1. Key Principles of Statistical Arbitrage

A. Mean Reversion

One of the core assumptions behind statistical arbitrage is the idea of **mean reversion**, which suggests that prices of related assets tend to move toward a long-term average. Stat arb strategies often involve identifying pairs or groups of assets whose prices have diverged from their historical relationship and betting that they will eventually revert to their mean.

Example:

- Stocks in the same industry, such as HdfcBank and KotakBank, may have highly correlated price movements. If their prices diverge significantly due to short-term market inefficiencies, a statistical arbitrage strategy might short the overperforming stock and go long on the underperforming one, expecting them to converge over time.

B. Cointegration

Cointegration is a statistical method used to test whether two or more time series (like stock prices) share a long-term equilibrium relationship. If two stocks are cointegrated, their prices may diverge in the short term but are likely to revert to their historical equilibrium. This makes cointegration a key tool in statistical arbitrage for identifying potential trading pairs.

Mathematically, two time series X_t and Y_t are cointegrated if there exists a linear combination of the two that is stationary:

$$Z_t = X_t - \beta Y_t$$

Where Z_t is the stationary residual, and β is a constant.

---

2. Common Statistical Arbitrage Strategies

A. Pairs Trading

Pairs trading is one of the most popular statistical arbitrage strategies. It involves identifying two securities that are historically correlated or cointegrated and trading them based on the divergence from their historical relationship.

Steps involved in pairs trading:

1. **Identify pairs**: Use statistical methods (such as cointegration tests) to find two stocks with a stable long-term relationship.
2. **Monitor divergence**: Track the price spread between the two stocks.
3. **Trade on divergence**: When the spread deviates significantly from its historical mean, take a long position in the underperforming stock and a short position in the overperforming stock.
4. **Exit the trade**: Close both positions when the spread reverts to its mean.

B. Index Arbitrage

Index arbitrage involves taking advantage of price discrepancies between an index (e.g., S&P 500) and its constituent stocks or between an index ETF and futures contracts. Traders buy the underpriced instrument and sell the overpriced one, expecting the prices to converge.

C. Market Neutral Strategies

Market neutral strategies aim to generate profits regardless of market direction. This is done by simultaneously taking long and short positions in pairs or groups of securities to hedge out market risk (beta). The goal is to isolate alpha—profits generated from exploiting relative mispricings rather than broader market movements.

Pairs Trading Strategy: KOTAKBANK and HDFCBANK

What is Pairs Trading?

Pairs trading is a market-neutral strategy that capitalizes on the price relationship between two highly correlated stocks. The strategy involves selecting two stocks that historically move together, such as **KOTAKBANK** and **HDFCBANK**. When one stock deviates significantly from its typical price relationship with the other, a trading opportunity is created. You buy the undervalued stock and sell the overvalued one, betting that their prices will converge to their historical norm.

This strategy is ideal for traders seeking a low-risk, market-neutral approach, as profits are derived from the relative movement of two stocks, not from market direction.

