Pairs Trading Project User Manual

Submitted By TUSHAR PANDEY (21113170)

Email: t_pandey@ce.iitr.ac.in

1. Introduction

Pairs trading is a market-neutral trading strategy that involves taking opposite positions in two correlated securities to exploit pricing inefficiencies. This manual guides you through implementing a pairs trading strategy using historical stock data.

2. Setup and Installation

2.1 Prerequisites

Ensure you have Python installed. The required libraries for this project are:

- pandas
- yfinance
- matplotlib
- seaborn
- numpy
- scikit-learn

2.2 Installing Required Libraries

You can install the required libraries using pip:

pip install pandas yfinance matplotlib seaborn numpy scikit-learn

3. Data Retrieval

3.1 Define Stock Tickers

Define the list of stock tickers you want to analyze. For example I have used:

```
tickers = ['AAPL', 'ADBE', 'ORCL', 'EBAY', 'MSFT', 'QCOM', 'HPQ', 'JNPR',
'AMD', 'IBM', 'SPY']
```

3.2 Download Historical Data

Use the yfinance library to download historical stock data:

```
# Download historical data for the stocks
data = yf.download(tickers, start="2018-01-01", end="2022-01-01")['Adj
Close']
```

4. Data Preparation

4.1 Define Specific Tickers for Analysis

Select the specific tickers you want to analyze in pairs e.g. I choose Adobe and Microsoft:

tickers = ["ADBE", "MSFT"]

4.2 Calculate Daily Returns

Calculate daily returns for the selected tickers:

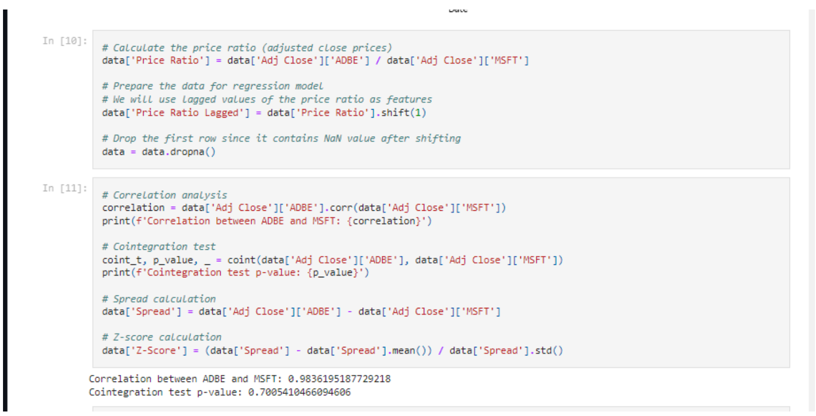
Calculating Returns

In [17]:
```
# Calculate returns
data['Return'] = data['Adj Close']['ADBE'].pct_change() - data['Adj Close']['MSFT'].pct_change()

# Calculate strategy returns
data['Strategy Return'] = data['Return'] * data['Position'].shift(1)

# Calculate cumulative returns
data['Cumulative Return'] = (1 + data['Return']).cumprod()
data['Cumulative Strategy Return'] = (1 + data['Strategy Return']).cumprod()
```

In [18]:
```
# Plot cumulative returns
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Cumulative Return'], label='Market Return', color='blue')
plt.plot(data.index, data['Cumulative Strategy Return'], label='Strategy Return', color='red')
plt.title('Cumulative Returns: Market vs. Strategy')
plt.xlabel('Date')
plt.ylabel('Cumulative Return')
plt.legend()
plt.grid(True)
plt.show()

# Print final cumulative returns
market_return_percentage = data['Cumulative Return'].iloc[-1] * 100
strategy_return_percentage = data['Cumulative Strategy Return'].iloc[-1] * 100
print(f'Final Market Return: {-100+market_return_percentage:.2f}%')
print(f'Final Strategy Return: {-100+strategy_return_percentage:.2f}%')
```

Cumulative Returns: Market vs. Strategy

4.3 Calculate the Spread and Z-scores



In [10]:
```
# Calculate the price ratio (adjusted close prices)
data['Price Ratio'] = data['Adj Close']['ADBE'] / data['Adj Close']['MSFT']

# Prepare the data for regression model
# We will use lagged values of the price ratio as features
data['Price Ratio Lagged'] = data['Price Ratio'].shift(1)

# Drop the first row since it contains NaN value after shifting
data = data.dropna()
```

In [11]:
```
# Correlation analysis
correlation = data['Adj Close']['ADBE'].corr(data['Adj Close']['MSFT'])
print(f'Correlation between ADBE and MSFT: {correlation}')

# Cointegration test
coint_t, p_value, _ = coint(data['Adj Close']['ADBE'], data['Adj Close']['MSFT'])
print(f'Cointegration test p-value: {p_value}')

# Spread calculation
data['Spread'] = data['Adj Close']['ADBE'] - data['Adj Close']['MSFT']

# Z-score calculation
data['Z-Score'] = (data['Spread'] - data['Spread'].mean()) / data['Spread'].std()
```

```
Correlation between ADBE and MSFT: 0.9836195187729218
Cointegration test p-value: 0.7005410466094606
```

