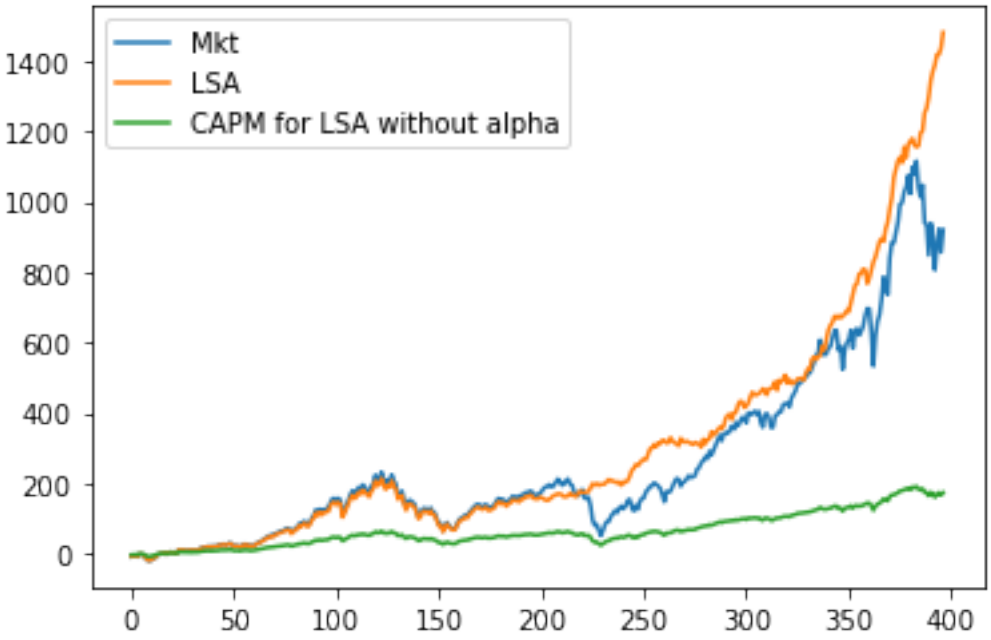
```
------- For strategy LBHA: -------
The average monthly return of the strategy is
0.48594642075200606
the volatility is
2.085658032439009
The average monthly Sharpe Ratio is
0.23299429398007837
The beta of the strategy is
0.004510732886909078
The alpha of the strategy is
0.4828456043485644
Alpha is positive (0.4828456043485644) and significant at the 5% level (p-value:
7.046348829725176e-06).

------- For strategy LSA: -------
```