---

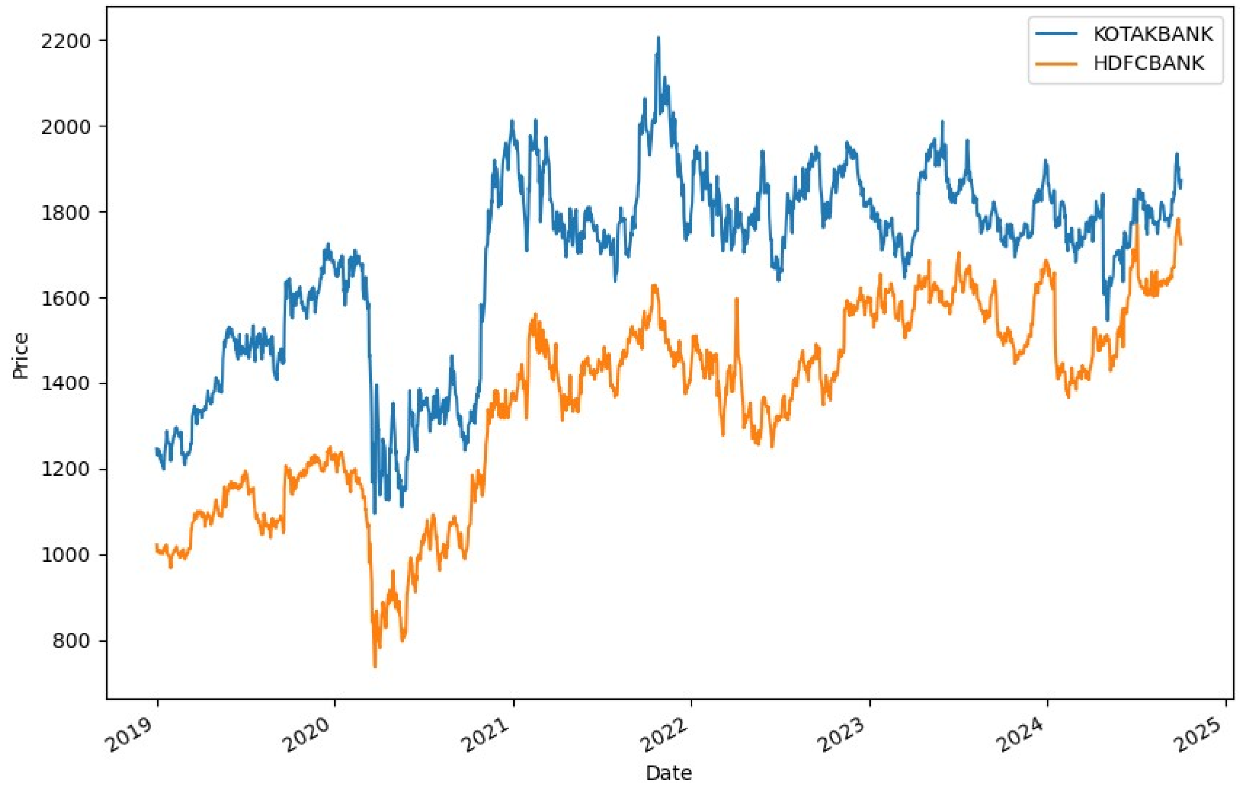
1. Data Collection and Visualization

First, we import the necessary libraries for data manipulation, visualization, and statistical analysis. Then, we use the `yfinance` API to fetch historical stock prices for KOTAKBANK and HDFCBANK.

```
#importing libraries for data manipulation and visualization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import yfinance as yf
import warnings
warnings.filterwarnings('ignore')

#downloading data from yfinance api
end = '2024-12-31'
start = '2019-1-1'
```

```python
#downloading KOTAKBANK and HDFCBANK last 5 years close prices from
yfinance
x = yf.download('KOTAKBANK.NS',start,end)['Adj Close']
y = yf.download('HDFCBANK.NS',start,end)['Adj Close']

df = pd.concat([x,y],axis=1)
df.columns = ['KOTAKBANK','HDFCBANK']

# Plotting the prices for visualization
df.plot(figsize=(10,7))
plt.ylabel("Price")
plt.show()
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```



Explanation:

- We retrieve historical adjusted close prices for **KOTAKBANK** and **HDFCBANK** over a 5-year period (2019-2024).
- The prices are plotted to visually inspect any correlation between the two stocks. If the prices move together closely, they may be suitable candidates for pairs trading.

Interpretation:

The price chart should show a clear **co-movement** between the two stocks, indicating a potential trading relationship. However, before executing the strategy, we need to analyze the statistical relationship more rigorously.

---

2. Hedge Ratio Calculation Using Linear Regression

In pairs trading, the **hedge ratio** indicates the degree to which one stock's price moves in relation to the other. We calculate the hedge ratio by regressing KOTAKBANK's price on HDFCBANK's price using linear regression.

```python
# Performing linear regression between HDFC Bank ('HDFCBANK') and
Kotak Mahindra Bank ('KOTAKBANK')
X = df['HDFCBANK']
y = df['KOTAKBANK']

model = np.polyfit(X,y,deg=1)
hedge_ratio = model[0]
hr = round(hedge_ratio, 3)
print(f'The hedge ratio is {hr}')
```

