**Objective:**

* The user is developing a trading strategy focused on identifying potential uptrends by entering long positions at consecutive support levels formed by prior valleys.
* The strategy involves a willingness to accept small losses on initial long entries if the support levels fail, with the goal of capturing a significant profit when a support level eventually holds and an uptrend begins.
* The bot will use a CSV file to download historical data for initial setup, then switch to polling for real-time price updates.
* The bot will also identify and track resistance levels, formed by prior peaks.
* The process will be: 1) Fetch candles, 2) Calculate S/R with candles, 3) Trading bot.

**Deliverables:**

1. **Support and Resistance Calculation Code:**
   * A script or function that takes candlestick data (from a CSV file) as input and outputs the support and resistance levels, labeled with their respective tiers (S1, S2, S3... and R1, R2, R3...).
2. **Trading Bot:**
   * A bot that uses the calculated support and resistance levels to execute the trading strategy, including:
     + Identifying potential long entry points.
     + Placing buy orders.
     + Managing stop-loss and take-profit.
     + Determining when to switch from stop-loss to take-profit management.
3. **Backtesting Module:**
   * A module or script that takes historical candlestick data (preferably the same format as the input for the S/R calculation) and simulates the trading strategy's performance.
   * Output: A Pandas DataFrame containing detailed information, including candle data and the S/R levels present at the time of each candle:
     + "Time": The timestamp of the candle.
     + "Open": The opening price of the candle.
     + "High": The highest price of the candle.
     + "Low": The lowest price of the candle.
     + "Close": The closing price of the candle.
     + "Volume": The trading volume
     + "Symbol": The symbol of the traded asset
     + "SR\_Proximity": A label indicating the candle's relationship to support and resistance levels:
       - "No": The candle is not at any identified support or resistance level.
       - "S1": The candle's price is at the S1 support level.
       - "S2": The candle's price is at the S2 support level.
       - "S3": The candle's price is at the S3 support level.
       - ...
       - "S10": The candle's price is at the S10 support level.
       - "R1": The candle's price is at the R1 resistance level.
       - "R2": The candle's price is at the R2 resistance level.
       - "R3": The candle's price is at the R3 resistance level.
       - ...
       - "R10": The candle's price is at the R10 resistance level.

**Key Elements of the Strategy:**

1. **Support and Resistance Level Identification:**
   * The bot will download a CSV file containing historical price data **for initial setup.**
   * Identify the 10 most recent support levels from this data, based on prior swing lows (valleys).
   * Identify the 10 most recent resistance levels from this data, based on prior swing highs (peaks).
     + The identification of significant swing lows and swing highs will be done using the scipy.signal.argrelextrema function.
     + The algorithm will look for local minima (for support) and local maxima (for resistance) within a defined window of data points.
     + Additional filtering criteria, such as volume and recency, may be applied to select the most relevant levels.
   * The user is working on defining the criteria for selecting these "significant" swing lows and swing highs.
   * **After initial setup, the bot will use polling to get real-time price data.**
   * **The bot will manage support and resistance levels using a Pandas DataFrame. The initial S/R levels will be loaded from the CSV file into a Pandas DataFrame at the bot's startup, and this DataFrame will be updated in place as new price data is received and S/R levels shift.**
   * To efficiently manage and update S/R levels, the bot will use Pandas DataFrames.
     + The initial CSV data will be loaded into a Pandas DataFrame.
     + S/R levels will also be stored and updated within a Pandas DataFrame.
   * **It's important to note that support and resistance levels can occur at the same price level, even if separated by significant time and price action. The strategy will prioritize these price levels, rather than the age of the support/resistance.**
2. **Long Entries at Support:**
   * Place a long entry order when the price approaches each identified support level.
   * Each long entry is accompanied by a stop-loss order to limit potential losses.
3. **Uptrend Confirmation and Trade Management:**
   * The strategy uses a prior **LHL** pattern (low-high-low) to determine when to switch from stop-loss management to take-profit management.
     + If, after a long entry at a support level, the price moves \*above\* the midpoint of the **price variation** (the difference between the low and high) of the prior **LHL** pattern, the bot will switch to take-profit management.
     + If the price does not move above this midpoint, the bot will continue to manage the trade using the stop-loss.
   * Take-profit is calculated as a percentage decrease from the highest point reached \*after\* the price moves above the midpoint.
4. **Profit Target and Risk Management:**
   * Aim for a substantial profit on the winning long trade that catches the uptrend, intended to outweigh the accumulated losses from previous failed long entries.
   * The strategy involves accepting a series of small losses in pursuit of a larger gain.
   * Important to define the maximum number of consecutive losses and how to manage capital.

**Points of Refinement:**

* **Data Sources and Format:**
  + The bot will use a CSV file containing historical candlestick data.
  + The CSV file will contain the following columns: "Time", "Open", "High", "Low", "Close", "Volume", and "Symbol".
  + The CSV data will be read into a pandas DataFrame for efficient processing.
  + The bot uses the CCXT library to fetch candlestick data from the Bitget exchange.
  + The data fetching process involves symbol conversion (Bitget symbol to CCXT symbol) and includes error handling for network and exchange errors.
* **Criteria for Selecting Support and Resistance Levels:**
  + We need to define what makes a swing low "significant" for support and a swing high "significant" for resistance (e.g., depth of pullback, volume, time since formation).
  + This will be applied to both the CSV data and the real-time data.
    - The primary method for identifying these levels will be the scipy.signal.argrelextrema function.
    - The algorithm will look for local minima (for support) and local maxima (for resistance) within a defined window of data points.
  + A preliminary code will process the CSV file to generate a Pandas DataFrame with candle data and a ‘SR\_Proximity’ column, indicating the S/R status for each candle. This is for validating the S/R identification process.
* **Order Placement and Management:**
  + The bot uses the CCXT library to place long orders with specified margin and leverage.
  + The bot checks the current leverage of the trading account before placing an order, and sets it if necessary, with retry logic for network errors.
  + Long orders are placed using the create\_market\_buy\_order function in CCXT, with the order size calculated based on the specified margin (USDT) and leverage.
* **Closing Positions:**
  + The bot closes long positions using a direct V1 API call to Bitget's placeOrder endpoint with the close\_long side.
  + This involves generating a signature for the API request using the provided API key, secret key, and passphrase.
  + The closing order specifies the symbol, margin coin, size, side (close\_long), and order type (market).
* **Error Handling and State Management:**
  + The code includes error handling for various scenarios, such as insufficient funds, exchange errors, and network errors.
  + The bot logs these errors using the logging module.
  + The bot updates its internal state (using a dictionary-like structure) to track whether it's in a position (in\_position), the position size, and the entry price.
  + This state is updated after a successful order placement or closure.