Trading Backtester

Executive summary

We built a daily multi-asset backtester in Python. It downloads adjusted OHLCV(Open, High, Low, Close, Volume) data from Yahoo Finance, saves it into per-ticker CSV files, applies pluggable trading strategies, and turns every signal into an explicit trade with costs.

The part we care about most is not the final return. It is making the project somewhat accurate where no future prices leak into a particular day’s signal, a warm cache makes zero network requests after initial download, and the dashboard shows the detailed assumptions beside the result.

Reference run used throughout

Input	Value	Explanation
Ticker universe	AAPL, AMZN, META, MSFT, NVDA + SPY	Five stocks plus S&P 500 as the benchmark
Data range	2016-01-04 to 2026-05-15	Daily adjusted prices used in the saved run
Starting capital	$100,000	Split equally across six independent accounts
Execution clock	Close(T) → open(T+1)	Signals use yesterday-style information, then fill next day
Base costs	5 bps + 2 bps slippage	A small one-way cost is charged on turnover

1. How one daily bar becomes a trade

A backtest is a replay of decisions. For each ticker, the program reads one daily bar at a time, calculates indicators from prices already seen, decides whether the target position is long, flat, or short, and then executes that decision on the next available open.

The delay is important since ff a crossover becomes visible after the close on Tuesday, the program cannot pretend it bought at Tuesday's close. It enters at Wednesday's open instead. When the final date arrives, the engine explicitly liquidates the remaining position so the ending wealth is not an unsold mark.

The information cutoff

Step	What the backtester uses	Why it matters
1	Close(T) and earlier rows	Indicators never read tomorrow's close
2	Open(T+1) for a new position	The signal has a realistic execution delay
3	Close(T) for a same-day exit	The ledger records a concrete exit price
4	Last-date close for liquidation	Final wealth includes every open position

The indicator toolbox

The project includes simple moving averages (SMA), exponential moving averages (EMA), RSI, MACD, and ATR. They are backward-looking calculations. The built-in strategies output discrete positions in {-1, 0, +1}; the default configuration stays long-or-flat, while optional shorts come with a clear warning that borrow costs are not modeled.

The core goal is that every number shown in the report should be traceable to a price row, a signal, a fill, or a documented assumption.

2. How the data cache works

Yahoo Finance is useful for learning but downloading data every time would make the experiment slow and hard to reproduce. The data layer therefore treats the per-ticker CSV cache as the source for normal runs. The dashboard reads the cache and it only contacts Yahoo after the user clicks the refresh button.

Cache event	What happens	Why we built it this way
No CSV yet	Download the configured date range and write a tidy file	A first run can create a complete local dataset
Fresh cache	Load the CSV and make zero network requests	The same experiment is fast and repeatable
Stale cache	Download only dates after the last cached date, drop duplicates, append atomically	Refreshes do not re-download ten years of history making it more optimized
Bad response or bad values	Retry with backoff, validate OHLCV, and never overwrite a good cache	A failed refresh cannot corrupt the research data

The cache schema is simple: date, open, high, low, close, volume, and retrieval timestamp. Validation rejects empty frames, duplicate dates, NaNs, non-positive prices, and negative volume. These checks help ensure that the cached data is valid and suitable for repeatable backtests.

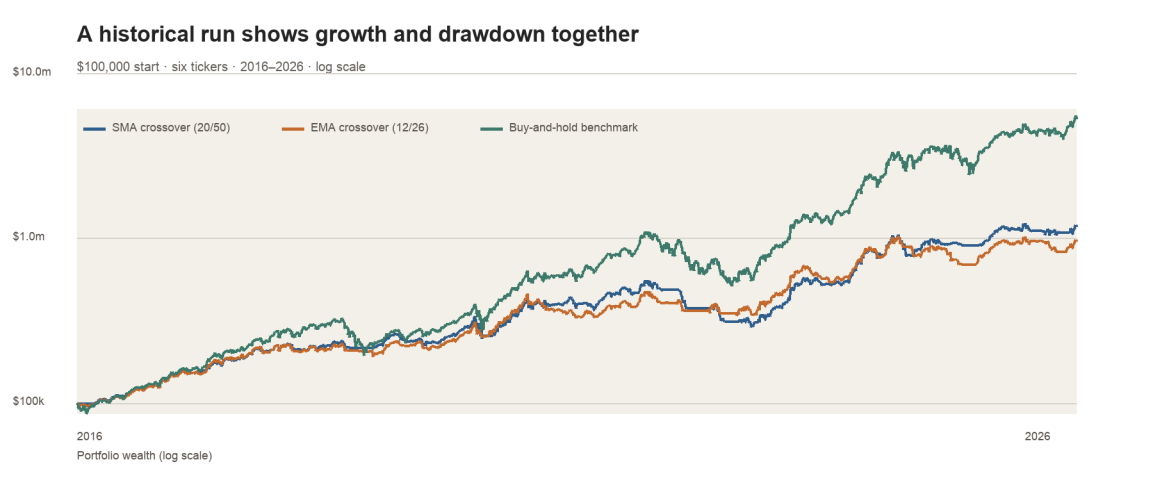

Figure 1. A saved historical run: the buy-and-hold benchmark grows faster, while the two crossover strategies follow different drawdown paths.