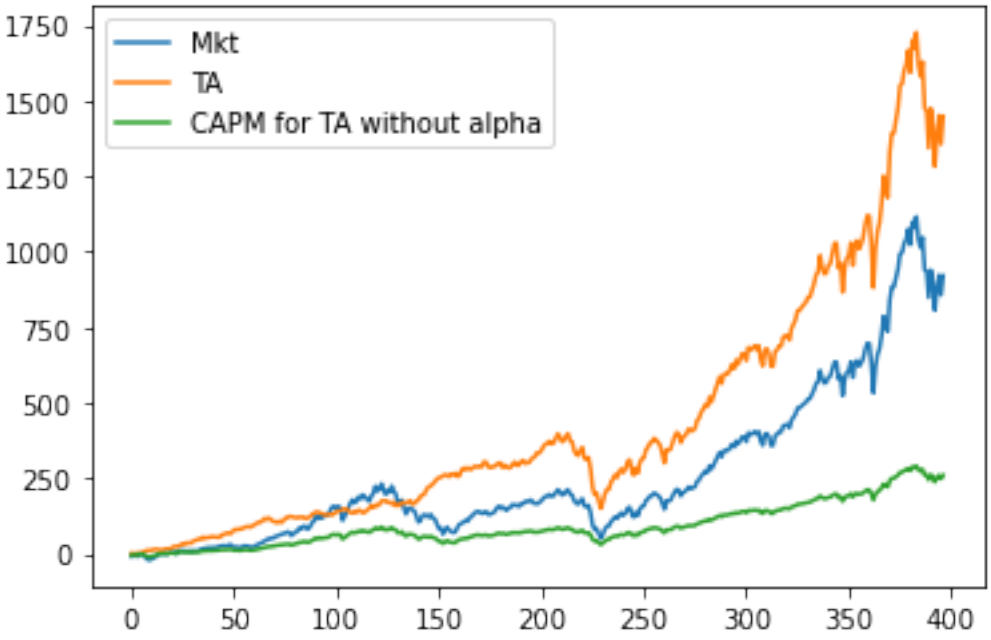
```
The average monthly return of the strategy is
0.748627183622996
the volatility is
3.1723253621783734
The average monthly Sharpe Ratio is
0.23598688600747064
The beta of the strategy is
0.3915663888474703
The alpha of the strategy is
0.4794524149067157
Alpha is positive (0.4794524149067157) and significant at the 5% level (p-value:
0.0004158094231576417).

------- For strategy TA: -------
The average monthly return of the strategy is
0.7527404182120594
the volatility is
3.4570217361430466
The average monthly Sharpe Ratio is
0.21774246032131758
The beta of the strategy is
0.5077677902172448
The alpha of the strategy is
0.40368523526951944
Alpha is positive (0.40368523526951944) and significant at the 5% level
(p-value: 0.0025305521758051588).

------- For strategy HV: -------
The average monthly return of the strategy is
0.7132774373094987
the volatility is
3.835627986175194
The average monthly Sharpe Ratio is
0.18596105771476631
The beta of the strategy is
0.8110207385778987
The alpha of the strategy is
0.1557568585556085
Alpha is positive (0.1557568585556085) and significant at the 5% level (p-value:
0.01699258675973323).

------- For strategy LV: -------
The average monthly return of the strategy is
0.09825735558999502
the volatility is
0.09443191030868535
The average monthly Sharpe Ratio is
1.0405100910148362
```

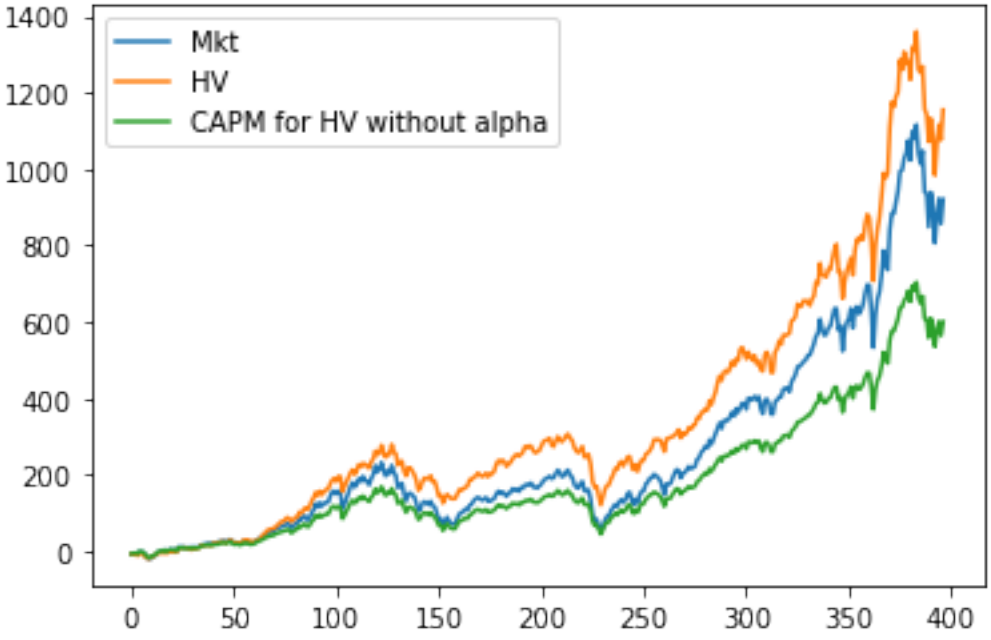
```
The beta of the strategy is
-0.0006284077416968671
The alpha of the strategy is
0.0986893423829081
Alpha is positive (0.0986893423829081) and significant at the 5% level (p-value:
2.108589302816927e-64).

------- For strategy NA: -------
The average monthly return of the strategy is
-0.05267926272345478
the volatility is
2.6956410409938267
The average monthly Sharpe Ratio is
-0.019542387848507148
The beta of the strategy is
0.5119778329867377
The alpha of the strategy is
-0.40462855844237305
Alpha is not positive and significant at the 5% level (alpha:
-0.40462855844237305, p-value: 5.192864818287161e-08).

------- For strategy HB: -------
The average monthly return of the strategy is
2.4333350965536824
the volatility is
13.384603427871243
The average monthly Sharpe Ratio is
0.18180106042489547
The beta of the strategy is
2.9978735923008775
The alpha of the strategy is
0.3725046631158165
Alpha is positive (0.3725046631158165) and significant at the 5% level (p-value:
9.6110729568853e-25).

------- For strategy LB: -------
The average monthly return of the strategy is
0.4673149840304968
the volatility is
1.9267646547507409
The average monthly Sharpe Ratio is
0.24253869453036617
The beta of the strategy is
0.020905486781044837
The alpha of the strategy is
0.452943909981592
Alpha is positive (0.452943909981592) and significant at the 5% level (p-value:
4.9991196047999876e-06).
```

