Homework 1 Submission

Output:

```
Loading and processing data...
Loaded 397 rows of data from 199001 to 202301
Question 1a: Market Portfolio Statistics
Average Monthly Return: 0.8960
Volatility: 4.4535
Sharpe Ratio: 0.2012
Question 1b: CA Strategy Statistics
Average Monthly Return: 0.9425
Volatility: 2.6183
Sharpe Ratio: 0.3599
Question 1d: CAPM Estimates for CA Strategy
Alpha: 0.397965
Beta: 0.488712
Alpha t-statistic: 5.320164
Alpha p-value: 0.000000
Alpha Significant: True
R-squared: 0.686956
Detailed Regression Results:
                            OLS Regression Results
==============================================================================
Dep. Variable:                     CA   R-squared:                       0.687
Model:                            OLS   Adj. R-squared:                  0.686
Method:                 Least Squares   F-statistic:                     866.8
Date:                Wed, 02 Apr 2025   Prob (F-statistic):          1.17e-101
Time:                        16:53:08   Log-Likelihood:                -716.07
No. Observations:                 397   AIC:                             1436.
Df Residuals:                     395   BIC:                             1444.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3980      0.075      5.320      0.000       0.251       0.545
Mkt-RF         0.4887      0.017     29.442      0.000       0.456       0.521
==============================================================================
Omnibus:                        0.933   Durbin-Watson:                   2.286
Prob(Omnibus):                  0.627   Jarque-Bera (JB):                0.991
Skew:                           0.041   Prob(JB):                        0.609
Kurtosis:                       2.769   Cond. No.                         4.57
==============================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Question 1g: CA Strategy Evaluation
Alpha: 0.3980 (p-value: 0.0000)
Alpha is statistically significant at the 5% level
Beta: 0.4887
R-squared: 0.6870
Conclusion: CA strategy generates positive and statistically significant alpha,
making it potentially suitable as a hedge fund strategy.
Question 2: Strategy Evaluations
LBHA Strategy Analysis:
Statistics:
  Average Monthly Return: 0.6944
  Volatility: 2.1102
  Sharpe Ratio: 0.3291
CAPM Results:
  Alpha: 0.482844
  Beta: 0.004513
  Alpha t-statistic: 4.553189
  Alpha p-value: 0.000007
  Alpha Significant: True
  R-squared: 0.000093
LSA Strategy Analysis:
```

```
Statistics:
  Average Monthly Return: 0.9571
  Volatility: 3.1668
  Sharpe Ratio: 0.3022
CAPM Results:
  Alpha: 0.479427
  Beta: 0.391546
  Alpha t-statistic: 3.559889
  Alpha p-value: 0.000416
  Alpha Significant: True
  R-squared: 0.302923
TA Strategy Analysis:
Statistics:
  Average Monthly Return: 0.9612
  Volatility: 3.4561
  Sharpe Ratio: 0.2781
CAPM Results:
  Alpha: 0.403635
  Beta: 0.507766
  Alpha t-statistic: 3.038762
  Alpha p-value: 0.002533
  Alpha Significant: True
  R-squared: 0.428989
HV Strategy Analysis:
Statistics:
  Average Monthly Return: 0.9218
  Volatility: 3.8352
  Sharpe Ratio: 0.2403
CAPM Results:
  Alpha: 0.155687
  Beta: 0.811003
  Alpha t-statistic: 2.395850
  Alpha p-value: 0.017046
  Alpha Significant: True
  R-squared: 0.888986
LV Strategy Analysis:
Statistics:
  Average Monthly Return: 0.3067
  Volatility: 0.2147
  Sharpe Ratio: 1.4285
CAPM Results:
  Alpha: 0.098690
  Beta: -0.000629
  Alpha t-statistic: 20.562490
  Alpha p-value: 0.000000
  Alpha Significant: True
  R-squared: 0.000881
NA Strategy Analysis:
Statistics:
  Average Monthly Return: 0.1558
  Volatility: 2.6908
  Sharpe Ratio: 0.0579
CAPM Results:
  Alpha: -0.404671
  Beta: 0.511964
  Alpha t-statistic: -5.552380
  Alpha p-value: 0.000000
  Alpha Significant: True
  R-squared: 0.717262
LB Strategy Analysis:
Statistics:
  Average Monthly Return: 0.6758
  Volatility: 1.9407
  Sharpe Ratio: 0.3482
CAPM Results:
  Alpha: 0.452938
  Beta: 0.020911
  Alpha t-statistic: 4.628618
  Alpha p-value: 0.000005
  Alpha Significant: True
  R-squared: 0.002342
HB Strategy Analysis:
Statistics:
  Average Monthly Return: 2.6418
  Volatility: 13.3758
  Sharpe Ratio: 0.1975
CAPM Results:
  Alpha: 0.372236
  Beta: 2.997826
  Alpha t-statistic: 10.998913
  Alpha p-value: 0.000000
  Alpha Significant: True
  R-squared: 0.997527
Question 2a: LBHA vs Market Analysis
LBHA strategy outperforms the market in terms of Sharpe ratio.
LBHA strategy generates positive and statistically significant alpha.
Question 2b: LSA Analysis
LSA strategy generates positive and statistically significant alpha.
Question 2c: TA Analysis
TA strategy generates positive and statistically significant alpha.
Question 2d: HV vs LV Comparison
LV strategy has a higher Sharpe ratio than HV strategy.
Question 2e: NA Strategy Analysis
NA strategy generates negative and statistically significant alpha.
Question 2f: LB vs HB Comparison
LB Alpha: 0.4529 (p-value: 0.0000)
HB Alpha: 0.3722 (p-value: 0.0000)
LB manager is better due to higher and significant alpha.
Question 3: Fee Analysis
LB Strategy Fee Analysis:
Before Fee Alpha: 0.4529382319664177
Before Fee Beta: 0.02091068169789078
After Fee Alpha: 0.20045059368615564
After Fee Beta: 0.017734263844179844
Total Fees on $100M investment: $277.11M
Management Fees: $173.20M
Incentive Fees: $103.91M
Final Value: $496.93M
HB Strategy Fee Analysis:
Before Fee Alpha: 0.3722355286311332
Before Fee Beta: 2.9978257133733366
After Fee Alpha: -0.09785937711349652
After Fee Beta: 2.929985482305454
Total Fees on $100M investment: $7535.25M
Management Fees: $1585.92M
Incentive Fees: $5949.33M
Final Value: $12030.85M
Fee Comparison:
HB strategy earned higher fees: $7535.25M vs $277.11M for LB
Difference: $7258.15M
Question 3d: Discussion on High Beta Hedge Funds
High beta hedge funds tend to earn higher fees because:
1. They generate higher absolute returns in bull markets, leading to larger incentive fees
2. Their AUM grows faster, leading to higher management fees
3. They can market themselves based on absolute returns rather than alpha
4. The '2 and 20' fee structure rewards total returns, not just alpha generation
5. Many investors focus on total returns rather than understanding risk-adjusted performance
6. In prolonged bull markets, high beta funds appear more attractive than low beta funds
Conclusion
This analysis demonstrates the importance of separating alpha from beta when evaluating
hedge fund performance. While high beta strategies may generate larger total returns and
higher fees in bull markets, true skill is measured by consistent alpha generation.
Investors should be willing to pay for alpha, not beta which can be obtained cheaply through
passive index funds. The discrepancy between fee structures (based on total returns) and
performance evaluation (based on alpha) creates potential misalignment of incentives in
the hedge fund industry.
```

Plots:

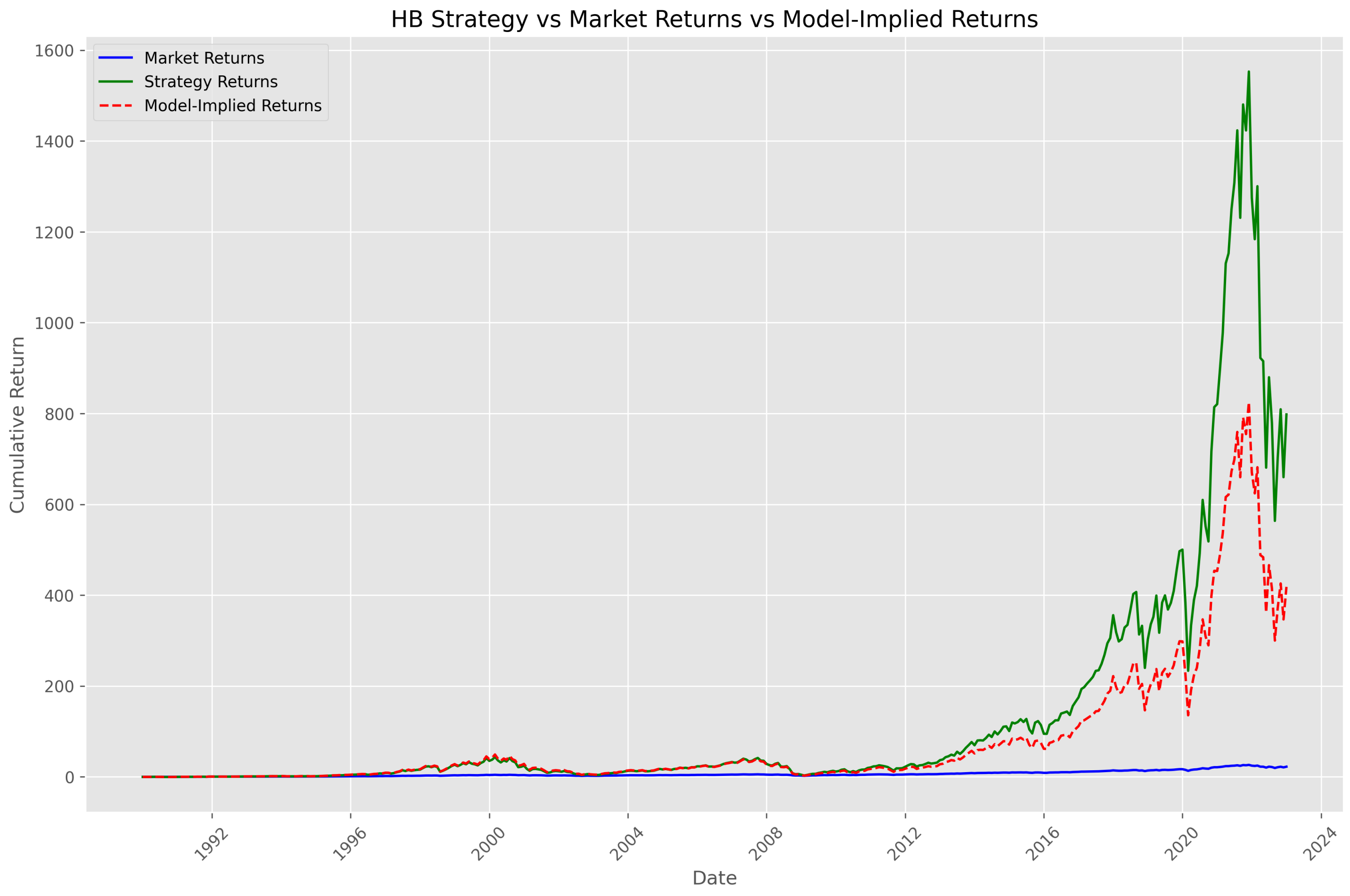

Figure: HB_returns_plot.png

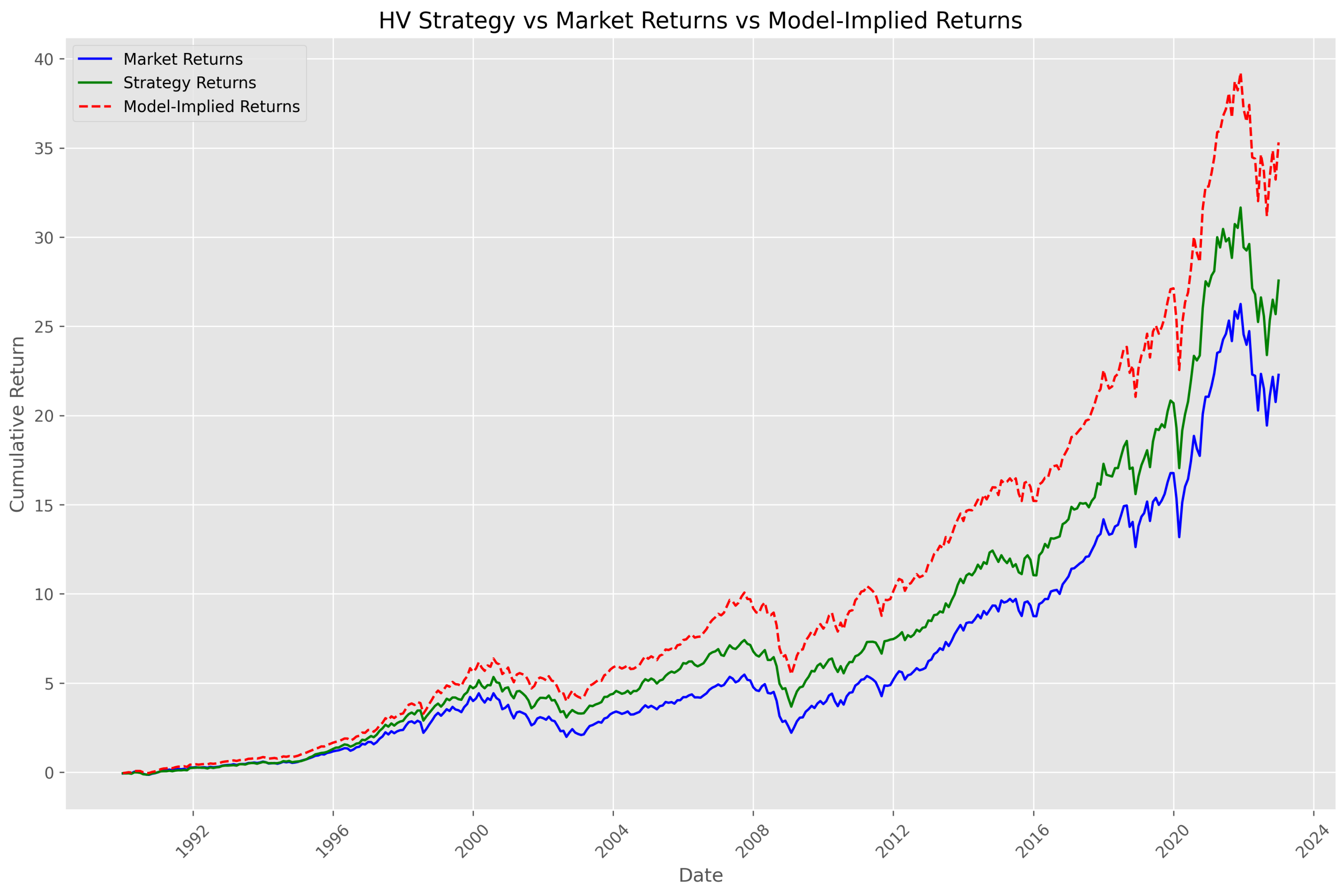

Figure: HV_returns_plot.png

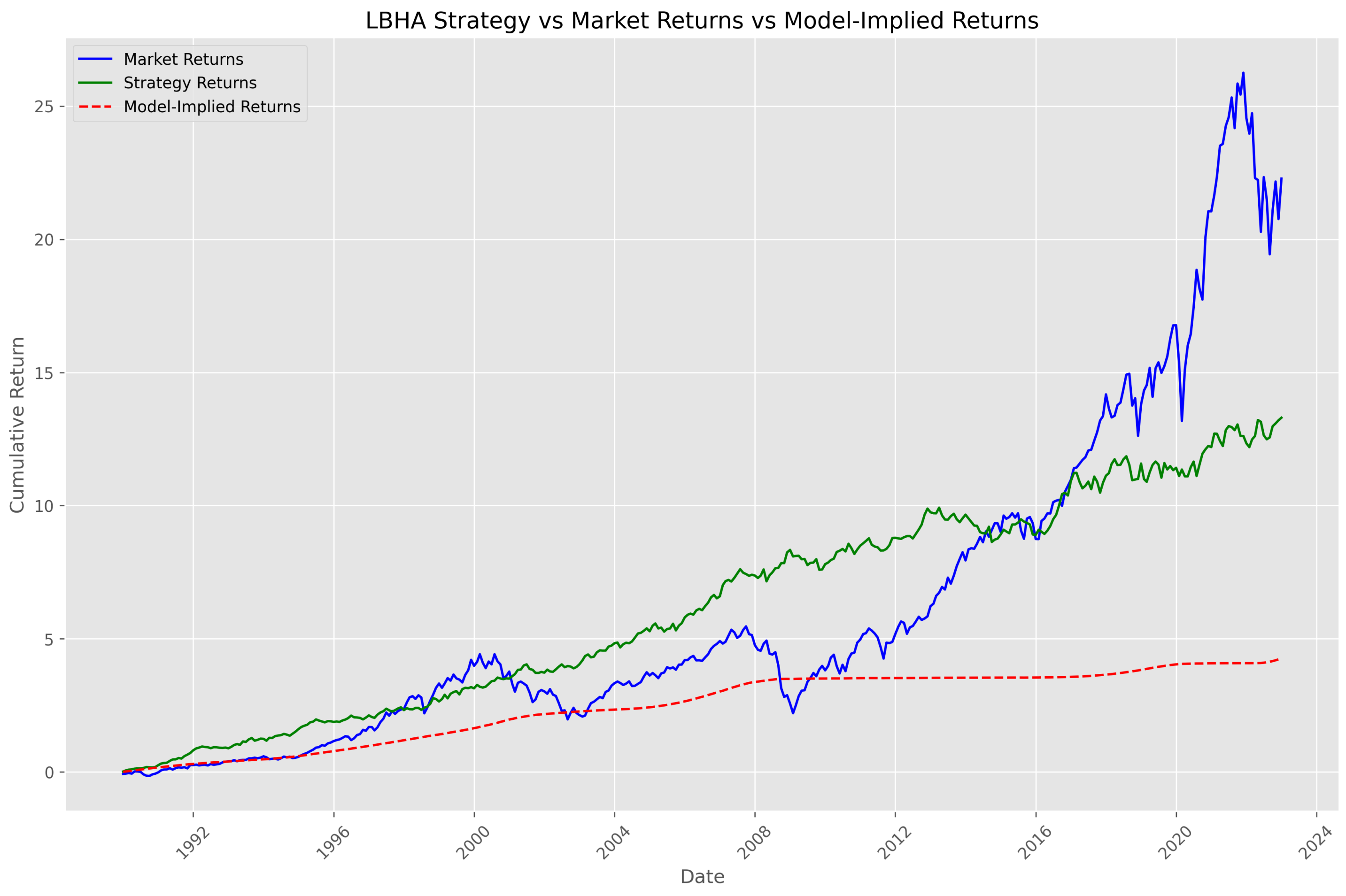

Figure: LBHA_returns_plot.png

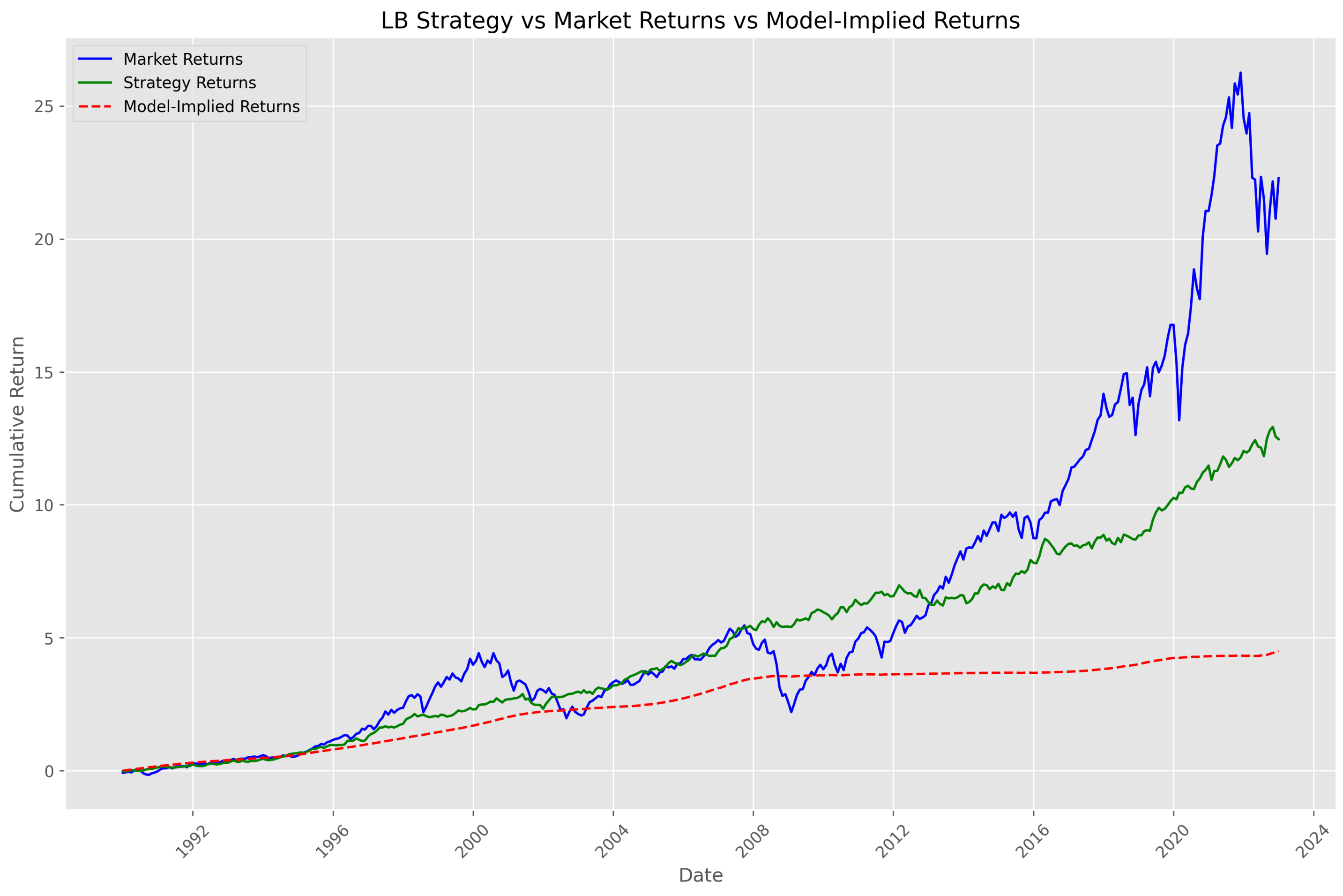

Figure: LB_returns_plot.png

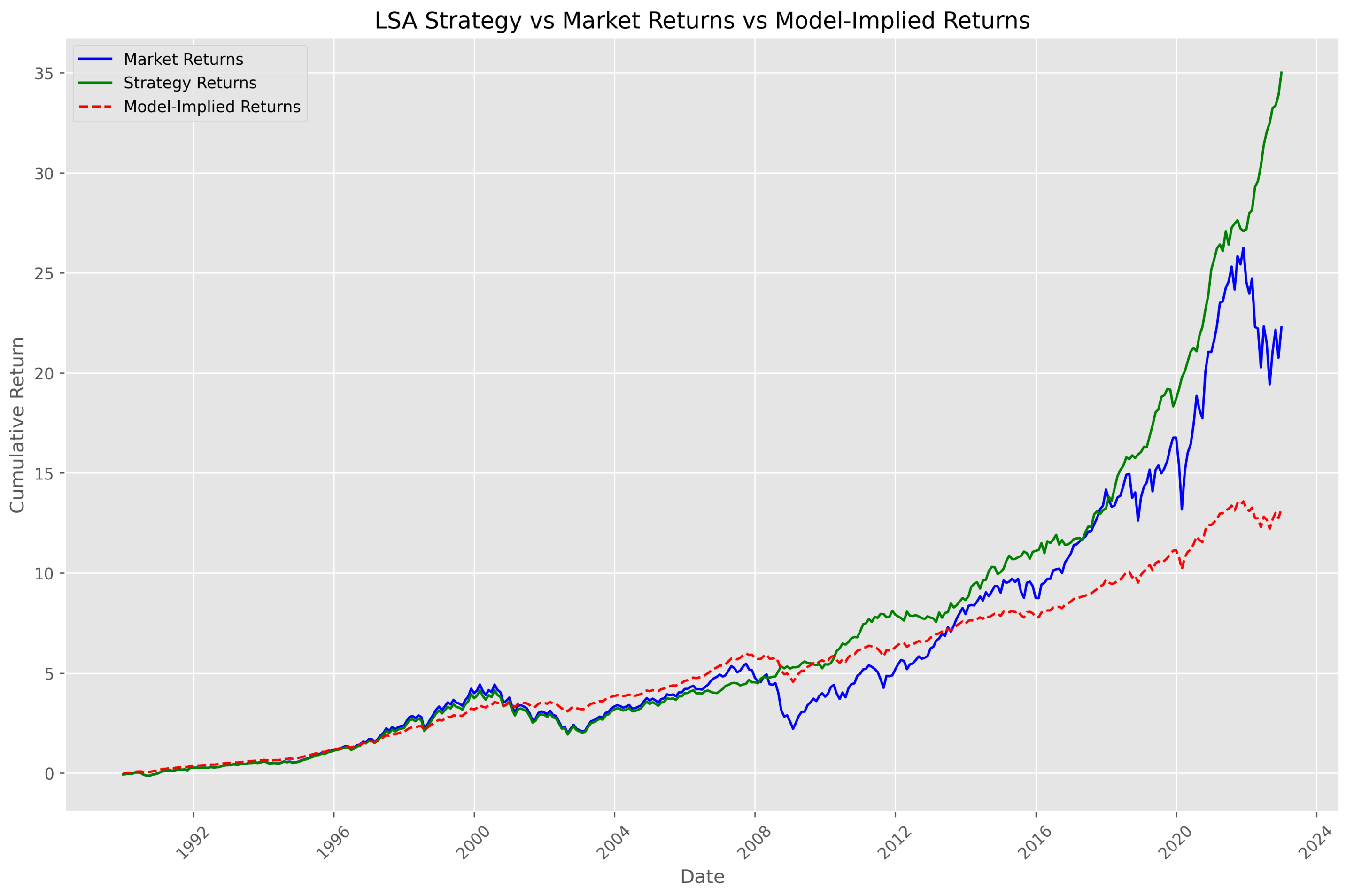

Figure: LSA_returns_plot.png

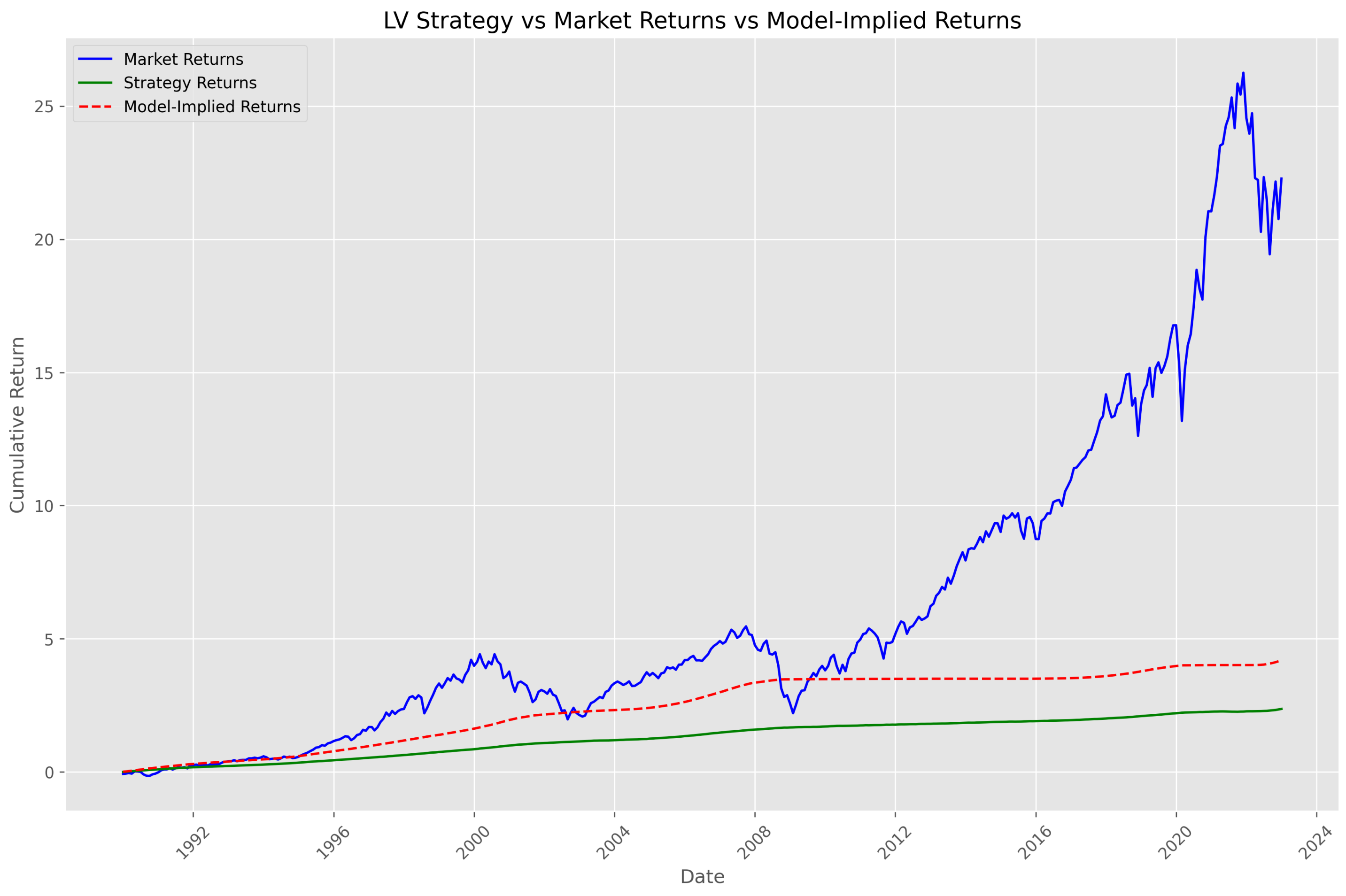

Figure: LV_returns_plot.png

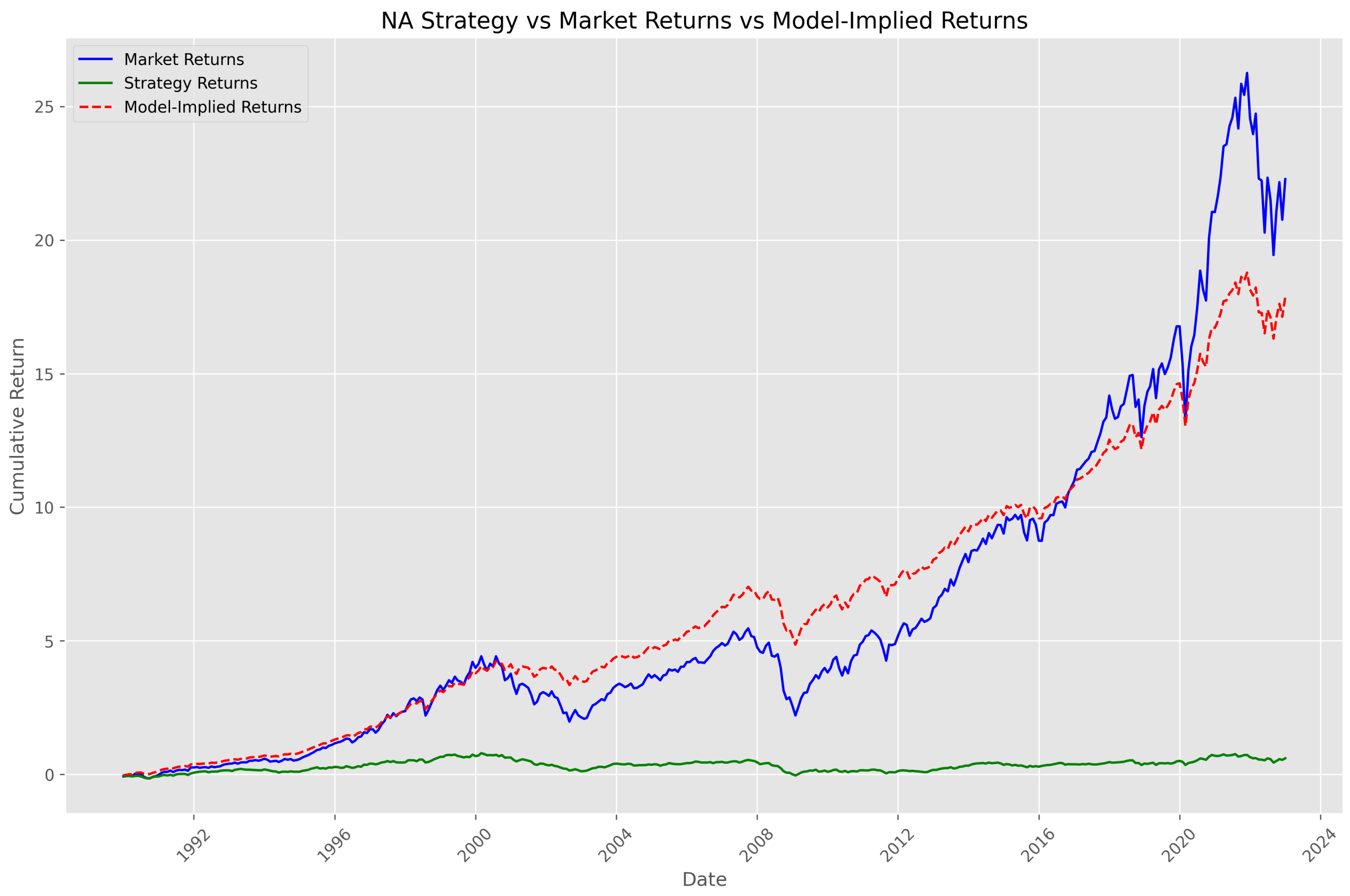

Figure: NA_returns_plot.png

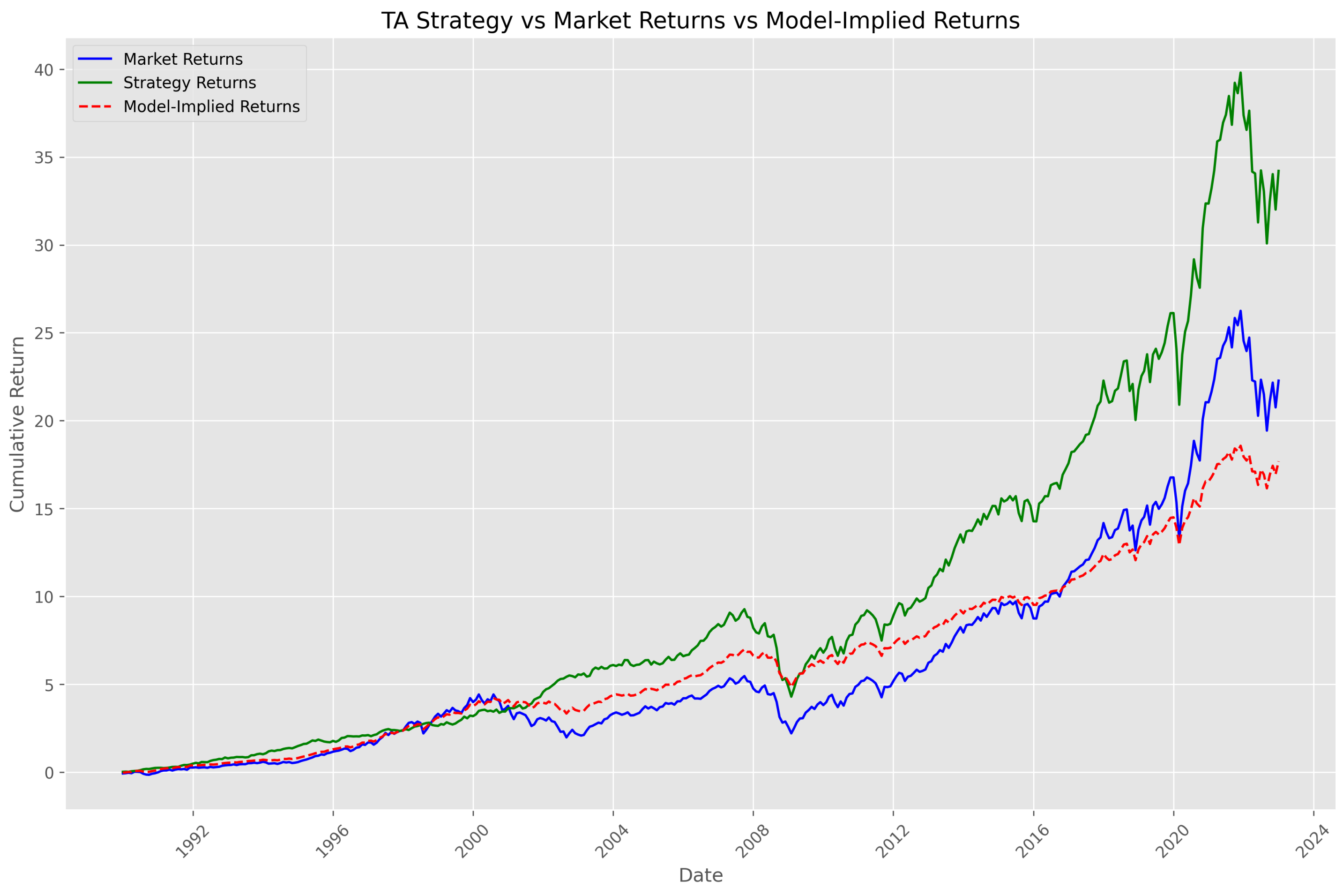

Figure: TA_returns_plot.png

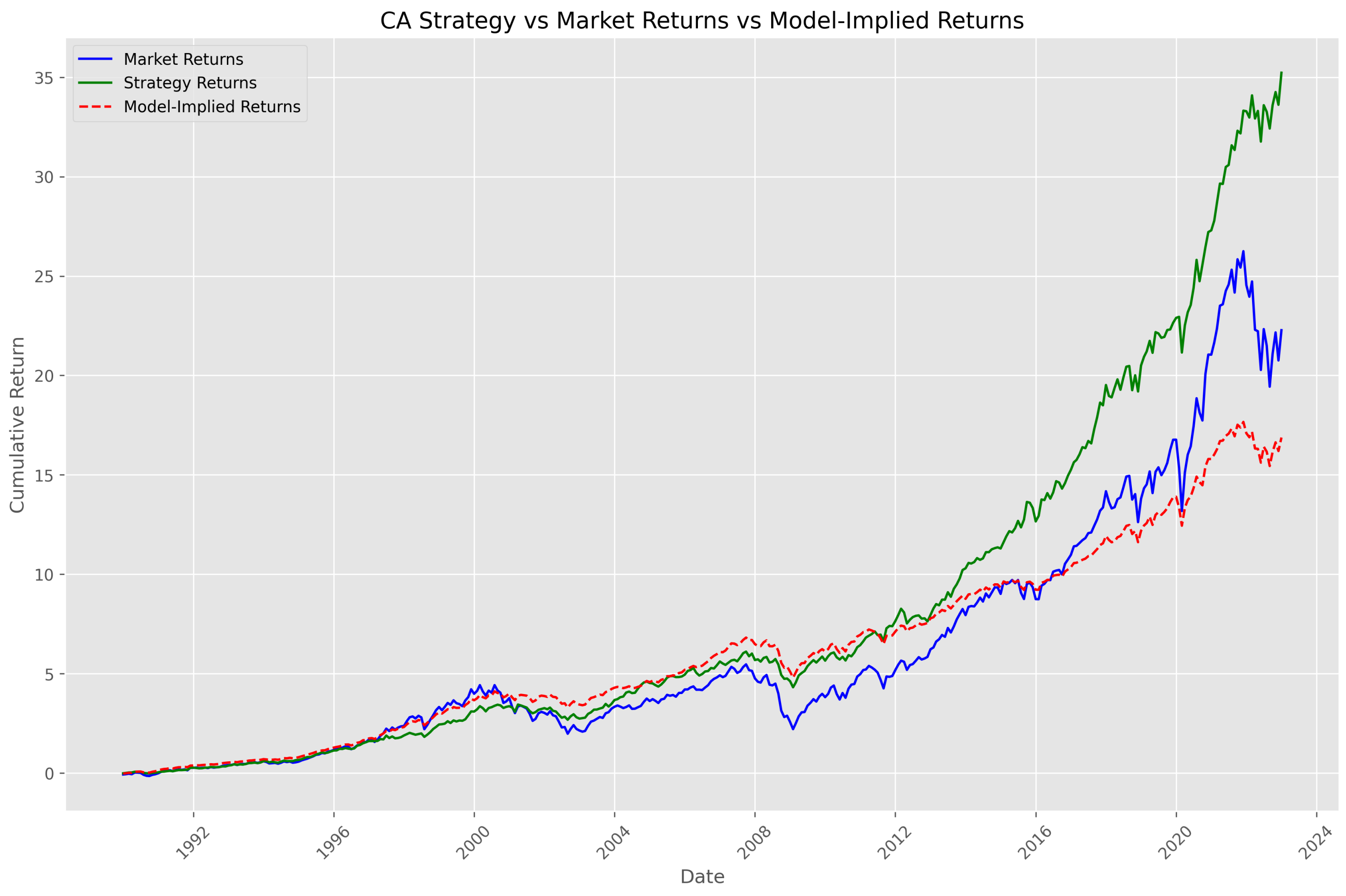

Figure: ca_returns_plot.png

Code:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.regression.linear_model import OLS
from scipy import stats
import os
import warnings
warnings.filterwarnings("ignore")
# Create plots directory if it doesn't exist
os.makedirs('plots', exist_ok=True)
# Set figure size and style for plots
plt.rcParams['figure.figsize'] = (12, 8)
plt.style.use('ggplot')
def load_data():
    """
    Load and prepare the factor and strategy datasets
    """
    # Load data
    factors_df = pd.read_csv('F-F_Research_Data_Factors.CSV', skiprows=0)
    strategies_df = pd.read_csv('ps1_strategies.csv')
    # Clean factors dataframe
    # Rename the first column which might be unnamed
    factors_df = factors_df.rename(columns={factors_df.columns[0]: 'date'})
    # Convert string values to floats (handling possible spaces)
    for col in factors_df.columns:
        if col != 'date':
            factors_df[col] = factors_df[col].astype(str).str.strip().astype(float)
    # Filter data to match strategy dates
    start_date = strategies_df['date'].min()
    end_date = strategies_df['date'].max()
    factors_df = factors_df[(factors_df['date'] >= start_date) & (factors_df['date'] <= end_date)]
    # Merge the datasets on date
    merged_df = pd.merge(strategies_df, factors_df, on='date')
    # Convert date to datetime for better plotting (YYYYMM format)
    merged_df['year'] = merged_df['date'] // 100
    merged_df['month'] = merged_df['date'] % 100
    merged_df['datetime'] = pd.to_datetime(merged_df[['year', 'month']].assign(day=1))
    return merged_df
def calculate_stats(returns):
    """
    Calculate average monthly return, volatility, and Sharpe ratio
    """
    avg_return = returns.mean()
    volatility = returns.std()
    sharpe = avg_return / volatility
    return {
        'Average Monthly Return': avg_return,
        'Volatility': volatility,
        'Sharpe Ratio': sharpe
    }
def estimate_capm(excess_returns, market_excess_returns):
