April 27th, 2024
Aayush Khandekar

FE520 - Introduction to Python for Financial Applications

Project Title: stockinsights

Link: https://github.com/AayushKhandekar/stockinsights

Introduction:
Conducting financial analysis and visualization is an important part of financial projects. This package titled 'stockinsights' aims at aiding the process of financial analysis by offering various tools for analyzing and visualizing stock data. The package supports common stock market metrics such as logarithmic returns, daily returns, rolling volatility, Sharpe ratio, price-to-earnings (P/E) ratio, price-to-sales (P/S) ratio, and the moving average convergence divergence (MACD). It also offers visualization functions for comparing stock prices against the S&P 500 index and charting stock data using line plots or candlestick charts.

Functions Implemented:

1. visualize(ticker, start = None, end = None, frequency = "daily", chart = "line")
   The visualize function downloads and visualizes stock data for a given ticker symbol over a specified period. It supports line plots and can be extended to candlestick charts. The frequency of the data can be specified as daily or weekly.

2. visualize_against_sp500(ticker, start = None, end = None, frequency = "daily")
   The visualize_against_sp500 function visualizes a given stock's adjusted closing price against the S&P 500 index over a specified date range. It supports daily and weekly frequencies.

3. log_returns(ticker, start = None, end = None, frequency = "daily")
   The log_returns function calculates the logarithmic returns for a specified stock over a given date range and frequency. It uses daily or weekly data to compute the log returns, allowing for financial analysis and other time series operations.

4. daily_returns(ticker, start = None, end = None, frequency = "daily")
   The daily_returns function calculates the daily returns (percentage change) for a given stock over a specified date range and frequency. It uses daily or weekly adjusted closing prices to compute the returns.

5. rolling_volatility(ticker, start = None, end = None, frequency = "daily", window = 20)
   The rolling_volatility function calculates the rolling standard deviation (volatility) of daily returns for a given stock over a specified period and frequency. It supports daily and weekly frequencies, with a customizable window size for the rolling calculation.

6. sharpe_ratio(ticker, start = None, end = None, risk_free_rate = 0.01, frequency = 'daily')
   The sharpe_ratio function calculates the Sharpe ratio for a given stock over a specified date range and frequency. It measures the risk-adjusted return of an investment, comparing the expected excess return to the standard deviation of returns.

7. pe_ratio(ticker, period = "1d")
   The pe_ratio function calculates the Price/Earnings (P/E) ratio for a given stock. The P/E ratio is a measure of the stock's price relative to its earnings per share.

8. ps_ratio(ticker, start = None, end = None)
   The ps_ratio function calculates the Price/Sales (P/S) ratio for a given stock over a specified period. The P/S ratio measures the stock's price relative to its revenue.

9. macd(ticker, start=None, end=None)
   The macd function calculates and visualizes the Moving Average Convergence Divergence (MACD) for a given stock over a specified period. It plots the MACD line and the Signal line.

Design Decisions:
The package was developed while keeping user flexibility and simplicity in mind. The following key design decisions were made:

1. Functionality: The package addresses common financial analysis needs, offering a range of tools for analyzing stock data. The package includes functions for visualizing stock prices, calculating various metrics, and comparing stock performance against broader market indices.

2. Flexibility: The package provides flexibility in terms of frequency (daily or weekly) and date ranges. This allows users to analyze stock data over custom periods, accommodating different analysis scenarios.

3. Visualization: The package uses Matplotlib as a package for developing stock visualizations. The functions for visualizing stock data and comparing against the S&P 500 index enable users to gain insights into market trends.

4. Error Handling: Error handling was implemented to ensure that users errors when encountering issues, such as invalid inputs, missing data, or computation errors.

Challenges Faced:
While developing the package, several challenges emerged and were encountered. They are listed below:

1. Data Availability: Data retrieval relied on the Yahoo Finance (yfinance) library, which occasionally experiences disruptions or data inconsistencies. These were handled appropriately with the help of error handling by providing informative errors.

2. Input Validation: Ensuring valid inputs for all functions was a significant challenge. Date ranges, frequency options, and other user inputs required validation to prevent unexpected errors during function execution.

3. Charting Limitations: While the package supports line and candlestick charts, the implementation of candlestick charts using MPLFinance proved more complex than anticipated. This led to temporary restrictions on candlestick charting while exploring alternative solutions.

4. Data Processing: Calculating metrics like Sharpe ratio, rolling volatility, and logarithmic returns required careful handling of data, particularly when dealing with missing or inconsistent data points.

Conclusion:
Overall, the financial analysis and visualization package provides a comprehensive set of tools for analyzing and visualizing stock data. Despite challenges which are mentioned above, the package achieves its goals of offering flexible and simple functions for financial analysis. Future improvements could focus on developing more functions related to financial metrics, enhancing candlestick charting, and improving error handling supported by the package.