```
The hedge ratio is 0.851
```

Explanation:

- We use **linear regression** (`np.polyfit()`) to find the **slope** between KOTAKBANK and HDFCBANK prices, which represents the **hedge ratio**.
- The hedge ratio of **0.851** means that for every 1 unit change in HDFCBANK's price, KOTAKBANK's price moves by 0.851 units.

Interpretation:

The hedge ratio tells us how much of one stock to buy or sell relative to the other. In this case, you would need to short 0.851 shares of HDFCBANK for every share of KOTAKBANK that you buy when executing the pairs trade.

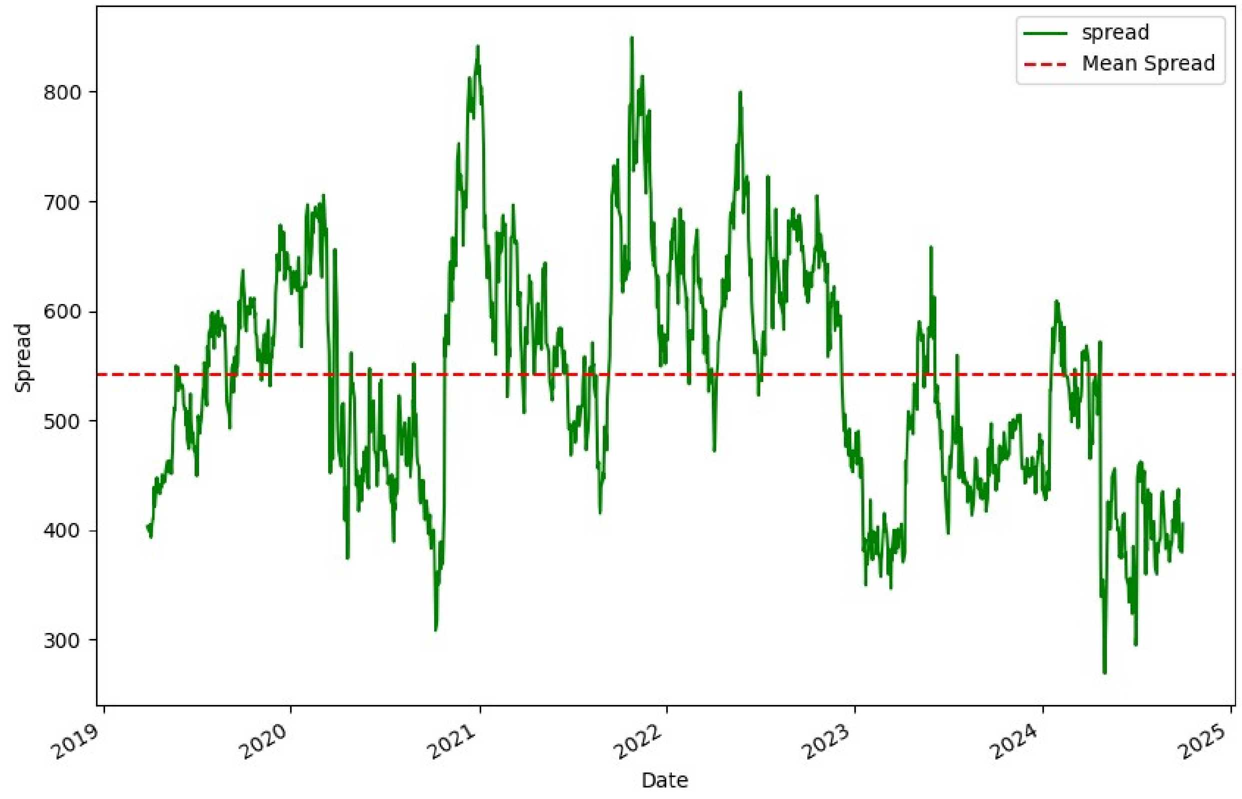
---

3. Calculating and Plotting the Spread

The **spread** is the difference between KOTAKBANK's price and the hedge ratio-adjusted price of HDFCBANK. A significant divergence in the spread from its historical mean signals a trading opportunity.

```python
# Calculating the spread
df['spread'] = df.KOTAKBANK - hr * df.HDFCBANK

# Plotting the spread
```

```python
df.spread.plot(figsize=(10,7), color='g')
plt.axhline(df.spread.mean(), color='red', linestyle='--', label='Mean
Spread')
plt.ylabel("Spread")
plt.legend()
plt.show()
```



Explanation:

- We calculate the **spread** by subtracting HDFCBANK’s price (multiplied by the hedge ratio) from KOTAKBANK’s price.
- The **mean spread** is plotted as a red horizontal line, and deviations from this mean highlight potential trading opportunities.

Interpretation:

A **positive spread** suggests KOTAKBANK is overpriced relative to HDFCBANK, while a **negative spread** suggests it is underpriced. The strategy involves going long or short based on these spread divergences, assuming they will revert to the mean.

4. Testing for Cointegration

Cointegration is a critical concept in pairs trading. It indicates that the price relationship between two assets is mean-reverting. We use the **Augmented Dickey-Fuller (ADF) test** to check if the spread is stationary.