    """
    Estimate CAPM model and return alpha, beta, and other statistics
    """
    # Add constant for alpha term
    X = sm.add_constant(market_excess_returns)
    # Perform regression
    model = OLS(excess_returns, X).fit()
    # Get parameters
    alpha = model.params[0]
    beta = model.params[1]
    # Get t-statistics and p-values for significance testing
    alpha_t_stat = model.tvalues[0]
    alpha_p_value = model.pvalues[0]
    # Determine if alpha is significant at 5% level
    alpha_significant = alpha_p_value < 0.05
    return {
        'Alpha': alpha,
        'Beta': beta,
        'Alpha t-statistic': alpha_t_stat,
        'Alpha p-value': alpha_p_value,
        'Alpha Significant': alpha_significant,
        'R-squared': model.rsquared,
        'Model': model
    }
def calculate_capm_returns(beta, risk_free_rate, market_excess_returns):
    """
    Calculate CAPM implied returns: R■p = Rf + β■p(RM - Rf)
    """
    return risk_free_rate + beta * market_excess_returns
def plot_cumulative_returns(datetime, market_returns, strategy_returns, model_returns, title):
    """
    Plot cumulative returns for market, strategy, and model-implied strategy
    """
    # Calculate cumulative returns (using log returns for additivity)
    cum_market = np.exp(np.log(1 + market_returns / 100).cumsum()) - 1
    cum_strategy = np.exp(np.log(1 + strategy_returns / 100).cumsum()) - 1
    cum_model = np.exp(np.log(1 + model_returns / 100).cumsum()) - 1
    # Plot results
    plt.figure()
    plt.plot(datetime, cum_market, 'b-', label='Market Returns')
    plt.plot(datetime, cum_strategy, 'g-', label='Strategy Returns')
    plt.plot(datetime, cum_model, 'r--', label='Model-Implied Returns')
    plt.title(title)
    plt.xlabel('Date')
    plt.ylabel('Cumulative Return')
    plt.legend()
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    return plt.gcf()  # Return the figure for later use
def analyze_strategy(df, strategy_name, market_col='Mkt-RF', rf_col='RF'):
    """Analyze a strategy with CAPM and plot results"""
    # Calculate strategy's total returns
    strategy_excess_returns = df[strategy_name]
    strategy_total_returns = strategy_excess_returns + df[rf_col]
    # Calculate statistics
    stats = calculate_stats(strategy_total_returns)
    # Estimate CAPM
    capm_results = estimate_capm(strategy_excess_returns, df[market_col])
    # Calculate CAPM implied returns
    implied_returns = calculate_capm_returns(capm_results['Beta'], df[rf_col], df[market_col])
    # Plot cumulative returns