3. How the strategies are compared

The project does not treat one strategy as “the best”, rather It runs the same universe, starting capital, execution clock, and cost assumptions through different rule sets so the comparison is fair.

Strategy	Explanation	Why it is included
SMA crossover (20/50)	Long when the 20-day average is above the 50-day average	A slower trend-following baseline
EMA crossover (12/26)	Long when the faster exponential average is above the slower one	A more reactive trend-following comparison
Buy-and-hold benchmark	Stay invested in the pooled universe	A reference for what passive exposure did
RSI / MACD / ATR / custom	Additional built-ins and validated uploads	Makes the engine genuinely pluggable

What the reference run says

Path	Ending wealth	Total return	CAGR	Sharpe	Max drawdown	Trades
SMA 20/50 (5 bps)	$1.18m	1,076.6%	26.9%	1.11	-48.3%	316
EMA 12/26 (5 bps)	$957k	857.0%	24.4%	1.09	-32.7%	468
Buy-and-hold benchmark	$5.33m	5,233.9%	46.9%	1.25	-53.4%	12

These figures should not be interpreted as an investment recommendation. They illustrate the methodological trade-off where the benchmark captured more of the long-term market appreciation, whereas the crossover strategies maintained lower market exposure and exhibited different drawdown profiles. A backtest is most informative when it presents these trade-offs alongside the headline return rather than reducing the analysis to a single performance figure.

4. How portfolio and risk metrics fit together

Each ticker receives an equal share of the starting capital and runs as an independent cash account. The pooled portfolio is the exact sum of those account paths. It is not created by interleaving trades from different stocks and compounding them in an accidental order.

The report records more than just the return, it also stores volatility, Sharpe ratio, Sortino ratio, maximum drawdown and etc. Cost scenarios are run separately so we can see how sensitive the conclusion is to friction.

One stock can disproportionately dominate the portfolio

Ticker	SMA total return	SMA max drawdown	Trades	Exposure
AAPL	301.1%	-28.3%	31	63.8%
AMZN	135.7%	-55.5%	26	65.6%
META	597.6%	-36.1%	24	64.0%
MSFT	238.5%	-40.2%	27	68.4%
NVDA	5,033.8%	-64.9%	27	68.6%
SPY	152.7%	-28.3%	23	72.2%

The per-ticker breakdown is important for interpreting the pooled result. Aggregate performance may be disproportionately influenced by a single asset with an unusually strong historical period, even when that asset also experienced substantial drawdown. The dashboard allows the pooled portfolio and individual asset paths to be examined together.

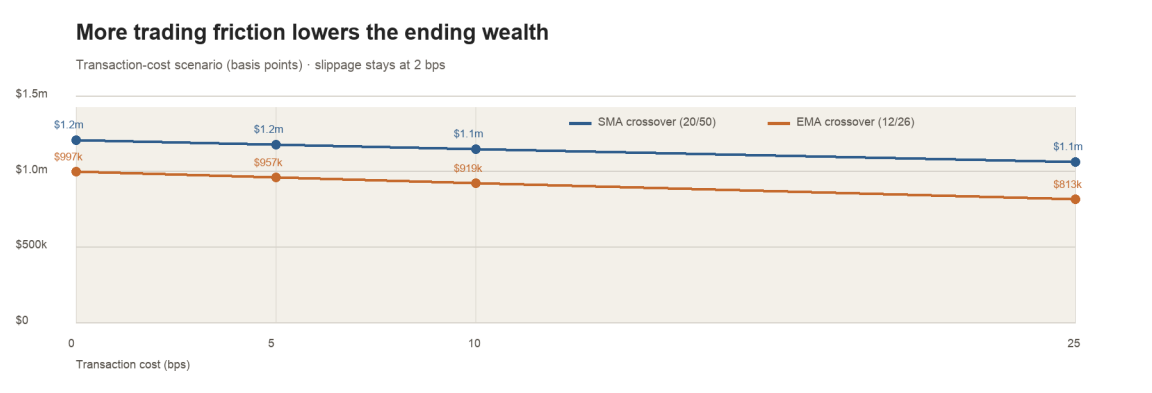

Figure 2. More transaction cost lowers ending wealth in both crossover runs. The ranking does not change, but the gap matters.