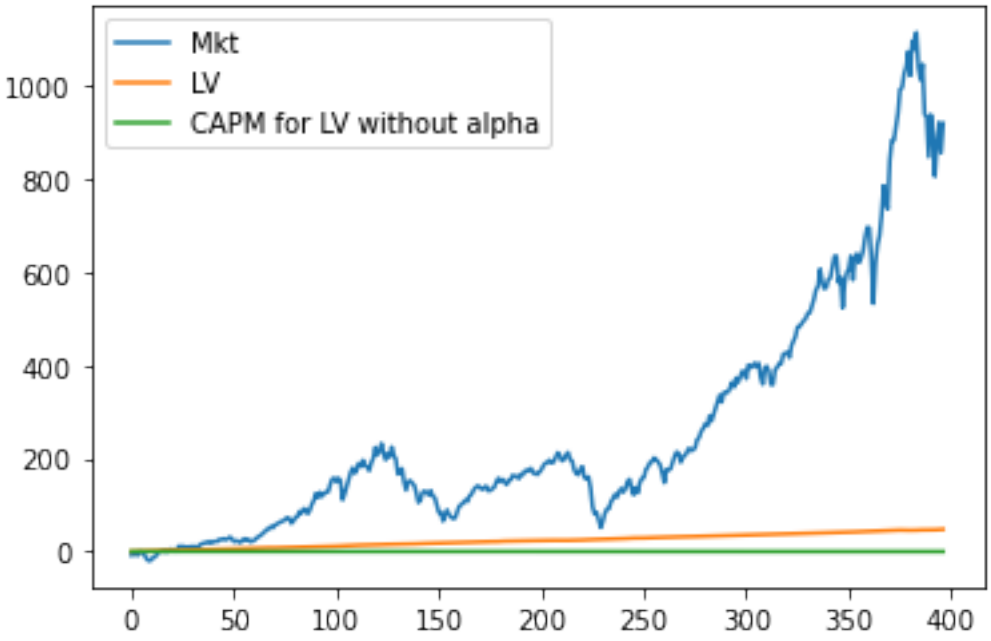
```
[ ]: (0.4673149840304968,
      1.9267646547507409,
      0.24253869453036617,
      0.020905486781044837,
      0.452943909981592)
```

