Algorithmic Trading Strategy Report

1. Introduction  
This report explains, in simple and human language, how I built a basic but meaningful trading strategy using the Mock PS dataset from Ebullient Securities. My goal was not to create a highly complex machine learning model, but to build a clear and interpretable signal-generation pipeline that could be tested for profitability and stability without using any future data.

I worked using two Jupyter notebooks:  
- mock\_ps: This was my first practice notebook where I explored and experimented.  
- mock\_ps1: This is my final notebook where I implemented the actual full strategy.

2. Loading and Exploring the Data  
I first loaded the 'day0.csv' dataset into a Pandas DataFrame. The file had more than 22,000 rows and around 437 columns. The dataset included a 'Time' column and multiple technical features with prefixes like PB, VB, and BB. Based on their names, I assumed:  
- PB = Price-Based indicators  
- VB = Volume-Based indicators  
- BB = Order Book / Bid-Ask Based indicators  
I checked missing values, number of unique entries, and data types to understand the structure better. I noticed that many features had missing values at the start, which I assumed were due to rolling calculations like moving averages that need initial warm-up periods.

3. Data Cleaning and Preprocessing  
To prepare the data, I followed a structured cleaning approach:  
- I used forward fill to fill missing values and backward fill to handle gaps at the beginning.  
- I removed columns with very low variance because they don’t add useful information.  
- I applied MinMaxScaler to normalize PB, VB, and BB-related features into a 0–1 range.  
- I made sure there were no duplicate timestamps or rows.  
After these steps, the dataset was stable and consistent for generating signals.

4. Feature Engineering  
I engineered a few new meaningful features inspired by the ZeltaAutomations prep guide and basic market logic:  
- Entropy (on Volume-Based features): This helped measure how random or uncertain the volume activity was. Lower entropy means clearer market direction, while higher entropy indicates noise. For signal generation, I preferred low entropy to get clearer signals.  
- Asymmetry: This helped identify market imbalance. If buy-side strength was higher than sell-side or vice versa, it could show a directional pressure.  
- Z-score (row-wise): I standardized each tick across all features to detect unusual movements. A high positive z-score might indicate upward pressure, and a negative one might indicate downward movement.  
I applied a lag (shifted these features by one time step) to avoid using any future information for current signal generation.

5. Signal Generation Logic  
I designed a simple rule-based system using threshold conditions:  
- Long (Buy) Signal (+1): When entropy was low (indicating a clear situation), asymmetry showed strong buy-side bias, and z-score was significantly positive.  
- Short (Sell) Signal (-1): When entropy was low but asymmetry showed weakness on the buy side, and z-score was negative.  
- Hold (0): When none of the above conditions were met.  
I also shifted the signal column by 1 tick to avoid lookahead bias and ensure the signal is based only on past information.

6. Backtesting and Evaluation  
I backtested the signals using a simple assumption: enter at the current tick when a signal appears and exit after one tick. I calculated Profit and Loss (PnL) using cumulative returns.  
Key Results (Day 0):  
- Net Profit: ~2.07%  
- Sharpe Ratio: ~2.63, showing a good risk-adjusted return.  
- Max Drawdown: Around -0.32%, which is very low and indicates the strategy is stable.  
- Calmar Ratio: Approximately 6, meaning strong returns relative to drawdown.  
- Number of Trades: Fewer than 10 trades per day, which matches the challenge rules.

7. Challenges Faced  
During the process, I faced several issues:  
- Missing values at the start due to rolling computations.  
- Too many features (over 400), which made feature selection tricky.  
- Avoiding forward bias was a continuous concern.  
- Initially, returns were poor, so I needed to rethink the thresholds and engineer better custom features.

8. Summary and Conclusion  
This project helped me understand the full workflow of building a trading strategy: from raw data cleaning to feature engineering, signal generation, and backtesting. Even though I did not use a machine learning model here, the results were stable, understandable, and profitable. This version serves as a strong baseline that can be further improved later using more advanced techniques like decision trees or ensemble models.