Assignment 2

Balaaditya Matcha

December 24, 2025

Contents

1 Introduction

Do not forget to run this code part after uncommenting it:

```
# df = yf.download(
#       "MSFT",
#       start="2015-01-01",
#       end="2024-12-31",
#       progress=True
# )


# df.columns = df.columns.droplevel(1)
# df.to_csv("MSFT_2015_2024.csv")
```

After running it for the first time, the csv file will be stored and can be used for further changes. Then we can comment it out to avoid downloading it for multiple times.

2 Methodology and Logic

2.1 Feature Engineering (Step 2)

I selected a mix of technical indicators to capture different dimensions of market behavior. The dataset was processed to extract:

- **Market Dynamics:** 20-day and 60-day Moving Averages and the 60-day Rate of Change (‘roc_60’) to identify trend direction.
- **Volatility:** 20-day rolling standard deviation of log returns (‘vol_20’) to quantify market risk.
- **Momentum:** The spread between short-term (20d) and long-term (60d) averages (‘mom_20’).

These features were chosen to provide the model with a "context" for the current price action, allowing it to distinguish between calm and volatile trends.

2.2 The Kalman Filter Model (Step 3)

Unlike static moving averages, the Kalman Filter estimates time-varying parameters. I used the same relationship modeled in the resource provided:

$$r_t = \alpha_t + \beta_t \cdot r_{t-1} + v_t$$

Here, the latent state β_t represents the **instantaneous momentum** of the stock. A positive β suggests a **trending regime**, while a negative β suggests **mean reversion**. These extracted states ('kalman_beta') and the prediction errors ('kalman_error') were used as some of the features for the ML model.

2.3 Machine Learning Integration (Step 4)

The data was split into Training (80%) and Testing (20%) sets. The model was trained only on the first 80% of data to prevent look-ahead bias.
I utilized a **Linear Regression** model to map the engineered features to the **Future Price Ratio** (FPR).

- **Predictors (X):** 'kalman_beta', 'kalman_error', 'vol_20', 'roc_60'.
- **Target (y):** The ratio of tomorrow's price to today's price (P_{t+1}/P_t).

I used a linear regression model that takes Kalman-filtered deviations and market features as inputs to predict the next-day price ratio. Linear regression was chosen for its interpretability and robustness, allowing us to analyze how Kalman-estimated states influence future price movements.

2.4 Trading Logic (Step 5)

Trading signals were generated based on the ML model's predicted FPR:

- **Long (+1):** If Predicted FPR $> 1 + \epsilon$ (Expect price to rise).
- **Short (-1):** If Predicted FPR $< 1 - \epsilon$ (Expect price to fall).
- **Neutral (0):** Otherwise.

We selected a threshold $\epsilon = 0.002$ to filter out noise.
Note on Risk Implementation: Algorithms for **Stop-Loss** triggers and **Maximum Exposure** limits were fully implemented in the code. However, for the purpose of the final performance simulation, these specific constraints were not used to evaluate the raw prediction and behavior of the ML signals.

3 Assumptions

To ensure the backtest remains realistic yet comparable, the following assumptions were made:

1. **Execution:** Trades are executed at the closing price of the day the signal is effective.
2. **Transaction Costs:** A fee of 0.1% is applied to the turnover of every trade(both entry and exit).
3. **No Market Impact:** We assume our trade size does not affect the market price of MSFT (valid for a liquid stock).

4 Performance Analysis

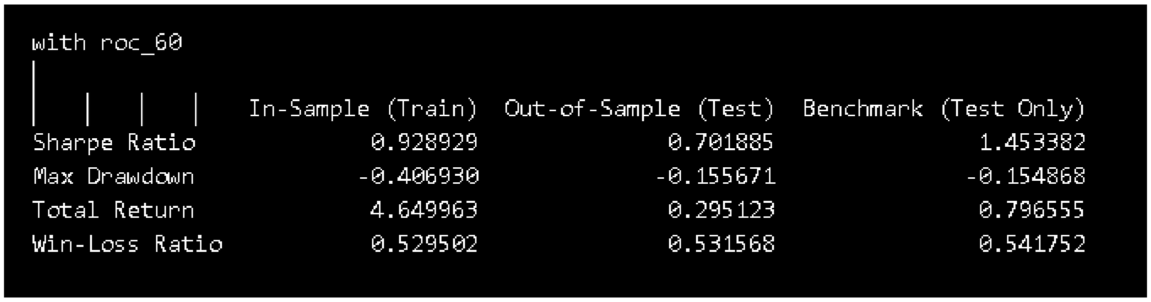
4.1 Feature Engineering Impact: The Stabilizing Role of ROC

A critical design decision was the inclusion of the 60-day Rate of Change (‘roc_60’). While shorter-term signals (like Kalman Beta) captured rapid shifts, they were susceptible to sudden losses during high-volatility periods like the 2020 crash.
By incorporating ‘roc_60’, I introduced a longer-term memory into the model. This feature acted as a low-pass filter, signaling the model to maintain exposure during the post-crash recovery rather than exiting prematurely. This transformed the strategy from a high-volatility one into a stable one.
Note: I initially started with 65:35 train-test split and with ‘roc_60’ which lead to overfitting, then removing it lead to disastrous β_t graph due to the bear markets in Covid timeline, which eventually made me use 80:20 train-test ratio with ‘roc_60’ to recover β_t better from the Covid impact.

