Backtest Engine for assessment of Algorithmic Trading Strategies

Introduction

Validating a trading strategy's performance through historical simulation is essential in gauging its potential profitability. This project set out to build a backtest engine in Python that processes 20 years of S&P500 closing price data and outputs key return and risk metrics to assess a strategy's viability for investors. The model leverages *pandas* and *numpy* for efficient data manipulation, accelerating computation across the dataset whilst ensuring accuracy of results through improved clarity.

Although the framework has been designed to be flexible for any strategy, this analysis focuses on the 50/200-day simple moving average (SMA) crossover – a trend-based method that generates signals when the short-term average crosses the long-term average, indicating potential momentum shifts.

Hypothesis

- SMA strategy will possess a higher Sharpe ratio due to signals to sell at downward trend swings, reducing volatility in portfolio value.
- Maximum drawdown will be less for SMA due to death cross signal at downward trajectories.
- Pure returns will be greater for the B&H strategy due to the overall stable, upwards trending nature of the S&P500. The SMA strategy will only signal to buy when the momentum is firmly positive, causing initial returns of a market recovery to be missed.

Key Findings

- Negligible Difference in returns between B&H and SMA strategies (~8.5% annualised)
- SMA achieved a slightly higher Sharpe Ratio of 0.451 compared to 0.445 of B&H, indicating marginally better returns per unit risk.
- SMA strategy experienced ***significantly reduced Max. Drawdown*** compared to B&H (6% improvement), indicating superior ability to limit losses.

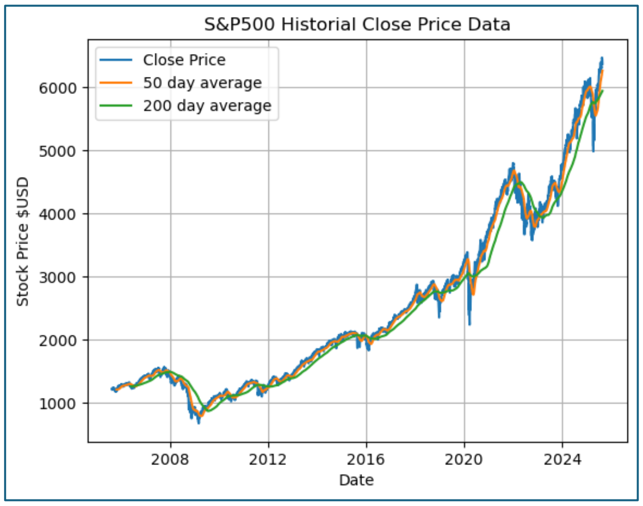

Figure 1 - S&P500 Equity Curve with moving averages (2005-2025)

Conclusion

The outputs of the backtest engine confirmed the primary hypothesis of superior risk mitigation of the SMA crossover strategy but refuted the claim of lower returns. This could be attributed to the strategy's preservation of capital during bear markets (2008 financial crisis, 2022 Covid crash), offsetting the missed opportunities for returns at momentum swings. Future development of the program could incorporate transaction costs and dynamic position-sizing at signals based on liquid capital, significantly improving the simulation's realism and real-world applicability.

Code Logic

1. Data Acquisition and Processing - obtain S&P500 data from *yfinance* API by specifying desired parameters and clean to preferred numerical indexing format.
2. Calculate Moving Averages – initialize new columns in dataframe, vectorized calculations of moving average using *rolling.mean()* pandas function.
3. Identify Crossovers – check whether crossover takes place at each daily interval using *for* and *if* loops.
4. Define Trading Strategies – for SMA, crossover will signal buy/sell of 5 shares under the condition of sufficient liquid funds. *for* loops to calculate portfolio value and drawdown each day for risk metric calculations. Functions take initial invested amount and outputs ROI, Sharpe Ratio, MDD. Initial code omits transaction costs.
5. Implementation of backtest strategy – based on input parameters, *print* displays output message
6. Results visualization – use of matplotlib to plot equity curve and drawdown and verify signal locations.

Note: The code structure intentionally utilises manual loops for core financial processes—such as signal generation and portfolio accounting—to ensure clarity and strengthen understanding of the mechanics involved. For larger-scale data analysis, these could be replaced with vectorized operations in *pandas* and *numpy* to improve computational efficiency and reduce cost.