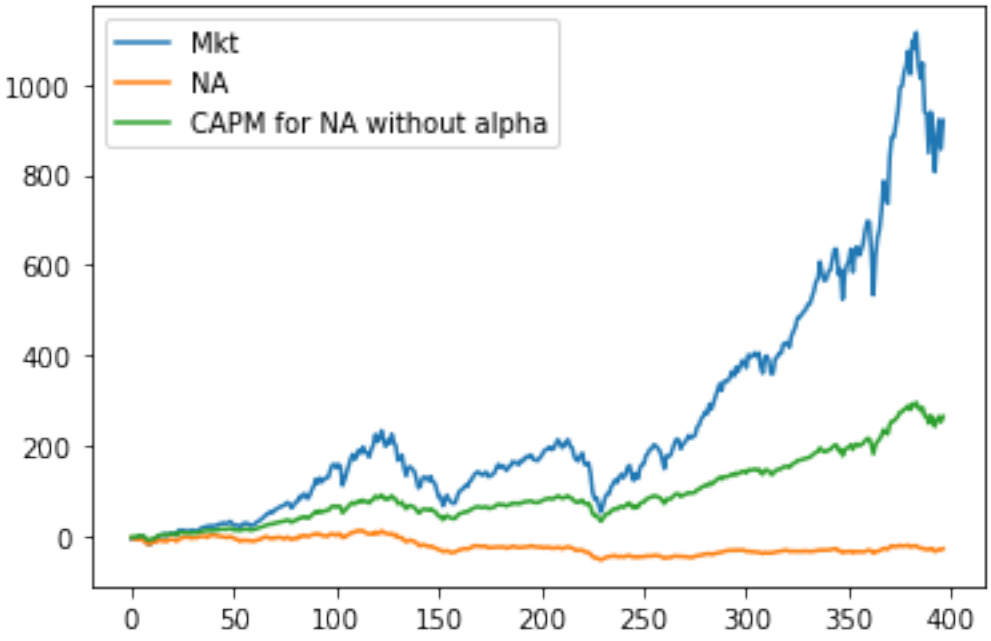


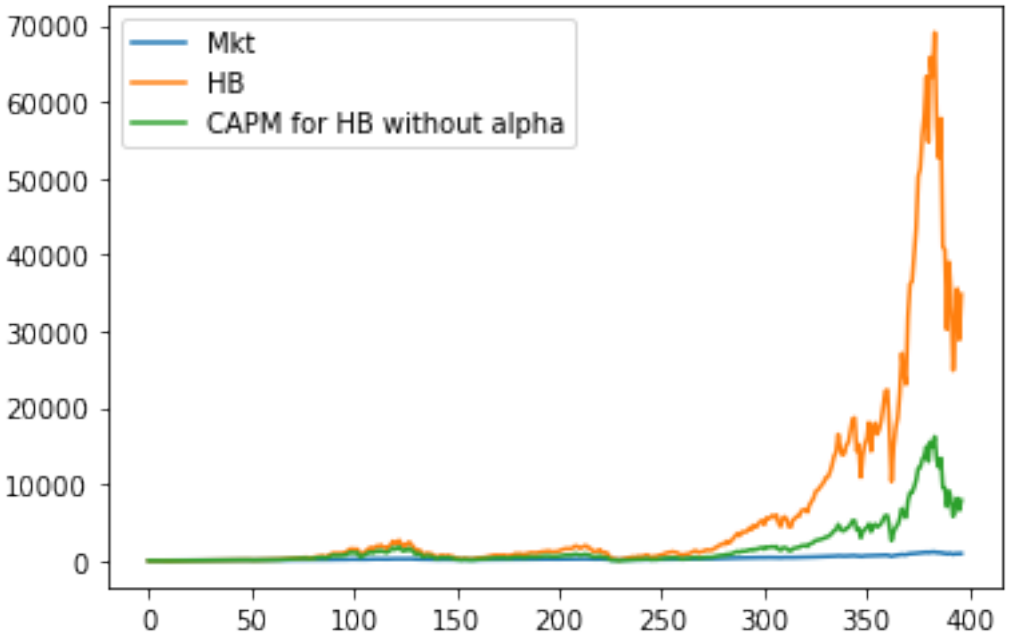


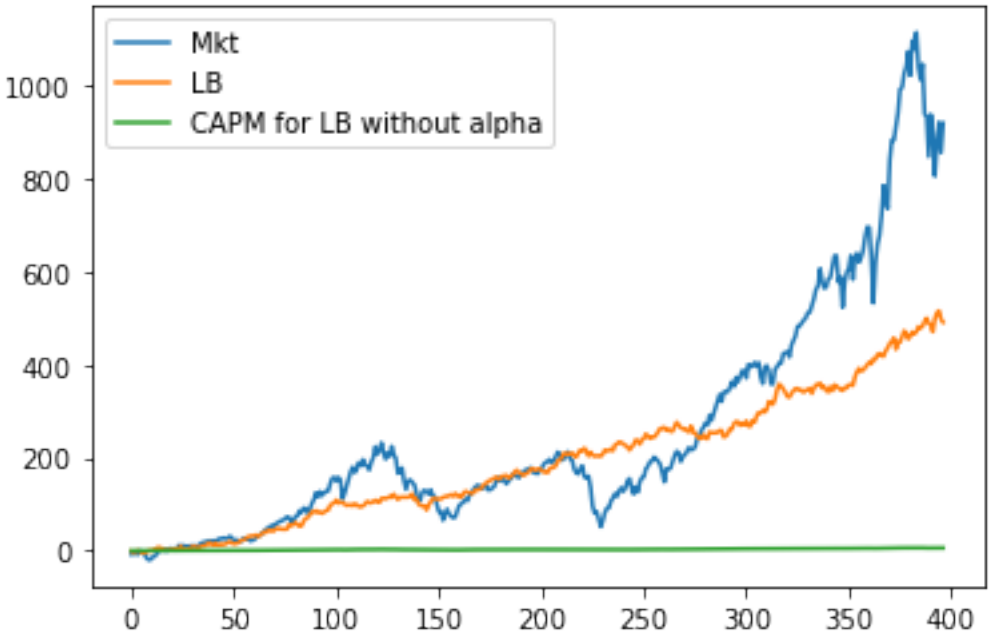

Mkt
TA
CAPM for TA without alpha
1750
1500
1250
1000
750
500
250
0
0
50
100
150
200
250
300
350
400


Mkt
HV
CAPM for HV without alpha
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400


Mkt
LV
CAPM for LV without alpha
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400


Mkt
NA
CAPM for NA without alpha
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400





2.1 A.

LBHA seems to beat the market during market downturns, while it loses out when the market is doing exceptionally well.

2.2 B.

We have positive alpha and it is significant. One of the issues that could come up is that investors could be wondering why the strategy did not generate alpha in the past. To allay concerns, we could show that if we calculate alpha after it starts, then p-values are much better and we actualyl have significant positive alpha. A possible reason for this is significant policy change, rather than an issue with the strategy.

2.3 C.

TA also has a positive significant alpha. It does seem like a strategy investors would be interested in given the amount of alpha generated. It does, however, have fairly high beta, meaning lots of market exposure, but that could be something people are OK with since it still beats the market.

2.4 D.

Both have significant and positive alpha. The HV strategy is much more correlated with the market given the high beta, and much more suceptible to downturns in the market, so a hedge fund investor likely would not be as interested in HV strategy vs the LV strategy, which generates slightly less alpha but is a much more safe investment. The LV strategy is also negatively coorelated with the market (negative beta), which means it is a good option if the investor is looking to diversify from the market portfolio.

2.5 E.

The strategy is not all for nothing, we could short the strategy instead, giving us a positive alpha! It could also give you insights for other strategies that could yield positive alpha instead. All information is good information.