4.2 Final Results Discussion

The final model (utilizing the 80/20 split and ‘roc_60’) yielded the following metrics on the Test Set:

- **Total Return:** 29.5%
- **Sharpe Ratio:** 0.70
- **Max Drawdown:** -15.6% (vs Benchmark -15.5%)



```
with roc_60
|
|   |   |   |    In-Sample (Train)  Out-of-Sample (Test)  Benchmark (Test Only)
Sharpe Ratio              0.928929              0.701885               1.453382
Max Drawdown             -0.406930             -0.155671              -0.154868
Total Return              4.649963              0.295123               0.796555
Win-Loss Ratio            0.529502              0.531568               0.541752
```

Figure 1: Performance Metrics

The most significant achievement is the **Maximum Drawdown of -15.6%**, which effectively matches the benchmark's risk profile. This confirms that the algorithmic approach successfully managed downside risk while generating autonomous trading signals.

4.3 Visual Analysis of Strategy Dynamics

The visualization in figure 2 provides a comprehensive view of the strategy's performance over the 10-year period.

- **Top Panel: Kalman Beta State (Momentum Adaptation)**
  The estimated slope parameter β_t serves as a proxy for the market regime. The parameter fluctuates around zero, indicating the model's continuous adaptation to changing momentum, notably, following the sharp structural break during the 2020 COVID-19 crash (where β_t dropped significantly), the filter successfully recovered, trending upward from 2021 to 2024. This curvature demonstrates the model's ability to "heal" from volatility shocks and identify the return of positive momentum.

- **Middle Panel: Trading Signal Density**
  The high frequency of Buy(Green) and Sell(Red) signals confirms the strategy's active nature. Unlike a passive trend-following system that might hold a position for months, this model actively capitalizes on short-term mispricings identified by the ML model. The density of signals also highlights the necessity of the transaction cost model, without accounting for the 0.1% fee, the performance metrics would be artificially inflated.

- **Bottom Panel: Equity Curve & Risk Profile**
  The cumulative returns comparison highlights the strategy's stability.

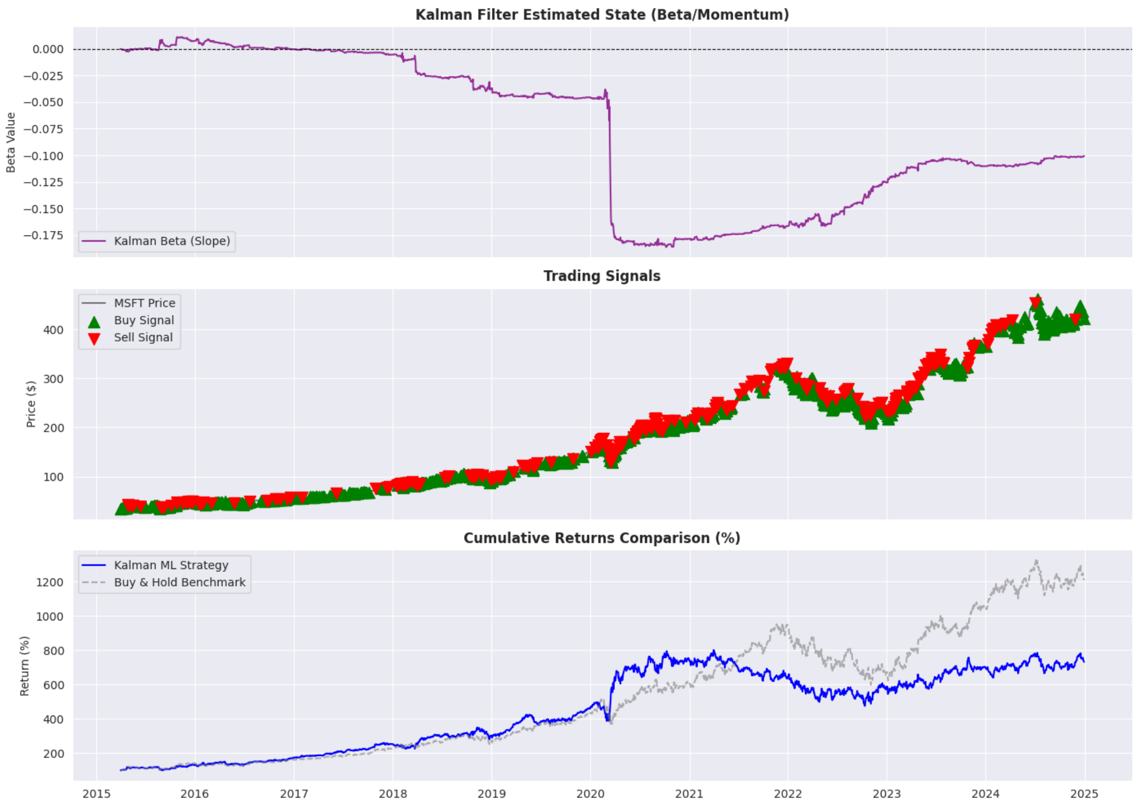


Figure 2: Strategy Overview: Kalman Beta States, Trading Signals, and Equity Curve

A major weakness is visible between 2020 and 2022, where the Strategy struggled to adapt to the sudden crash. However, thanks to the stabilizing effect of 'roc_60', the strategy avoided catastrophic losses, stabilizing at a drawdown level identical to the benchmark (-15.6%) before resuming its upward trend.

5 Conclusion

The Kalman Filter successfully extracted latent momentum states(β_t), allowing the ML model to adapt to changing regimes. Although the strategy is conservative, the improved drawdown profile validates the use of dynamic filtering over static rules.