5. Signal Generation and Returns

5.1 Calculating the Z-Score and Generate Buy/Sell Signals

In [15]:

```
# Define buy and sell signals
data['Signal'] = 0
data.loc[data['Z-Score'] > 1.0, 'Signal'] = -1  # Sell signal
data.loc[data['Z-Score'] < 0.5, 'Signal'] = 1  # Buy signal

# Exiting positions
data['Exit Signal'] = 0
data.loc[(data['Z-Score'] > -0.2) & (data['Z-Score'] < 0.2), 'Exit Signal'] = 1

# Combine signals
data['Position'] = data['Signal']
data['Position'] = data['Position'].replace(to_replace=0, method='ffill')

# Reset position to 0 on exit signal
data.loc[data['Exit Signal'] == 1, 'Position'] = 0
```

In [16]:

```
# Plot buy and sell signals
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Spread'], label='Spread', color='blue')
plt.scatter(data.index[data['Signal'] == 1], data['Spread'][data['Signal'] == 1], label='Buy Signal', marker='^', color='
plt.scatter(data.index[data['Signal'] == -1], data['Spread'][data['Signal'] == -1], label='Sell Signal', marker='v', colo
plt.title('Buy and Sell Signals on Spread')
plt.xlabel('Date')
plt.ylabel('Spread')
plt.legend()
plt.grid(True)
plt.show()

# Plot Z-score with buy and sell signals
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Z-Score'], label='Z-Score', color='blue')
plt.axhline(0, color='black', linestyle='--')
plt.axhline(1.1, color='blue', linestyle='--')
plt.axhline(-1.1, color='blue', linestyle='--')
plt.scatter(data.index[data['Signal'] == 1], data['Z-Score'][data['Signal'] == 1], label='Buy Signal', marker='^', color=
plt.scatter(data.index[data['Signal'] == -1], data['Z-Score'][data['Signal'] == -1], label='Sell Signal', marker='v', col
plt.title('Z-Score with Buy and Sell Signals')
plt.xlabel('Date')
plt.ylabel('Z-Score')
plt.legend()
plt.grid(True)
plt.show()
```

5.2 Calculating Returns

Calculating Returns

In [17]:

```
# Calculate returns
data['Return'] = data['Adj Close']['ADBE'].pct_change() - data['Adj Close']['MSFT'].pct_change()

# Calculate strategy returns
data['Strategy Return'] = data['Return'] * data['Position'].shift(1)

# Calculate cumulative returns
data['Cumulative Return'] = (1 + data['Return']).cumprod()
data['Cumulative Strategy Return'] = (1 + data['Strategy Return']).cumprod()
```

In [18]:

```
# Plot cumulative returns
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Cumulative Return'], label='Market Return', color='blue')
plt.plot(data.index, data['Cumulative Strategy Return'], label='Strategy Return', color='red')
plt.title('Cumulative Returns: Market vs. Strategy')
plt.xlabel('Date')
plt.ylabel('Cumulative Return')
plt.legend()
plt.grid(True)
plt.show()

# Print final cumulative returns
market_return_percentage = data['Cumulative Return'].iloc[-1] * 100
strategy_return_percentage = data['Cumulative Strategy Return'].iloc[-1] * 100
print(f'Final Market Return: {-100+market_return_percentage:.2f}%')
print(f'Final Strategy Return: {-100+strategy_return_percentage:.2f}%')
```

6. Backtesting

6.1 Selecting the time frame

We now select duration for which we have to test the model, we choose 01-01-2015 to 01-01-2016

In [23]:

```python
# Load historical data for ADBE and MSFT
tickers = ['ADBE', 'MSFT']
data = yf.download(tickers, start='2015-01-01', end='2016-01-01')

# Calculate the price ratio (or spread) between ADBE and MSFT
data = data.dropna()
data['Ratio'] = data['Adj Close']['ADBE'] / data['Adj Close']['MSFT']

# Compute rolling mean and standard deviation of the ratio (spread)
lookback_period = 50
data['Rolling Mean'] = data['Ratio'].rolling(window=lookback_period).mean()
data['Rolling Std'] = data['Ratio'].rolling(window=lookback_period).std()

# Calculate Z-score of the spread
data['Z-Score'] = (data['Ratio'] - data['Rolling Mean']) / data['Rolling Std']
```

6.2 Repeat step 4 & 5

7. References

1. Pairs Trade: Definition, How Strategy Works, and Example

2. Pairs Trading: Performance of a Relative Value Arbitrage Rule

3. https://medium.com/aimonks/pairs-trading-using-machine-learning-for-the-selection-of-pairs-24920cbed1b6