Algorithmic trading - Shaurya Tiwari

Implemented three algorithmic trading strategies for the S&P 500 using Python, leveraging libraries such as pandas, NumPy, and Matplotlib. These strategies encompassed the creation of an Equal-weight index fund, Momentum investing, and Value investing, aiming to enhance portfolio performance by applying a variety of financial theories.

1.Equal-Weight S&P 500 Index Fund Script

Introduction

This document outlines a Python script designed to facilitate the creation of an equal-weight S&P 500 index fund. The S&P 500 serves as a leading indicator for U.S. equities, representing the market's best performance. Unlike traditional funds that weight investments by market capitalization, this script aims to distribute investments evenly across all constituents of the S&P 500.

Getting Started

Library Imports

The script utilizes several open-source software libraries for numerical computing, data manipulation, HTTP requests, Excel file creation, and mathematical functions. These include:

- NumPy
- Pandas
- Requests
- XlsxWriter
- Math

Importing Stock List

The S&P 500 constituents are dynamic and change over time. For this exercise, a static list of stocks is used, which should be downloaded and placed in a designated directory for access by the script.

Acquiring API Token

Market data is fetched using the IEX Cloud API, requiring an API token for access. It's recommended to store this token securely in a `secrets.py` file within the same directory as the script.

Script Functionality

Making API Calls

The script structures API calls to IEX Cloud to retrieve necessary stock information, such as market capitalization and stock price.

Parsing API Data

Data retrieved from API calls is parsed to extract relevant information, preparing it for further processing and analysis.

Data Organization

A pandas DataFrame is employed to organize stock data, acting like a spreadsheet to store ticker symbols, stock prices, market capitalizations, and the calculated number of shares to buy for each stock.

Batch API Calls

To enhance performance, the script utilizes batch API calls, reducing the total number of HTTP requests. This approach not only speeds up the data retrieval process but may also reduce costs associated with API usage.

Calculating Shares to Buy

The script calculates the number of shares to buy for each stock based on the total value of the user's investment portfolio. This ensures an equal-weight distribution of the investment across all S&P 500 constituents.

Output Formatting

The final step involves formatting and exporting the calculated data to an Excel file using the XlsxWriter library. This file provides a clear and accessible recommendation for trades, tailored to the size of the user's investment portfolio.

Conclusion

This Python script offers a comprehensive solution for investors looking to create an equal-weight S&P 500 index fund. From fetching up-to-date stock information to calculating investment distribution and exporting actionable insights, the script simplifies the investment process, making it accessible to a wide range of users, including recruiters seeking to understand the project's functionality and potential impact.

2. Quantitative Momentum Strategy

Introduction

This document details a Python script for implementing a quantitative momentum investing strategy. Momentum investing involves investing in stocks that have shown an increase in price the most. The strategy focuses on selecting the 50 stocks with the highest price momentum from the S&P 500 and calculates recommended trades for an equal-weight portfolio of these stocks.

Getting Started

Library Imports

The script relies on several key Python libraries for data manipulation, HTTP requests, numerical computing, and Excel file writing. These libraries include Pandas, Numpy, Requests, XlsxWriter, Math, and Scipy for statistical functions.

Importing Stock List and API Token

To fetch stock data, the script uses a list of S&P 500 stocks and an API token from IEX Cloud. Ensure you have the `sp_500_stocks.csv` file and a valid IEX Cloud API token stored securely.

Script Functionality

Fetching Momentum Data

The first step involves making API calls to IEX Cloud to retrieve one-year price returns for each stock. This data is crucial for identifying high-momentum stocks.

Building the DataFrame

The script constructs a DataFrame to hold stock tickers, prices, one-year price returns, and a placeholder for the number of shares to buy. This DataFrame is populated by making batch API calls to improve efficiency, fetching data for up to 100 stocks per request.

Identifying High-Momentum Stocks

To focus on high-momentum stocks, the script sorts the DataFrame by one-year price return in descending order and selects the top 50 stocks. This subset represents the high-momentum stocks for the investment strategy.

Calculating Shares to Buy

The script calculates the number of shares to buy for each stock in the high-momentum portfolio based on the total value of the user's investment portfolio. This ensures an equal-weight investment across all selected stocks.

Enhancing the Momentum Strategy

To refine the strategy, the script also considers the quality of momentum by looking at price returns over different time frames: one month, three months, six months, and one year. Stocks are scored based on their momentum percentile ranks across these time frames, and an overall High-Quality Momentum (HQM) Score is calculated.

Selecting the Best Momentum Stocks

The top 50 stocks with the highest HQM Scores are selected for the investment portfolio. This approach aims to identify stocks with "slow and steady" momentum, indicative of high-quality momentum.

Output Formatting and Excel Export

Finally, the script formats the output and exports the investment recommendations to an Excel file using the XlsxWriter library. The Excel file includes detailed information on each selected stock, such as price, momentum scores, and the calculated number of shares to buy.

Conclusion

This Python script provides a comprehensive approach to implementing a quantitative momentum investing strategy. By selecting stocks with high momentum and considering the quality of momentum, the strategy aims to construct a balanced, equal-weight portfolio of high-momentum stocks from the S&P 500, providing clear and actionable investment recommendations through a well-organized Excel output.

