Develop a Systematic 0-DTE Option Selling Strategy on SPX

Objective

Create a Python script that backtests and deploys a systematic 0-DTE (zero days to expiration) option selling strategy on the S&P 500 index (SPX).

Background

0-DTE option selling is a strategy that involves selling options on the day they expire. For SPX, these options expire at the market close. This strategy aims to profit from rapid time decay and mean reversion in short-term market movements.

Key Concepts

1. **0-DTE Options**: Options that expire on the same day they are traded.
2. **SPX (S&P 500 Index)**: A market-capitalization-weighted index of 500 leading U.S. publicly traded companies.
3. **Option Spreads**: Simultaneously buying and selling options at different strike prices to limit risk.
4. **VIX (Volatility Index)**: A measure of market expectations of near-term volatility conveyed by S&P 500 stock index option prices.

Formulas and Functions

1. **Expected Move Calculation**: $$ \text{Expected Move} = \text{VIX} \times 0.0065 \times \text{move\_adjustment} $$

   This formula estimates the expected price movement of SPX based on the VIX.

2. **Strike Price Selection**:

   - Call Short Strike: $$ \text{call\_short} = \text{open\_price} \times (1 + \text{expected\_move}) $$
   - Put Short Strike: $$ \text{put\_short} = \text{open\_price} \times (1 - \text{expected\_move}) $$

3. **Moving Average Calculations**:

   - 20-day MA: $$ \text{MA}_20 = \frac{\sum_{i=1}^{20} \text{price}_i}{20} $$
   - 50-day MA: $$ \text{MA}_50 = \frac{\sum_{i=1}^{50} \text{price}_i}{50} $$
   - 200-day MA: $$ \text{MA}_{200} = \frac{\sum_{i=1}^{200} \text{price}_i}{200} $$

4. **Profit/Loss Calculation**: For calls: $$ \text{PnL} = \min(\max(\text{short\_strike} - \text{trade\_price}, -\text{spread\_width}), 0) $$ For puts: $$ \text{PnL} = \min(\max(\text{trade\_price} - \text{short\_strike}, -\text{spread\_width}), 0) $$

Step-by-Step Guide

1. **Set Up Environment**:
    - Import necessary libraries (requests, pandas, numpy, matplotlib, datetime).
    - Set up API key for data retrieval.
2. **Define Parameters**:
    - Set strategy parameters (move_adjustment, spread_width).
    - Define trading dates, tickers, and trade time.
3. **Data Retrieval Function**:
    - Create a function to fetch historical price data from the Polygon API.
    - Implement error handling and rate limiting.
4. **Market Regime Analysis**:
    - Implement moving average calculations.
    - Create a function to determine market regime (bullish, bearish, neutral).
5. **Option Strike Selection**:
    - Develop logic to select appropriate option strikes based on expected move and market regime.
6. **Backtesting Loop**:
    - Iterate through trading dates.
    - For each date: a. Fetch required data (SPX, VIX, ETF prices). b. Calculate expected move and determine option strikes. c. Analyze market regime. d. Simulate option trade execution. e. Calculate and record trade results.
7. **Performance Analysis**:
    - Calculate overall strategy performance metrics (win rate, average P/L, max drawdown).
    - Visualize results using matplotlib.
8. **Risk Management**:
    - Implement position sizing based on account balance.
    - Add stop-loss and take-profit logic.
9. **Optimization (Optional)**:
    - Create functions to optimize strategy parameters using historical data.
10. **Documentation and Testing**:

- Add comprehensive comments explaining each section of the code.
- Implement unit tests for critical functions.

Deliverables

1. A Python script containing all the functionality described above.
2. A brief report (1-2 pages) explaining your implementation choices and analyzing the backtest results.