```python
# Importing the ADF test function from statsmodels.tsa.stattools
from statsmodels.tsa.stattools import adfuller

# Applying the ADF test to the 'spread' series
adf_result = adfuller(df.spread, maxlag=1)

# Printing the ADF test results
print("ADF Statistic:", adf_result[0])
print("p-value:", adf_result[1])
print("Critical Values:", adf_result[4])

# Check if the spread series is stationary based on the p-value
if adf_result[1] < 0.05:
    print("The spread series is likely stationary.")
else:
    print("The spread series is likely not stationary.")
```

```
ADF Statistic: -4.129854798926771
p-value: 0.0008641393148863315
Critical Values: {'1%': -3.434966750462565, '5%': -2.8635789736973725,
'10%': -2.5678555388041384}
The spread series is likely stationary.
```

Explanation:

- The **ADF test** checks whether the spread is **stationary** (i.e., it reverts to a mean over time).
- If the **p-value** is below 0.05, the spread is considered stationary, confirming the pairs trade's viability.

Interpretation:

With a **p-value of 0.00086**, the spread is stationary, meaning KOTAKBANK and HDFCBANK have a long-term equilibrium, and the spread will likely revert to the mean over time.

---

5. Implementing a Mean Reversion Trading Strategy

Using the spread, we implement a **mean reversion strategy** that buys the pair when the spread is below a lower threshold (long entry) and sells when the spread is above an upper threshold (short entry).

```python
def mean_reversion_strategy(df, period, std_dev):
    # Moving Average
    df['moving_average'] = df.spread.rolling(period).mean()
    # Moving Standard Deviation
    df['moving_std_dev'] = df.spread.rolling(period).std()

    # Upper and lower bands
    df['upper_band'] = df.moving_average + std_dev * df.moving_std_dev
    df['lower_band'] = df.moving_average - std_dev * df.moving_std_dev

    # Long and short positions
    df['long_entry'] = df.spread < df.lower_band
    df['long_exit'] = df.spread >= df.moving_average
    df['short_entry'] = df.spread > df.upper_band
    df['short_exit'] = df.spread <= df.moving_average

    # Filling long positions
    df['positions_long'] = np.nan
    df.loc[df.long_entry, 'positions_long'] = 1
    df.loc[df.long_exit, 'positions_long'] = 0
    df.positions_long = df.positions_long.fillna(method='ffill')

    # Filling short positions
    df['positions_short'] = np.nan
    df.loc[df.short_entry, 'positions_short'] = -1
    df.loc[df.short_exit, 'positions_short'] = 0
    df.positions_short = df.positions_short.fillna(method='ffill')