3. Quantitative Value Strategy

Introduction

This documentation outlines the Python script for a quantitative value investing strategy. This strategy focuses on selecting the 50 stocks with the best value metrics from the S&P 500, aiming for an equal-weight portfolio of these stocks.

Library Imports

The script utilizes several Python libraries for numerical computing, data manipulation, HTTP requests, and Excel file creation. These libraries include NumPy, Pandas, Requests, XlsxWriter, Math, and SciPy's stats module.

Importing Stock List & API Token

The script begins by importing a list of S&P 500 stocks and an API token for accessing financial data through the IEX Cloud API. This step is crucial for retrieving up-to-date stock information and valuation metrics.

Making the First API Call

The initial API call fetches the price-to-earnings (P/E) ratio for a given stock symbol. This ratio is a fundamental component of the value strategy, as it helps identify undervalued stocks.

Executing Batch API Calls & Building the DataFrame

To efficiently gather data for all S&P 500 stocks, the script executes batch API calls. This approach minimizes the number of requests sent and speeds up data retrieval. The information collected includes not just the P/E ratio but also price-to-book (P/B), price-to-sales (P/S), enterprise value to EBITDA (EV/EBITDA), and enterprise value to gross profit (EV/GP) ratios.

Handling Missing Data

Not all stocks have complete data for the required metrics. The script handles missing data by either dropping those stocks from consideration or imputing missing values with averages. This step ensures the robustness of the strategy.

Calculating Value Percentiles

For each stock, the script calculates percentile scores across the different valuation metrics. These percentiles help in ranking stocks based on their relative value.

Selecting the 50 Best Value Stocks

With all stocks scored, the script selects the top 50 stocks that exhibit the best value characteristics according to their RV (Robust Value) scores. This subset represents the portfolio's constituents.

Calculating the Number of Shares to Buy

Based on the user's input for the total portfolio value, the script calculates the number of shares to buy for each stock. This calculation assumes an equal-weight investment across the 50 selected stocks.

Formatting Excel Output

Using the XlsxWriter library, the script generates an Excel file detailing the selected stocks, their valuation metrics, and the calculated number of shares to buy. This file serves as a practical guide for executing the investment strategy.

Conclusion

The quantitative value strategy script offers a systematic approach to value investing by leveraging financial ratios and market data. By focusing on undervalued stocks, it aims to construct a portfolio positioned for superior returns relative to the market.

Why I failed and what I learnt?

The Python scripts for creating an equal-weight S&P 500 index fund, implementing a quantitative momentum strategy, and a quantitative value strategy offer innovative approaches to portfolio management and stock selection. However, several potential drawbacks may limit their usefulness in practical, day-to-day trading strategies:

1. Static and Infrequent Rebalancing

- Equal-Weight S&P 500 Index Fund: This strategy requires periodic rebalancing to maintain equal weightings, which can be costly and labor-intensive. Daily market fluctuations can quickly alter the equal distribution, making the strategy less effective in rapidly changing markets.
- Quantitative Momentum and Value Strategies: These strategies select stocks based on past performance metrics (momentum) or valuation metrics (value), which can change over time. The reliance on historical data without frequent updates may miss short-term market movements and opportunities.

2. Overlooking Market Capitalization

- Equal-Weight Strategy: By ignoring market cap, this strategy may over-expose the portfolio to smaller companies, which can be riskier than their larger counterparts. This increased volatility may not be suitable for all investors, especially in turbulent market conditions.

3. High Transaction Costs

- All Strategies: Regularly rebalancing to maintain equal weights or to adjust the portfolio based on new momentum or value data can result in high transaction costs. These costs can eat into the overall returns, especially in strategies requiring frequent trades.

4. Ignoring Market Conditions and Macro Factors

- Momentum and Value Strategies: These strategies do not consider current market conditions, economic indicators, or macroeconomic factors. Stocks with high momentum or low valuation metrics may not perform well in all market conditions, making these strategies less effective during market downturns or periods of high volatility.

5. Data and Execution Delay

- All Strategies: The effectiveness of these strategies is heavily reliant on the timeliness and accuracy of data from the IEX Cloud API. Delays in data retrieval or execution can lead to missed opportunities or investing based on outdated information.

6. Risk of Value Traps and Momentum Crashes

- Value Strategy: There's a risk of investing in "value traps," companies that appear undervalued but have underlying problems that could lead to poor performance.
- Momentum Strategy: Similarly, stocks with high momentum can suddenly reverse, leading to "momentum crashes" where the strategy suffers significant losses.

7. Limited to S&P 500 Stocks

- All Strategies: By focusing solely on S&P 500 companies, these strategies may miss opportunities in smaller, potentially more dynamic stocks outside the index. This limitation can restrict portfolio diversification and potential returns.

8. Complexity and Accessibility

- All Strategies: While the scripts aim to simplify the investment process, the complexity of managing API calls, data parsing, and the need for regular monitoring may be daunting for some investors, especially those without a technical background.

In conclusion, while these Python scripts provide structured approaches to investing, their practical application in day-to-day trading may be hindered by issues related to market dynamics, costs, data reliance, and the inherent risks of the strategies. Investors should consider these limitations and adapt their investment approach to align with their risk