    fig = plot_cumulative_returns(
        df['datetime'],
        df[market_col] + df[rf_col],  # Market total returns
        strategy_total_returns,        # Strategy total returns
        implied_returns + df[rf_col],  # Model-implied total returns
        f'{strategy_name} Strategy vs Market Returns vs Model-Implied Returns'
    )
    # Save the plot
    fig.savefig(f'plots/{strategy_name}_returns_plot.png', dpi=300, bbox_inches='tight')
    return {
        'Statistics': stats,
        'CAPM': capm_results,
        'Figure': fig
    }
def calculate_after_fee_capm(df, strategy_name):
    """Calculate after-fee returns and estimate CAPM"""
    # Get before-fee statistics
    strategy_excess_returns = df[strategy_name]
    strategy_total_returns = strategy_excess_returns + df['RF']
    # Calculate before-fee CAPM
    before_fee_capm = estimate_capm(strategy_excess_returns, df['Mkt-RF'])
    # Apply the 1.8% and 20% fee structure to get after-fee returns
    # For a more precise calculation, we need to simulate the actual cash flows with fees
    # Initialize variables for fee simulation
    initial_investment = 100  # $100 million
    current_value = initial_investment
    max_value = initial_investment  # High water mark
    after_fee_returns = []
    management_fees = []
    incentive_fees = []
    # Fee parameters
    monthly_mgmt_fee_rate = 0.018 / 12  # 1.8% annually
    incentive_fee_rate = 0.2  # 20%
    # Calculate after-fee returns
    for i, monthly_return in enumerate(strategy_total_returns):
        # Calculate pre-fee value gain
        monthly_return_decimal = monthly_return / 100
        pre_fee_gain = current_value * monthly_return_decimal
        pre_fee_value = current_value + pre_fee_gain
        # Calculate management fee
        mgmt_fee = current_value * monthly_mgmt_fee_rate
        management_fees.append(mgmt_fee)
        # Calculate post-management fee value
        post_mgmt_value = pre_fee_value - mgmt_fee
        # Calculate incentive fee if there's a new high water mark
        if post_mgmt_value > max_value:
            incentive_fee = incentive_fee_rate * (post_mgmt_value - max_value)
            max_value = post_mgmt_value - incentive_fee  # Update high water mark
        else:
            incentive_fee = 0
        incentive_fees.append(incentive_fee)
        # Final value after all fees
        current_value = post_mgmt_value - incentive_fee
        # Calculate after-fee return for this period
        after_fee_return = (current_value / (pre_fee_value - pre_fee_gain) - 1) * 100
