# **Testing and analyzing** the viability of candlestick analysis by writing a crypto paper-trading bot using TA-lib

# HLR-

When I went looking for crypto trading bots, I found many tools for gathering market indicators, but I only found a few instances of trading bots calling out candlestick pattern analysis.

By writing a crypto trading bot and then testing it with both backtesting and walkforward testing, I hope to compare and contrast traditional techniques between following the weighted moving averages, standard technical analysis with candle pattern analysis.

By the end of writing the program I hope to achieve:

- a modifiable tool I can use for paper-trading and testing strategies

- a set of concrete analytics from which I can compare strategy outcomes, that I can modify as I see fit

- a working knowledge of tools that is directly transferable to Trality

- using PANDAS to analyze the data I generate to create objective and subjective data on the performance of the algorithms

# Tools I intend to use-

# TA-Lib

A commonly used library for gathering statistical data on real time markets. Several of the researched tools for writing bots have it as an available tool. It is also available in the service Trality. Also the tool I intend to use to gather candlestick patterns for analysis in conjunction with market indicators.

Why I intend to use this tool:

- it has been in active use since the 2000s

- it has a wrapper available for python

- it commonly appears in tutorials for creating intermediate trading bots

Negatives:

-while large parts of the library is written in C, I still couldn’t determine if the candlestick analysis was using any machine learning.

# Pandas

While PANDAS can also parse candlestick patterns, it has TA-lib as a requirement to do so. So far, I haven’t been able to find a well documented tool that identifies candlestick patterns without also requiring TA-lib. I intend to use Pandas in conjunction with TA-lib because of its extensive documentation, in contrast to other tools.

# **Finnhub**

Its not an exchange, but somewhere I can get candlestick OHLCV data for both stocks and Crypto, as I intend to backtest the algorithms on both.

- Is not a strict provider of data, but an aggregator.

- 20 years of financial data available

- Free service as well as paid

- Good documentation

- Can detect candlestick patterns

- social media sentiment analysis aggregator on keywords through API

- python wrapper available

# Bloomberg API

The one and only.

- paid service

- requires Bloomberg terminal access

- some of the most comprehensive data around

- somehow, the documentation is lacking.

- python wrapper available

# Tiingo API

- Slightly restrictive TOS in terms of data usage, extra licensing required over the paid service.

- both free tier and paid service is very up front and reasonably priced.

- API/python wrapper built with PANDAS in mind

- okay documentation

- active development, but small community

- python wrapper available, but it seems the documentation boils down to a page of examples

# Alphavantage

- no python wrapper, results in CSV/json. Not interested.

- ok documentation

- tools overshadowed by Finnhub

# Yahoo finance API

Was garbage, shutdown, unofficial.

# Other services-

# Trality

A cloud web-service tool used to write crypto trading bots using python. Users will be able to sell the use of their algorithms without providing source code. **TA-Lib is included as a preinstalled tool for analysis**. Trality can automatically backtest your trading algorithm. Users may also create crypto trading bots using their visual programming language. Realistically, its a very similar service to what I want to make, at least from the statistics gathered. Unfortunately, all the backtesting implementation is pretty black box from what I can see.

I might actually use this, but later. Anyway, they have a comprehensive tutorial for writing a basic, positive returning bot.  
https://www.trality.com/blog/strategy-building-basics-pt-i

- Great documentation

- Active public Discord server

- Free if you make under 5000 usd, then small subscription cost.

- Python, and seems to be used frequently

- Still must use an exchange provider to execute trades

- Cloud based

- Paying for favorable reviews. Take it as you will.

# Mudrex

A crypto bot platform

- Drag and drop interface

- int. prop. is maintained confedential.

- Preconfigured TA bots

- can make custom bots with TA indicators with the drag and drop

- Discord server more active than Trality

- Venture capital backed.

- can already curate prewritten bots, unlike Trality.

- IP costs for using someone else’s bot only apply when you have gains.

- web API has minimal documentation

- no tax assistance.

-Their evaluation criteria for a viable bot is:

* Max drawdown < 30% in a backtest over last 18 months
* Max 2 of months with 5% or more loss over the last 18 months
* No continuous three month period with a loss
* Max leverage of 3x
* On average 4 trades a month

# Coinrule

- free version available

- no premade bots, but there are premade strategies

- visual programming language based on TA triggers

# Coinbase/Binance

Exchanges I’ll be able to transition to to obtain both OHLCV and trading. I structured my initial code so that writing buy/sell functionality would be easy to change. If I eventually manage a non-dry run, I’d use these services because their API plays nicely with tax software.

- API calls for OHCLV

- API for buy/sell history simplifies doing taxes

- very little analytics.