    # Combined positions
    df['positions'] = df.positions_long + df.positions_short

    return df

# Applying the strategy
df = mean_reversion_strategy(df, 30, 1)
df.dropna(inplace=True)
```

Explanation:

- We compute the **moving average** and **moving standard deviation** of the spread over a 30-day rolling window.
- The strategy goes **long** when the spread falls below the lower band (mean - 1 standard deviation) and **short** when the spread rises above the upper band (mean + 1 standard deviation).
- **Position entries and exits** are based on whether the spread crosses these thresholds.

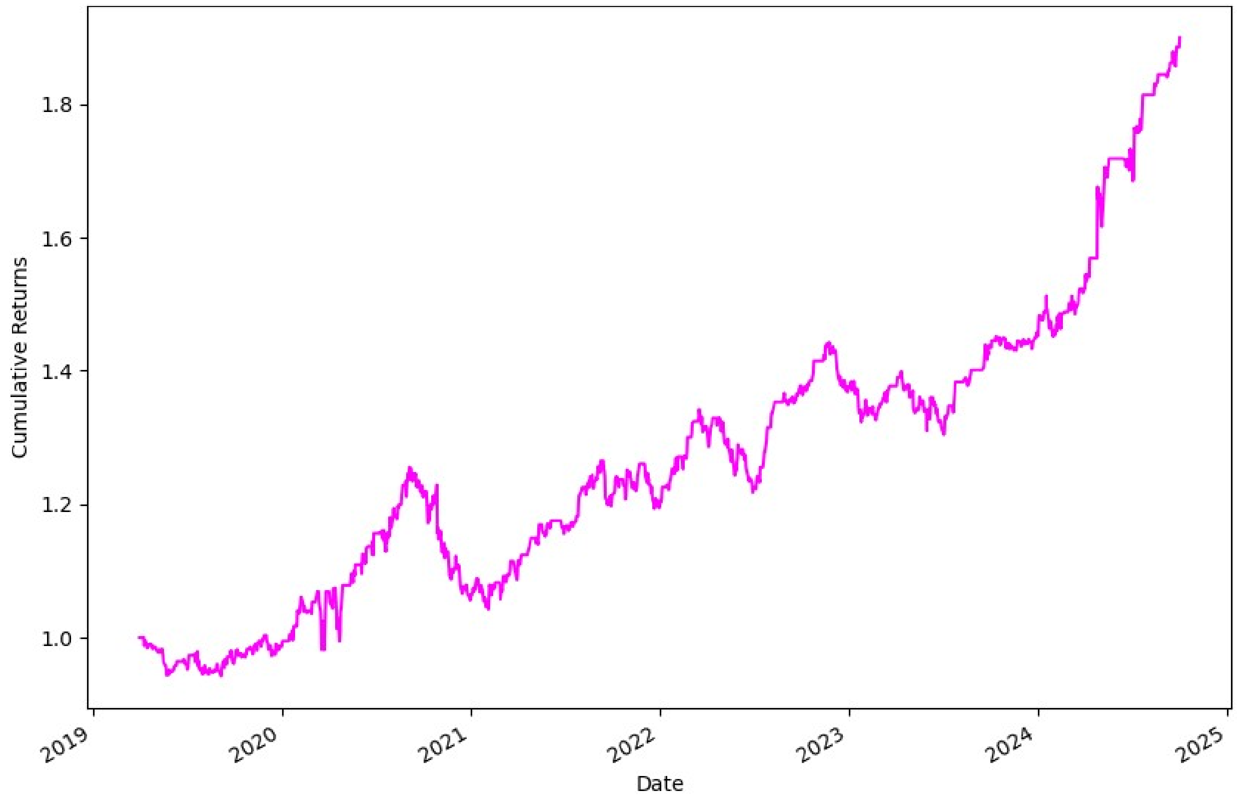
Interpretation:

This is a classic **mean reversion strategy**. When the spread deviates from its mean, the strategy bets that it will revert, generating profits as the prices of the two stocks converge.

6. Evaluating the Strategy: Cumulative Returns and Sharpe Ratio

We now calculate the **cumulative returns** and **Sharpe ratio** of the strategy to evaluate its performance.

```
# Calculating percentage change and cumulative returns
df['percentage_change'] = (df.spread - df.spread.shift(1)) / (hr *
df.HDFCBANK + df.KOTAKBANK)
df['strategy_returns'] = df.positions.shift(1) * df.percentage_change
df['cumulative_returns'] = (df.strategy_returns + 1).cumprod()

# Plotting cumulative returns
df.cumulative_returns.plot(label='Cumulative Returns', figsize=(10,7),
color='magenta')
plt.xlabel('Date')
plt.ylabel('Cumulative Returns')
plt.show()

# Calculating Sharpe Ratio
sharpe_ratio = (df['strategy_returns'].mean() * 252) /
(df['strategy_returns'].std() * np.sqrt(252))
print(f'Sharpe Ratio: {np.round(sharpe_ratio, 3)}')
```