        after_fee_returns.append(after_fee_return - df['RF'].iloc[i])  # Convert to excess return
    # Convert to numpy array
    after_fee_returns = np.array(after_fee_returns)
    # Calculate after-fee CAPM
    after_fee_capm = estimate_capm(after_fee_returns, df['Mkt-RF'])
    # Calculate total fees
    total_management_fees = sum(management_fees)
    total_incentive_fees = sum(incentive_fees)
    total_fees = total_management_fees + total_incentive_fees
    return {
        'Before Fee CAPM': before_fee_capm,
        'After Fee CAPM': after_fee_capm,
        'Management Fees': total_management_fees,
        'Incentive Fees': total_incentive_fees,
        'Total Fees': total_fees,
        'Final Value': current_value
    }
# Main execution
if __name__ == "__main__":
    # Load the data
    print("Loading and processing data...")
    df = load_data()
    print(f"Loaded {len(df)} rows of data from {df['date'].min()} to {df['date'].max()}")
    # Question 1: Build a Simple Analysis Tool
    # 1a: Calculate market portfolio statistics
    market_returns = df['Mkt-RF'] + df['RF']  # Convert from excess to total return
    market_stats = calculate_stats(market_returns)
    print("\nQuestion 1a: Market Portfolio Statistics")
    for key, value in market_stats.items():
        print(f"{key}: {value:.4f}")
    # 1b: Calculate CA strategy statistics
    ca_returns = df['CA'] + df['RF']  # Convert from excess to total return
    ca_stats = calculate_stats(ca_returns)
    print("\nQuestion 1b: CA Strategy Statistics")
    for key, value in ca_stats.items():
        print(f"{key}: {value:.4f}")
    # 1c-1d: Estimate CAPM for CA strategy
    ca_capm = estimate_capm(df['CA'], df['Mkt-RF'])
    print("\nQuestion 1d: CAPM Estimates for CA Strategy")
    for key, value in ca_capm.items():
        if key != 'Model':
            print(f"{key}: {value:.6f}" if isinstance(value, (int, float)) else f"{key}: {value}")
    # Print complete regression results
    print("\nDetailed Regression Results:")
    print(ca_capm['Model'].summary())
    # 1e: Calculate CAPM implied returns
    ca_implied_returns = calculate_capm_returns(ca_capm['Beta'], df['RF'], df['Mkt-RF'])
    # 1f: Plot cumulative returns
    ca_fig = plot_cumulative_returns(
        df['datetime'],
        market_returns,
        ca_returns,
        ca_implied_returns + df['RF'],  # Convert back to total return
        'CA Strategy vs Market Returns vs Model-Implied Returns'
    )
    # Save the plot
    ca_fig.savefig('plots/ca_returns_plot.png', dpi=300, bbox_inches='tight')
    # 1g: Evaluate if CA would make a good hedge fund strategy
