**Crypto Trading Signals**

**Overview**

This project analyzes different trading strategies (such as Simple Moving Average (SMA), Exponential Moving Average (EMA), MACD, RSI, and Bollinger Bands) applied to historical Bitcoin (BTC) and Ethereum (ETH) data. The goal is to evaluate the performance of these strategies using metrics like Sharpe Ratio, Annualized Volatility, Annualized Return, and Strategy vs Market Returns. The Sharpe Ratio is the primary metric used to rank the effectiveness of each strategy.

**Strategies Implemented**

* **Moving Averages (SMA and EMA)**: These strategies compare short-term and long-term averages to determine buy/sell signals.
* **RSI (Relative Strength Index)**: Uses momentum indicators to assess whether an asset is overbought or oversold.
* **MACD (Moving Average Convergence Divergence)**: Identifies changes in the strength, direction, momentum, and duration of a trend.
* **Bollinger Bands**: Measures volatility and overbought/oversold conditions with standard deviations.

**Key Metrics for Evaluation**

* **Sharpe Ratio**: Measures risk-adjusted return.
* **Annualized Volatility**: Assesses how much the strategy's value fluctuates yearly.
* **Annualized Return**: The compounded annual return from the strategy.
* **Strategy vs Market Return**: Compares strategy returns to overall market performance.

**Prerequisites**

Before running the code, ensure that you have the following installed on your system:

1. **Python 3.x**: Ensure that you have Python installed on your machine. If you don't have it, download and install it from [Python's official website](https://www.python.org/downloads/).
2. **Anaconda (Optional)**: Anaconda simplifies package management and deployment. You can download it from [here](https://www.anaconda.com/products/individual).
3. **Required Libraries**:
   * We have used several Python libraries to perform data analysis and backtesting. These libraries should be installed first. You can install them using the following steps:

**Installing the Libraries**

If you're using **Anaconda**, open the Anaconda prompt and run the following commands:

conda install pandas numpy matplotlib seaborn scikit-learn mplfinance statsmodels finta backtrader imbalanced-learn

conda install -c conda-forge ta-lib

Alternatively, if you're using **pip**, you can install the required libraries using:

pip install pandas numpy matplotlib seaborn scikit-learn mplfinance statsmodels finta backtrader imbalanced-learn

pip install ta-lib

1. **TA-Lib Installation**: If you're having trouble installing TA-Lib, we suggest using the Anaconda distribution as it includes precompiled binaries. If you still face issues, refer to [TA-Lib installation guides](https://www.ta-lib.org/hdr_dw.html).

**Setup Instructions**

**Step 1: Prepare the Data**

You need to have **Bitcoin (BTC)** and **Ethereum (ETH)** historical data. The data should contain the following columns: open, high, low, close, and volume. Ideally, this data should be in a **CSV file** format.

1. Place your CSV file in the project folder.
2. Rename it to data.csv (or update the filename in the code accordingly).

Ensure that the dataset is correctly formatted, with the date in the index column for proper analysis.

**Step 2: Understanding the Code Structure**

The main components of the code are:

1. **Import Libraries**: The code starts by importing necessary Python libraries like pandas, numpy, and others for data manipulation, analysis, and visualization.
2. **Data Processing**: The fetch\_data\_for\_time\_frame function is used to divide the data into time intervals such as 15 minutes, 1 hour, 1 week, etc.
3. **Strategy Definitions**: Several classes implement trading strategies based on indicators like SMA, EMA, RSI, MACD, and Bollinger Bands.
4. **Backtesting**: The Backtest class from the backtesting library runs each strategy on the dataset and calculates the performance metrics.
5. **Metrics Calculation**: The code calculates the Sharpe Ratio, Annualized Volatility, Annualized Return, and compares the strategy performance with the market.
6. **Results Output**: The results are saved into an Excel file containing the top and bottom 10 strategies sorted by Sharpe Ratio.

**Step 3: Running the Code**

1. **Run the Script**: Open your terminal or Anaconda prompt, navigate to your project folder, and execute the script with the following command:
2. python main.py
3. **Enter the Timeframe**: The script will prompt you to enter the time frame for the backtesting. Example inputs:
   * 15min
   * 1hr
   * 4hr
   * 1 week

Choose a valid time frame from the options presented.

1. **Backtest Execution**: The script will loop over the selected time intervals and run each strategy. For each strategy, it will calculate the performance metrics and store the results.
2. **Results File**: Once the backtest is complete, the script will generate an Excel file named <strategy\_name>\_top\_bottom\_10\_metrics.xlsx that includes:
   * **Top 10 Strategies**: Sorted by Sharpe Ratio (highest).
   * **Bottom 10 Strategies**: Sorted by Sharpe Ratio (lowest).

This file will help you compare the performance of different strategies across various timeframes.

**Step 4: Viewing the Results**

1. Open the Excel file generated by the script. Each sheet will contain the following information:
   * **Start and End Date**: The time frame for the backtest.
   * **Sharpe Ratio**: The risk-adjusted return of the strategy.
   * **Annualized Volatility**: How much the strategy fluctuated over the year.
   * **Annualized Return**: The compound yearly return from the strategy.
   * **Strategy vs Market Return**: How the strategy compares to the market performance.

**Example Output**

After running the code, the output in your Excel file might look like:

| **Start Date** | **End Date** | **Sharpe Ratio** | **Annualized Return** | **Annualized Volatility** | **Strategy vs Market Return** |
| --- | --- | --- | --- | --- | --- |
| 2024-01-01 | 2024-02-01 | 1.25 | 15% | 30% | 2.5 |
| 2024-02-01 | 2024-03-01 | 0.80 | 10% | 40% | 1.8 |

**Notes:**

* **Sharpe Ratio**: The higher the Sharpe ratio, the better the strategy's risk-adjusted return.
* **Annualized Return**: The compounded return over a year.
* **Strategy vs Market Return**: A ratio showing how well the strategy outperformed or underperformed the market.

**Additional Notes**

* **Customizing Timeframe**: You can customize the time intervals in the code if needed by modifying the time\_frame\_mapping dictionary.
* **Handling Large Datasets**: For very large datasets, you may encounter memory issues. Consider using a smaller subset of data or optimize the code for memory handling.
* **Max Drawdown**: If the strategy experiences a large drawdown (more than 20%), it is skipped.

**Troubleshooting**

* **Error: TA-Lib Not Found**: Ensure that TA-Lib is installed correctly using conda or pip. If using pip, the installation can sometimes fail on Windows, and Anaconda is the recommended approach.
* **Empty Results**: If no trades are executed or no results are saved, check if the strategies' conditions (such as moving averages crossing) are met in the chosen timeframes.