```
Sharpe Ratio: 1.048
```

Interpretation:

- The **cumulative returns** plot shows the total returns generated by the strategy over time.
- The **Sharpe ratio** of **1.048** indicates that the strategy has a reasonable risk-adjusted return. A Sharpe ratio above 1 is generally considered good, so a value close to 1 suggests moderate performance.

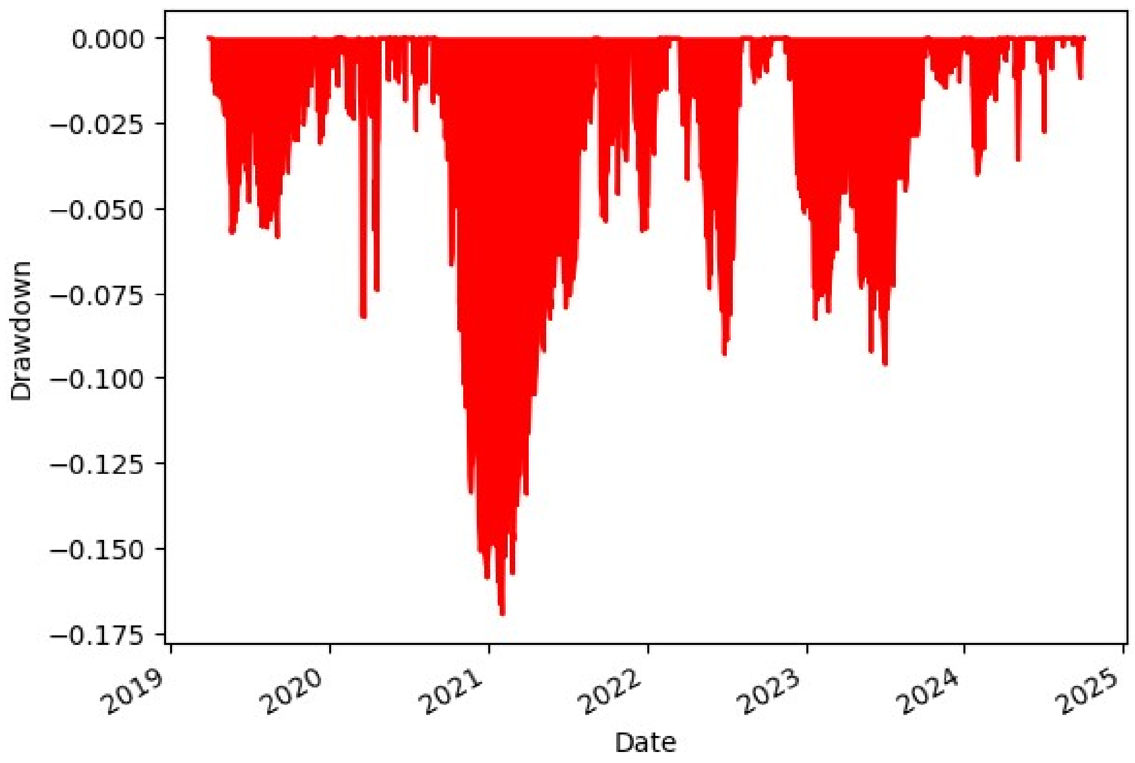
---

7. Drawdown Analysis

Drawdown represents the peak-to-trough decline in the portfolio's value, which is essential for understanding the risk of large losses.

```
def calc_drawdown(cum_rets):
    running_max = np.maximum.accumulate(cum_rets.dropna())
    drawdown = (cum_rets) / running_max - 1
    return drawdown

def plot_drawdown(drawdown):
    drawdown.plot(color='r')
    plt.fill_between(drawdown.index, drawdown, color='red')
    plt.ylabel('Drawdown')
    plt.show()

drawdown_strategy = calc_drawdown(df.cumulative_returns)
print("Maximum drawdown: {:.2f}%".format(drawdown_strategy.min() *
100))
plot_drawdown(drawdown_strategy)
```