    print("\nQuestion 1g: CA Strategy Evaluation")
    print(f"Alpha: {ca_capm['Alpha']:.4f} (p-value: {ca_capm['Alpha p-value']:.4f})")
    print(f"Alpha is {'statistically significant' if ca_capm['Alpha Significant'] else 'not
statistically significant'} at the 5% level")
    print(f"Beta: {ca_capm['Beta']:.4f}")
    print(f"R-squared: {ca_capm['R-squared']:.4f}")
    if ca_capm['Alpha'] > 0 and ca_capm['Alpha Significant']:
        print("Conclusion: CA strategy generates positive and statistically significant alpha,")
        print("making it potentially suitable as a hedge fund strategy.")
    else:
        print("Conclusion: CA strategy does not generate statistically significant alpha,")
        print("making it less attractive as a hedge fund strategy.")
    # Question 2: Evaluating Returns
    # Analyze all strategies for question 2
    strategies = ['LBHA', 'LSA', 'TA', 'HV', 'LV', 'NA', 'LB', 'HB']
    strategy_results = {}
    print("\nQuestion 2: Strategy Evaluations")
    for strategy in strategies:
        strategy_results[strategy] = analyze_strategy(df, strategy)
        print(f"\n{strategy} Strategy Analysis:")
        print("Statistics:")
        for key, value in strategy_results[strategy]['Statistics'].items():
            print(f"  {key}: {value:.4f}")
        print("CAPM Results:")
        for key, value in strategy_results[strategy]['CAPM'].items():
            if key != 'Model':
                print(f"  {key}: {value:.6f}" if isinstance(value, (int, float)) else f"  {key}:
{value}")
    # Question 2a: LBHA vs Market
    print("\nQuestion 2a: LBHA vs Market Analysis")
    if strategy_results['LBHA']['Statistics']['Sharpe Ratio'] > market_stats['Sharpe Ratio']:
        print("LBHA strategy outperforms the market in terms of Sharpe ratio.")
    else:
        print("LBHA strategy underperforms the market in terms of Sharpe ratio.")
    if strategy_results['LBHA']['CAPM']['Alpha'] > 0 and strategy_results['LBHA']['CAPM']['Alpha
Significant']:
        print("LBHA strategy generates positive and statistically significant alpha.")
    else:
        print("LBHA strategy does not generate statistically significant alpha.")
    # Question 2b: LSA Analysis
    print("\nQuestion 2b: LSA Analysis")
    if strategy_results['LSA']['CAPM']['Alpha'] > 0 and strategy_results['LSA']['CAPM']['Alpha
Significant']:
        print("LSA strategy generates positive and statistically significant alpha.")
    else:
        print("LSA strategy does not generate statistically significant alpha.")
    # Question 2c: TA Analysis
