Stock Correlation Analysis Project

Overview

This Python tool analyzes the historical relationship between two stocks to identify correlation patterns. It calculates correlation coefficients, generates statistical metrics, and creates detailed visualizations - then exports everything to Excel with comprehensive charts for analysis.

How It Works

Getting Started

The script starts by pulling 5 years of historical closing price data from Yahoo Finance for the specified stocks (Maharashtra Scooters and Bajaj Holdings). This timeframe provides enough data points to identify meaningful correlation patterns.

Correlation Analysis

For the two stocks, we calculate the Pearson correlation coefficient which measures how strongly they move together:

$$\text{Correlation} = \Sigma[(X_i - X\blacksquare)(Y_i - \blacksquare)] / \sqrt{[\Sigma(X_i - X\blacksquare)^2 \times \Sigma(Y_i - \blacksquare)^2]}$$

The correlation coefficient ranges from -1 to +1. For example, a correlation of 0.85 means the stocks have a strong positive relationship - when one goes up, the other typically does too.

Statistical Calculations

For each stock, the system calculates key metrics:

- **Mean:** Average closing price over the period
- **Median:** Middle value in the sorted price dataset
- **Standard Deviation:** Measure of price volatility
- **Min/Max:** Lowest and highest prices reached
- **Current Price:** Most recent closing price

Daily Returns Calculation

Daily percentage returns show how much the stock moved each day:

$$\text{Daily Return} = (\text{Price}_{\text{today}} - \text{Price}_{\text{yesterday}}) / \text{Price}_{\text{yesterday}} \times 100$$

Example: If a stock closes at ■150 today and was ■145 yesterday:
(150 - 145) / 145 × 100 = 3.45% return

Normalized Performance

To compare stocks with different price ranges, we normalize prices to a base of 100:

```
Normalized Value = (Current Price / Starting Price) × 100
```

This lets you see relative performance. If Stock A is at 150 and Stock B is at 120, Stock A has gained 50% while Stock B has gained 20% from the starting point.

Rolling Correlation

The script calculates a 30-day rolling correlation to show how the relationship changes over time. This uses a sliding window that recalculates correlation for each 30-day period, revealing if the stocks are becoming more or less correlated.

Output

Excel File

The exported Excel workbook contains four sheets with comprehensive data:

- **Stock Prices:** Daily closing prices for both stocks over 5 years
- **Correlation Matrix:** Full correlation matrix showing relationship strength
- **Statistics:** Mean, median, standard deviation, min, max, and current price for each stock
- **Summary:** Metadata including date range and number of data points analyzed

Visualization Charts

Six detailed charts provide visual insights:

- **Stock Prices Over Time:** Line chart showing price movements for both stocks
- **Normalized Performance:** Relative performance comparison starting at base 100
- **Correlation Heatmap:** Color-coded visualization of correlation strength
- **Daily Returns Distribution:** Histogram showing frequency of different return percentages
- **Price Relationship Scatter:** Scatter plot with correlation coefficient displayed
- **Rolling Correlation:** 30-day moving correlation showing trend changes

Understanding Correlation Values

The script automatically interprets correlation strength:

- **> 0.7:** Strong positive correlation (stocks move together)
- **0.3 to 0.7:** Moderate positive correlation
- **-0.3 to 0.3:** Weak or no correlation
- **-0.7 to -0.3:** Moderate negative correlation
- **< -0.7:** Strong negative correlation (stocks move in opposite directions)

Main Challenges

Data Retrieval Issues

Yahoo Finance API sometimes returns incomplete data or fails for certain ticker symbols. Resolved by implementing comprehensive error handling with try-except blocks around all data fetching operations. The script validates data before proceeding with calculations and provides clear error messages.

Handling Empty or Invalid Data

When one or both tickers return no data, correlation calculations fail. Fixed by checking data availability before correlation computation and providing informative warnings to users about which ticker failed and why.

Ticker Symbol Format

NSE stocks require specific formatting with .NS suffix (e.g., RELIANCE.NS). Users often forget this suffix leading to failed data retrieval. Added validation checks and helpful error messages explaining the correct format.

Chart Rendering Problems

matplotlib sometimes fails to display charts on systems without proper GUI backends. Solved by ensuring charts save to file regardless of display capability, so analysis results are preserved even if visual display fails.

Excel Export Complications

Mixed data types and missing values caused Excel export failures. Fixed by cleaning all DataFrames before export - converting data types consistently, filling NaN values appropriately, and validating sheet names for special characters.

Insufficient Data Points

Calculating rolling correlation requires enough historical data. When requesting shorter periods, 30-day windows might not have sufficient data. Added data length validation and informative messages when windows can't be calculated.

Memory Management

Loading 5 years of minute-level data for multiple stocks could cause memory issues. Optimized by fetching only daily closing prices (not full OHLCV data) and using pandas efficient data structures to minimize memory footprint.

Technical Implementation

The project uses Python 3.8+ with key libraries: yfinance for data retrieval, pandas for data manipulation, matplotlib and seaborn for visualizations, and openpyxl for Excel export. All calculations use vectorized pandas operations for performance. Files are timestamped to prevent overwriting previous analyses.