```
Maximum drawdown: -16.95%
```



Interpretation:

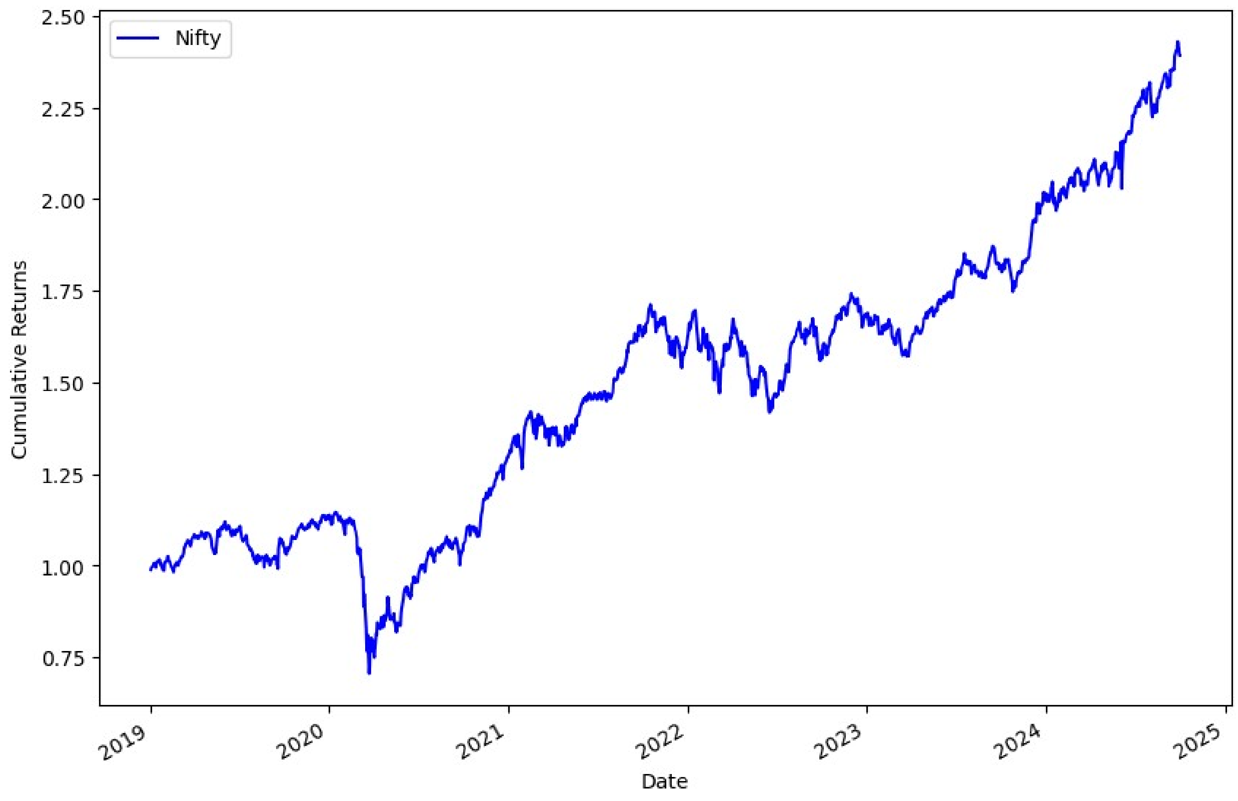
- The **maximum drawdown** of **-16.95%** shows the largest loss experienced by the strategy from its peak value. This is relatively low compared to traditional directional strategies, highlighting the lower-risk nature of pairs trading.

---

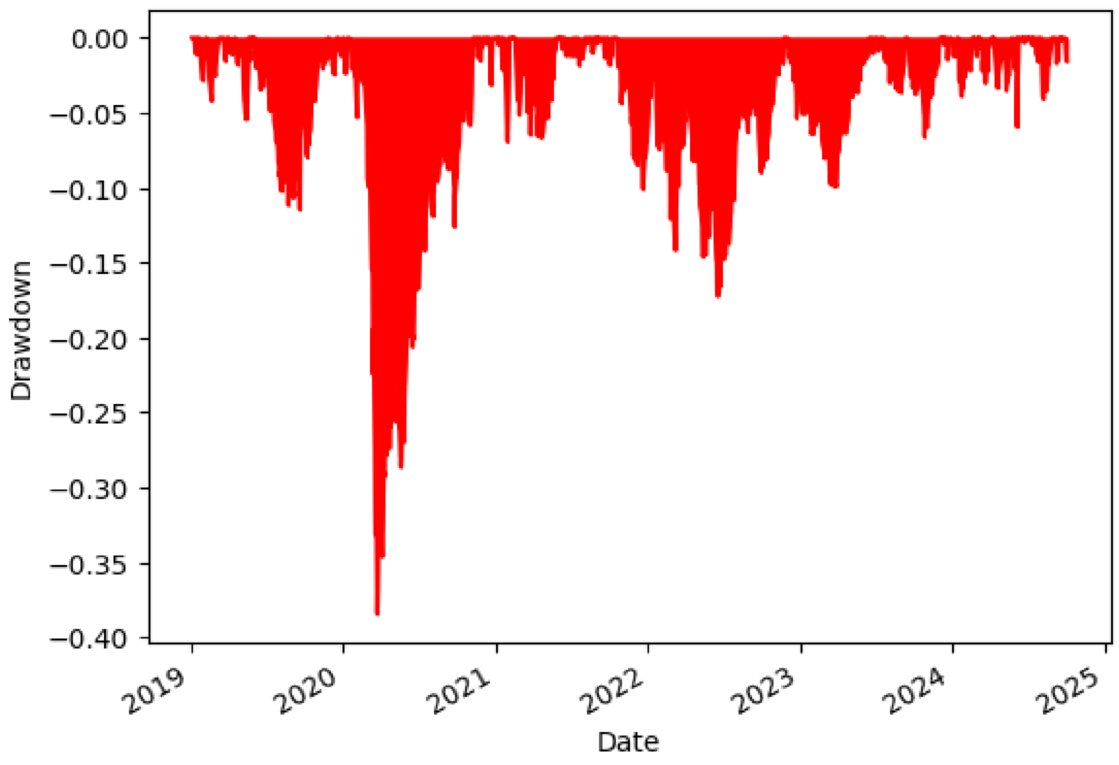
8. Strategy Comparison with NIFTY

To evaluate the relative performance of our strategy, we compare it with the **Nifty 50 index**.