    print("\nQuestion 2c: TA Analysis")
    if strategy_results['TA']['CAPM']['Alpha'] > 0 and strategy_results['TA']['CAPM']['Alpha
Significant']:
        print("TA strategy generates positive and statistically significant alpha.")
    else:
        print("TA strategy does not generate statistically significant alpha.")
    # Question 2d: HV vs LV Comparison
    print("\nQuestion 2d: HV vs LV Comparison")
    if strategy_results['HV']['Statistics']['Sharpe Ratio'] >
strategy_results['LV']['Statistics']['Sharpe Ratio']:
        print("HV strategy has a higher Sharpe ratio than LV strategy.")
    else:
        print("LV strategy has a higher Sharpe ratio than HV strategy.")
    # Question 2e: NA Strategy Analysis
    print("\nQuestion 2e: NA Strategy Analysis")
    if strategy_results['NA']['CAPM']['Alpha'] < 0 and strategy_results['NA']['CAPM']['Alpha
Significant']:
        print("NA strategy generates negative and statistically significant alpha.")
    else:
        print("NA strategy does not generate statistically significant negative alpha.")
    # Question 2f: LB vs HB Comparison
    print("\nQuestion 2f: LB vs HB Comparison")
    lb_alpha = strategy_results['LB']['CAPM']['Alpha']
    hb_alpha = strategy_results['HB']['CAPM']['Alpha']
    print(f"LB Alpha: {lb_alpha:.4f} (p-value: {strategy_results['LB']['CAPM']['Alpha
p-value']:.4f})")
    print(f"HB Alpha: {hb_alpha:.4f} (p-value: {strategy_results['HB']['CAPM']['Alpha
p-value']:.4f})")
    if lb_alpha > hb_alpha and strategy_results['LB']['CAPM']['Alpha Significant']:
        print("LB manager is better due to higher and significant alpha.")
    elif hb_alpha > lb_alpha and strategy_results['HB']['CAPM']['Alpha Significant']:
        print("HB manager is better due to higher and significant alpha.")
    else:
        if strategy_results['LB']['Statistics']['Sharpe Ratio'] >
strategy_results['HB']['Statistics']['Sharpe Ratio']:
            print("LB manager is better due to higher risk-adjusted returns (Sharpe ratio).")
        else:
            print("HB manager is better due to higher risk-adjusted returns (Sharpe ratio).")
    # Question 3: Fees Analysis
    # 3a-3b: Calculate before-fee and after-fee alphas and betas
    print("\nQuestion 3: Fee Analysis")
    # Only analyze LB and HB for fee comparison
    fee_strategies = ['LB', 'HB']
    fee_results = {}
    for strategy in fee_strategies:
        fee_results[strategy] = calculate_after_fee_capm(df, strategy)
        print(f"\n{strategy} Strategy Fee Analysis:")
        print("Before Fee Alpha:", fee_results[strategy]['Before Fee CAPM']['Alpha'])