- In terms of arbitrage between the exchanges, the potential for profit is negated by the slow API calls and even with that, the margins are extremely slim. Maybe there would be more potential for other services, but, not what I want to focus on.  
Source: https://medium.com/@gk\_/crypto-arbitrage-trading-on-binance-12dcb4d17b30

# Zenbot

A command line application for running and backtesting trading strategies written in javascript/node.js. From a test perspective it is interesting, but its offering of tools is too similar to PANDAS and TA-lib. It offers nice graphical analysis.

# Mitigating risks:

# Technical Analysis

The use of price history combined with statistical tools to make actionable decisions. This is the fundamental essence of algorithmic trading.

Overlap Studies   
Momentum Indicators   
Volume Indicators  
Volatility Indicators

**The above which would include: momentum, volume, moving averages, weighted moving averages, r-squared, volatility ratio, linear regression, and many other technical indicators.**

all help to make solely actionable decisions, which is why they are integral to algotrading, which is **why I want to try creating and testing algorithms that use both TA, TA-lib, and candlestick recognition.**

**The drawback of solely using TA, is that many indicators need to go into decisions.**

**The drawback of candle patterns, is that they alone, even sorted, probably wont work alone.**

# Backtesting

Using historical data to run your trading algorithms over large spans of data in order to determine if your algorithm actually responds to market changes well.

# Paper trading/ walk forward testing

Gathering live data to test your algorithm in real time. This helps mitigate future-leaking algorithms and can often signal improper backtesting implementation. I’m still on the fence of whether this is a large priority for me at the moment.

# Use Cases:

Unfortunately, I have a split concept here. I want to both create algorithms and have a defined easy way of testing and creating them, the split is; I want to verify the effectiveness of using Candlestick pattern recognition as a valid avenue of writing simple algorithms to predict market movement, and TA to control my positions.

To reiterate blandly my use cases are:

A user wants to write a trade algorithm, they inherit the class, write their algorithm, and the software will test whether the algorithm is robust, or at least the level of robustness.

The system generates this data by analyzing the trade history of the algorithms as well as charting trading position over time. Some examples being trading speed/turn around times, rate of profit, inter-algorithm performance, duplicating the same algorithm with different initialization parameters to find the maximum ROI and seeing how profit is affected and gauging consistency over time with graphs returned to the user. I see statistical value in these “meta-analytics” of not how to algorithm is doing, but how to calibrate any parameters needed, and how they affect the algorithm at different times is a possible avenue. If you have parameters, are they actually making a difference or only temporarily? Its to this regard that I think other trading tools are lacking.

Why TA-Lib candlestick recognition? Because to me, the efficacy of candlestick recognition is not a foregone conclusion, and its addition to TA might improve the efficacy of algorithms. It is by that compare and contrast basis that I wish to form my tests. My big question I wish to attempt to partially answer is: Is candlestick recognition a valid way of improving trading algorithms? And what contrasting data can I extract that shows the effect of doing so.

When I looked around for competing solutions, comprehensive candlestick analysis was not something I saw more than a few times. It looks sketchy, clearly if it worked it would be everywhere and a big deal or maybe its an open secret. Its a tool that is clearly in use, from what I can see.

# Constraints and timelines-

The system should:

- be able to identify poorly performing trade algorithms on performance metrics.

- ROI, gross/net profit, trade age, record why a trade (buy and sell) was made and what trigger was involved, and be able to display that in a graph.

- Back-calculate after the trading calculation to see if trades were timed correctly by shifting or expanding trade dates +/- 24 hours/one trade step.

- Back-calculating trade performance should provide a ham-fisted ratio to theoretical maximum return and actual return from the original trade

- Simulate a configurable flat tax

- Ultimately the goal is to try to implement the original ideas of automated trade analysis/graphing so a user may contrast and compare results. Whatever automated trade analysis gets done is extra.

- Because of the compare and contrast nature of the project, some effort should be maintained to decouple TA behavior and Candlestick analysis behavior. I intend to do this by implementing separate class types for each style of algorithm to fine tune behavior and build algorithm logic at the child class level.

- This might be difficult to do at the child class level, because candle signals might be triggers themselves. In any case, implementing different combined and separated approaches might not be possible without drawback. In any case, the idea is to attempt to convey the difference between combined and separate approaches of TA and Candle analysis

- If I cant implement back-calculation at the parent class level, we might need to make a separate class that walks over both the trade history and market history in step.

- Ultimately the amount of item purchased may be static, but if you can, more should be acted upon depending on the strength of the signals

# Timeline of implementations:

1. skeletonize class structure

2. implement buy/sell at parent class level

3. implement trade history

4. skeletonize child classes to implement separate algorithms

5. implement a basic bullshit trade algorithm to produce trade history for back data

6. graph trade data from backtests

7. begin to create trade algorithms using TA, Patterns, and the combinations.

8. compare strategies against HODL

9. begin to implement automated trade tweaking

10. graph trade meta-performance generated from trade tweaking against trade history

11. bonus: implement tax into trades