Algorithmic Trading with python

(Project Arwen)

Overview

Algorithmic trading has become increasingly popular in recent years, as traders seek to use automation and machine learning to make informed trading decisions. Python is a popular programming language for building such trading programs due to its ease of use, powerful data analysis libraries, and flexibility.

In this project, we use Python to develop an algorithmic trading program that analyzes historical market prices and makes trade decisions based on predefined rules and strategies. By combining data analysis with algorithmic trading, we aim to create a program that can identify profitable trading opportunities and execute trades automatically, without human intervention.

This project will involve collecting and preprocessing historical market data, analyzing the data using data analysis libraries, developing trading strategies, and implementing algorithms to execute trades. Ultimately, our goal is to create a powerful tool for traders that can help them maximize returns and reduce risk through automated trading..

Goals

1. Collect and preprocess historical market data for relevant assets (Here we are using crypto currency price data).
2. Analyze the data using data analysis libraries such to identify patterns and trends.
3. Develop trading strategies that take into account various factors such as risk tolerance, market volatility, and potential returns.
4. Implement algorithms that can automatically execute trades via exchange API based on predefined rules and strategies.
5. Test and refine the algorithmic trading program on historical data to improve performance and accuracy.
6. Deploy the program in live trading environments and monitor its performance closely to ensure it continues to meet the desired objectives.
7. Evaluate the program's effectiveness in maximizing returns and reducing risk compared to traditional manual trading methods.

Specifications

The program will collect OHLC historical market data for a specific crypto currency from reputable sources such as Binance public API. Stores the data in a Pandas Dataframe. Here I have specifically built a unique technical indicator combining existing ATR (Average True Range), MACD (Moving Average Convergence Divergence) which represents market volatility and momentum respectively. The OHLC price data is used to calculate corresponding OHLC data representing Heiken-Ashi Candlestick so that it reduces trend noises in trends then this is used to calculate the indicator histograms, baselines and volatility based risk management positions. Binance WebSocket API is used to fetch and add new data to the dataframe.

What is the Algorithm followed?

Buy orders are placed when the histogram shows a concave up and the tangent is parallel to X-axis while potential Sell orders or exhaustion of uptrend is represented by histogram showing concave down and tangent is parallel to X-axis.

The Win-Rate is maximized by buying the Asset again with a capital twice that of the previous Buy order capital thereby reducing the average Buy price down to approximately 1/6th of the difference between previous higher Buy price and current lower Buy price. And the asset can be sold in the instant when the indicator shows the Sell signal or to maximize the profit if the price goes up more we can use the volatility based trailing stop-loss. When the indicator first shows the Sell signal we can check if the trailing stop-loss position is above the average buy price and if it is then we only Sell when the price closes below the stop-loss price. This is effective since if the uptrend continues the trailing stop-loss follows the price to a higher level maximizing profit.

Milestones

This method out performed the historical market by returning more than 348% in average on multiple timeframes (1m, 3m, 5m, 15m, 30m, 1h, 2h, 4h, 8h, 1d) and several crypto currencies (BTC, ETH, MATIC, DOT, SOL, LTC, XRP, AVAX, FTM, CHR, GALA, SAND, BNB) over a 120 days historical data.

RELIANCE INDS 4h chart

It's so good, so can we run it on raspberry pie 24/7 and eat pie's while it makes money for us?

The answer goes both ways. Technical analysis methods such as this work for now because Less the number of peoples discover it and use it, greater the chance of this to work for long. The markets always tend to counteract these effects. It's like a rebellious child the more you try to control it the more it slips out of hand.

This method seems very profitable and effective for higher time frames such as 1h, 2h, 4h for a trader while 1d time frame is a good choice for a long term investor in finding bottoms and tops.

But when you use it in lower time frames such as 1m, 3m, 5m, or 15m where it has wider effects from a major news or sentiment shift such as in a quick Sell of occurs the indicator might show more Buy orders as the price tanks and shows a false indication of recovery. It is fine If we were dollar cost averaging, but remember the method we are using. We buy every time the indicator shows a Buy signal and each time we buy the capital should be twice that of the previous. This quickly increases the amount of capital you need to invest like if start with a 10$ and on your 5th Buy the capital needed is 160$ (ie $10 \times 2^{(n-1)}$, where n is the order number. here in this case 5). This tends to happen when markets are reacting to some unexpected world events like a stock collapse, FOMC meetings, CPI reports.

Hmm...So what should we do about it?

One way to solve this and to build a really efficient and effective trading Bot is to make it intelligent rather than specifically telling it to follow a set of rules. We could make a much bigger data set that includes social media market sentiment scores, price data of correlated stocks, major world events, news feeds. Then designing a Reinforcement Learning Model and training it on these dataset with a reward function equivalent to the profit percentage returns. The aim of the agent will be to maximize its returns. So it will try to find patterns in data to decide when to Buy, when to Sell or how much to Buy and Sell. This way our agent will learn to become a really really good trader. Later this Neural Network Model can be used to infer live markets. The capability of the model will of course be based on the quality of data it's been trained on and inferred on. It is hard, but I don't believe it's impossible to build one such system to predict financial markets.

Author:

Aswanth C M

BSc. Physical Science with Computer Science

Github LinkedIn