        print("Before Fee Beta:", fee_results[strategy]['Before Fee CAPM']['Beta'])
        print("After Fee Alpha:", fee_results[strategy]['After Fee CAPM']['Alpha'])
        print("After Fee Beta:", fee_results[strategy]['After Fee CAPM']['Beta'])
        print(f"Total Fees on $100M investment: ${fee_results[strategy]['Total Fees']:.2f}M")
        print(f"Management Fees: ${fee_results[strategy]['Management Fees']:.2f}M")
        print(f"Incentive Fees: ${fee_results[strategy]['Incentive Fees']:.2f}M")
        print(f"Final Value: ${fee_results[strategy]['Final Value']:.2f}M")
    # Compare fee earnings between strategies
    lb_fees = fee_results['LB']['Total Fees']
    hb_fees = fee_results['HB']['Total Fees']
    print("\nFee Comparison:")
    if lb_fees > hb_fees:
        print(f"LB strategy earned higher fees: ${lb_fees:.2f}M vs ${hb_fees:.2f}M for HB")
        print(f"Difference: ${lb_fees - hb_fees:.2f}M")
    else:
        print(f"HB strategy earned higher fees: ${hb_fees:.2f}M vs ${lb_fees:.2f}M for LB")
        print(f"Difference: ${hb_fees - lb_fees:.2f}M")
    # Question 3d: Discussion on high beta hedge funds
    print("\nQuestion 3d: Discussion on High Beta Hedge Funds")
    print("High beta hedge funds tend to earn higher fees because:")
    print("1. They generate higher absolute returns in bull markets, leading to larger incentive
fees")
    print("2. Their AUM grows faster, leading to higher management fees")
    print("3. They can market themselves based on absolute returns rather than alpha")
    print("4. The '2 and 20' fee structure rewards total returns, not just alpha generation")
    print("5. Many investors focus on total returns rather than understanding risk-adjusted
performance")
    print("6. In prolonged bull markets, high beta funds appear more attractive than low beta
funds")
    # Conclusion
    print("\nConclusion")
    print("This analysis demonstrates the importance of separating alpha from beta when evaluating")
    print("hedge fund performance. While high beta strategies may generate larger total returns
and")
    print("higher fees in bull markets, true skill is measured by consistent alpha generation.")
    print("Investors should be willing to pay for alpha, not beta which can be obtained cheaply
through")
    print("passive index funds. The discrepancy between fee structures (based on total returns)
and")
    print("performance evaluation (based on alpha) creates potential misalignment of incentives in")
    print("the hedge fund industry.")
```