2.6 F.

It likely depends on what you're looking for as an investor. Most hedge fund investors would look for more consistent returns even during downturn, which the LB strategy does far better than the HB one in that regard. Both have significant and high alpha, with the LB one being higher, with less volatility, so i'd say this one performs better.The LB strategy also allows the investor to diversify their portfolio away from the market portfolio, given the very low beta.

Problem 3 note: confirm that by "cumulative" he really means "current" value

[ ]:
```
np.seterr(all='raise')
```

[ ]:
```
{'divide': 'raise', 'over': 'raise', 'under': 'raise', 'invalid': 'raise'}
```

[ ]:
```
def calculate_after_fee_returns(before_fee_returns, management_fee_pct,␣
↪incentive_fee_pct, investment_amount):
    # Convert management and incentive fees to decimals
    management_fee = management_fee_pct / 100
    incentive_fee = incentive_fee_pct / 100

    # Initialize variables
```

```
    after_fee_returns = []

    total_fees = 0
    max_value = investment_amount
    current_value = investment_amount

    for ret in before_fee_returns:
        ret = ret/100
        # Calculate current value before fees
        if 1+ret < 0: print("wtf")
        prev_value = current_value
        current_value *= (1 + ret)

        # Calculate management fee and incentive fee
        management_fee_amount = current_value * management_fee
        incentive_fee_amount = max(0, current_value - max_value) * incentive_fee

        total_fees += management_fee_amount + incentive_fee_amount

        # Calculate after-fee value and return
        after_fee_value = current_value - management_fee_amount -␣
↪incentive_fee_amount
        after_fee_return = (after_fee_value / prev_value) - 1 if prev_value > 0␣
↪else 0
        # # If after-fee return is negative but before-fee return is positive,␣
↪print a warning
        # if after_fee_return < 0 and ret > 0:
        #     print(f"Negative after-fee return ({after_fee_return*100:.2f}%)␣
↪for before-fee return of {ret*100:.2f}%") if after_fee_return < 0 and ret >␣
↪0 else None
        #     print(f"Current value: {current_value}")
        #     print(f"Management fee amount: {management_fee_amount}")
        #     print(f"Incentive fee amount: {incentive_fee_amount}")
        #     print(f"After-fee value: {after_fee_value}")
        #     print(f"Max value: {max_value}")
        #     print(f"Before-fee return: {ret*100:.2f}%")
        #     print(f"After-fee return: {after_fee_return*100:.2f}%")
        #     print()
        after_fee_returns.append(after_fee_return*100)

        # Update max_value
        max_value = max(max_value, after_fee_value)

        # Update current_value for next iteration
        current_value = after_fee_value

    return after_fee_returns, total_fees
```

```
[ ]: def fee_analysis2(investment_amount, strat="CA", ff=pd.read_csv('data/
 ↪F-F_Research_Data_Factors.csv')):
        '''
        strategy: dataframe with columns 'date' and 'CA'
        ff: dataframe with columns 'Month', 'Mkt-RF', and 'RF'
        '''
        # Trim ff to match strategy
        sdf = pd.read_csv('data/ps1_strategies.csv')
        strategy = sdf[strat]
        ff_trim = ff[ff['Month'] >= sdf['date'].min()]
        ff_trim = ff_trim[ff_trim['Month'] <= sdf['date'].max()]
        ff_trim = ff_trim.reset_index(drop=True)

        # Calculate before-fee alphas and betas (part a)
        alpha_1, beta_1, p_1 = generate_capm(strategy.to_numpy(),␣
 ↪(ff_trim['Mkt-RF'] + ff_trim['RF']).to_numpy(), ff_trim.RF.to_numpy())
    # Calculate maximum return at each time point
        after_fee_returns, total_fees = calculate_after_fee_returns(strategy.
 ↪to_numpy(), 1.8/12, 20, investment_amount)



        # Calculate CAPM
        alpha_2, beta_2, p_1 = generate_capm(after_fee_returns, (ff_trim['Mkt-RF']␣
 ↪+ ff_trim['RF']).to_numpy(), ff_trim.RF.to_numpy())

        return alpha_1, beta_1, alpha_2, beta_2, total_fees
```

```
[ ]: def total_analysis(strats=["LB", "HB"], ff=pd.read_csv('data/
 ↪F-F_Research_Data_Factors.csv')):
        # bf = before fee
        # af = after fee

        bf_alpha = []
        bf_beta = []
        af_alpha = []
        af_beta = []
        tot_fees = []
        for strat in strats:
            alpha_1, beta_1, alpha_2, beta_2, tot_fee = fee_analysis2(100000000,␣
 ↪strat=strat, ff=ff)
            bf_alpha.append(alpha_1)
            bf_beta.append(beta_1)
            af_alpha.append(alpha_2)
            af_beta.append(beta_2)
            tot_fees.append(tot_fee)
```

```
        #tot_fees.append(tot_fee)

    hund_mill_fee = np.array(tot_fees)
    print(tot_fees)
    #hund_mill_fee = np.array(tot_fees) * 1e8
    print(hund_mill_fee.shape)
    higher_fee = np.argmax(hund_mill_fee)

    print("Part a \n-----------------")
    for i in range(len(strats)):
        print(
        """strategy: {s}
        before-fee alpha: {a}
        before-fee beta: {b}""".format(s=strats[i], a=bf_alpha[i],␣
↪b=bf_beta[i]))

    print("\nPart b \n-----------------")
    for i in range(len(strats)):
        print("""strategy: {s}
        after-fee alpha: {a}
        after-fee beta: {b}
        total fees paid: {f}""".format(s=strats[i], a=af_alpha[i],␣
↪b=af_beta[i], f=tot_fees[i]))

    print("\nPart c \n-----------------")
    print("Strategy {s} has higher fees after a $100 million investment.".
↪format(s=strats[higher_fee]))
    # Explanation for HB = higher fees:
    print("""The HB strategy has greater volatility since it is more closely␣
↪correlated to the market valuation.
        Thus, the HB strategy is capable of growing at a faster rate than the␣
↪LB strategy. Since the fees are based
        on the size of the fund, a faster-growing fund (HB) will accrue higher␣
↪fees at a faster rate. The incentive fee is based on the amount of growth,␣
↪so since we have higher growth, higher fees.""")

    print("\nPart d \n-----------------")
    print("""As a hedge fund manager, I would choose {high} since it results in␣
↪higher fees (which I gain)
    As a client, I would choose {low} since it results in lower fees (which are␣
↪a cost to me) and less outright market exposure, which we can get for free.
↪""".format(
        high=strats[higher_fee], low=strats[1-higher_fee]))
```