5. How the code and dashboard fit together

The project is split into small modules so the financial logic can be tested without running Streamlit. The dashboard consumes an immutable BacktestReport and it does not silently re-run the model when a chart is drawn.

File	Job	Why we kept it separate
config.py	Load and validate the YAML configuration	A run has one clear, checked source of assumptions
data.py	Yahoo download, validation, cache, incremental refresh	The core data contract is testable on its own
indicators.py	SMA, EMA, RSI, MACD, ATR	Each calculation stays small and backward-looking
strategies.py	Built-ins and registry for pluggable rules	New strategies do not require rewriting the engine
backtest.py / portfolio.py	Fills, costs, ledger, independent accounts, pooled path	Execution accounting stays separate from presentation
metrics.py / reporting.py	Risk statistics and frozen report objects	The UI receives a reproducible result, not loose globals
dashboard/app.py / strategy_loader.py	Interactive controls and six-gate custom upload validation	Users can explore safely without importing arbitrary code

The flow

User Inputs	Cache	Backtest	Show/Visualize
Tickers, dates, strategy, costs	Validated daily CSVs	Signals → next-open fills → metrics	Charts, ledger, diagnostics

Strengths

- **Efficient data reuse:** Once data is saved locally, normal runs read the CSV files without making new Yahoo requests. Tests confirm this behavior.
- **Safe cache updates:** New data is checked for valid dates, prices, volume, and columns before replacing the existing file. This prevents failed downloads from corrupting the cache.
- **Traceable simulation results:** Trading costs are applied to each transaction, open positions are closed at the end, and every trade is saved in a ledger for review.
- **Safer custom strategies:** Uploaded strategies must use correct format, return valid positions, avoid future data and unsafe imports, and have enough historical data before they can run.

Future Improvements

- **Test across multiple periods:** Use walk-forward and out-of-sample testing to check whether the system works beyond one historical period.
- **Improve historical data quality:** Use point-in-time data and include companies that were later delisted to reduce historical bias.
- **Model realistic trading conditions:** Add borrow costs, bid-ask spreads, market impact, liquidity limits and position-size limits.

6. How I checked whether the answers make sense

A backtest can look correct even when something is wrong on the back end. Multiple test have been written to check the data collection, cache, indicators, strategies, trade timing, portfolio calculations, reports, custom strategy uploads, and dashboard.

Check	What it means	Why it matters
Warm cache	After data is downloaded once, the program reuses the saved local files instead of downloading everything again.	Makes the project faster and more repeatable.
No look-ahead	The program cannot use future prices to make earlier decisions,	Prevents the simulation from cheating.
Next-open execution	A signal made after today's close is executed at the next day's open	Represents a more realistic trading timeline.
Ledger checks	Trade prices, quantities, fees, and costs all add up correctly.	Makes every simulation result traceable.
Cost checks	Adding higher trading costs should not improve the result.	Confirms that fees behave logically.
Portfolio checks	The total portfolio equals the combined value of the individual ticker accounts.	Confirms that the portfolio is being combined correctly.
Custom strategy checks	Uploaded strategies are tested before they are allowed to run.	Helps prevent invalid or unsafe code from affecting the simulation.

The tests support the implementation mechanics. They do not guarantee that the data is perfect or that historical results will repeat in the future.

7. Limitations, next steps, and what I learned

The model is useful for learning, but it also out parts of real trading that can have a large impact on results.

The biggest assumptions currently

- Yahoo Finance data is not point-in-time since prices can be revised and delisted names disappear.
- Daily bars hide intraday path, queue position, spread, market impact, and liquidity limits.
- Short positions do not include borrow costs, and the default strategies are long or flat.
- A historical backtest is not a forecast and is not investment advice.

A realistic improvement plan

Priority	Improvement	Why it would matter
1	Add walk-forward and out-of-sample evaluation	Reduce the chance that a rule only fits the past
2	Use point-in-time data and survivorship-aware universes	Make the information set closer to what was actually knowable
3	Add borrow, spread, impact, and capacity assumptions	Make net results less optimistic for real trading
4	Add strategy parameter sweeps with guardrails	Study stability instead of treating a single setting as definitive
5	Store data snapshots and run hashes	Make a result reproducible months later

Method Note

The numbers in this report come from the saved JSON reports under trading_backtester/data/results. The run used daily adjusted Yahoo Finance data from 2016-01-04 through 2026-05-15 for AAPL, AMZN, META, MSFT, NVDA, and SPY, with $100,000 starting capital, equal-weight allocation, next-open execution, same-day close exits, 5 bps transaction cost, and 2 bps slippage. Values are historical simulations for learning only, not live quotes or investment advice.