```
# Downloading Nifty data and calculate cumulative returns
end = '2024-12-31'
start = '2019-1-1'

start_date = pd.to_datetime(start)
end_date = pd.to_datetime(end)

nifty = yf.download('^NSEI', start=start_date, end=end_date)['Adj
Close']
nifty_cum_rets = (nifty.pct_change().dropna() + 1).cumprod()

# Plotting cumulative returns of Nifty and strategy
nifty_cum_rets.plot(label='Nifty', figsize=(10,7), color='blue')
plt.xlabel('Date')
plt.ylabel('Cumulative Returns')
plt.legend()
plt.show()

# Calculating Nifty drawdown
drawdown_nifty = calc_drawdown(nifty_cum_rets)
print("Maximum Nifty drawdown: {:.2f}%".format(drawdown_nifty.min() *
100))
plot_drawdown(drawdown_nifty)

 [*********************100%***********************]  1 of 1 completed
```



```
Maximum Nifty drawdown: -38.44%
```



Interpretation:

- Nifty’s **maximum drawdown** of **-38.44%** during the same period highlights the advantage of market-neutral strategies like pairs trading, which aim to mitigate large losses.

The pairs trading strategy between **KOTAKBANK** and **HDFCBANK** demonstrates how a **statistical arbitrage** approach can yield attractive, market-neutral returns. By identifying mean-reverting relationships through cointegration and trading on divergences in the price spread, this strategy provides a robust way to generate profits while minimizing exposure to broader market movements.

However, like any strategy, it is crucial to **manage risks** and perform regular backtesting to ensure that the relationships between the selected pairs remain stable over time.

4. Risks of Statistical Arbitrage

While statistical arbitrage strategies like pairs trading can be profitable, they are not without risks:

A. Model Risk

Statistical models rely on historical data, and there is no guarantee that past relationships will hold in the future. For example, if two historically correlated stocks decouple due to a fundamental change (such as a merger or change in business strategy), the strategy may lead to significant losses.

B. Execution Risk

In high-frequency environments, the speed of execution is critical. Delays in executing trades may cause the strategy to miss opportunities or execute trades at suboptimal prices.

C. Liquidity Risk

If the assets being traded have low liquidity, it may be difficult to enter or exit positions without significantly impacting the price.

D. Market Regime Changes

Statistical arbitrage assumes that market conditions remain relatively stable. Sudden changes in volatility, liquidity, or market direction (e.g., during financial crises) can break down the relationships that the models rely on.

Conclusion

Statistical arbitrage is a powerful strategy that leverages statistical techniques to exploit short-term price inefficiencies between related financial instruments. One of the most popular forms of statistical arbitrage is **pairs trading**, which involves identifying cointegrated pairs of stocks and trading on the divergence from their long-term relationship.

While statistical arbitrage can generate consistent returns in stable markets, it carries significant risks, especially during periods of market stress or when underlying statistical relationships break down. Therefore, careful risk management, robust model testing, and efficient execution are critical components of a successful stat arb strategy.