[ ]:
```
total_analysis()
```

```
[147849191.94575587, 3917475202.2262053]
```

```
(2,)
Part a
----------------
strategy: LB
        before-fee alpha: 0.452943909981592
        before-fee beta: 0.020905486781044837
strategy: HB
        before-fee alpha: 0.3725046631158165
        before-fee beta: 2.9978735923008775

Part b
----------------
strategy: LB
        after-fee alpha: 0.23454994106233937
        after-fee beta: 0.018252636596614236
        total fees paid: 147849191.94575587
strategy: HB
        after-fee alpha: -0.0659649295625027
        after-fee beta: 2.930809029915905
        total fees paid: 3917475202.2262053

Part c
----------------
Strategy HB has higher fees after a $100 million investment.
The HB strategy has greater volatility since it is more closely correlated to
the market valuation.
        Thus, the HB strategy is capable of growing at a faster rate than the LB
strategy. Since the fees are based
        on the size of the fund, a faster-growing fund (HB) will accrue higher
fees at a faster rate. The incentive fee is based on the amount of growth, so
since we have higher growth, higher fees.

Part d
----------------
As a hedge fund manager, I would choose HB since it results in higher fees
(which I gain)
    As a client, I would choose LB since it results in lower fees (which are a
cost to me) and less outright market exposure, which we can get for free.
```