How to Build a Stock Trading Bot from Scratch

Trading strategy, resources, and advice from someone who has done it before.

Ary Sharifian Jul 12, 2020 · 8 min read★

Photo by Nicolas Hoizey on Unsplash

Having trouble thinking of a strategy? Not sure which APIs and packages to use? Don't want to reinvent the wheel? I built a stock day trading program (github repo) from scratch and wanted to share some helpful resources as well as some advice on how to get started. I know starting a new project, especially in a foreign domain, is challenging, and I hope this article can help flatten the learning curve. Taking a step back, I also wanted to discuss my criteria for a "good" project. *There are many projects you can work on, so why work on this?*

Disclaimer: *this is not meant to represent financial advice. Any investments you make using the algorithm, strategy, or ideas below is at your own risk. I am not liable for any consequences related to or caused by the information contained in this article.*

Outline:

- Why I think building a trading bot is a "good" project: before spending time building an investment bot, its important to ask whether this is a good investment of your time.
- How to get started: I provide some example psuedocode to help you get started.
- Resources: a list of github repos, websites, other medium articles, youtube videos, books, and podcasts that helped me tremendously.

Why I think building an trading bot is a "good" project:

1. Incorporate technologies of interest

Given that the complexity of buying or selling a stock/cryptocurrency/forex is nearly infinite, there is plenty of room to introduce new technology. Are you a ***data scientist***and want to analyze some Tweets from Elon Musk or identify keywords in SEC Filings? An aspiring ***cloud engineer*** looking to use cloud for something like being able to run your scripts 24/7 on an EC2 instance? A ***backend developer*** wanting to provide end users an API to get signals from your algorithm of when to buy and sell? An ***ETL or data engineer*** wanting to play with big data tools like Kafka, Spark, DynamoDB and build a pipelines to streams price data and throw it into NoSQL/SQL databases. A ***frontend developer or financial analyst*** interested in using DASH or React+Flask to represent the algorithm's performance to end users? You want to incorporate some MACHINE LEARNING? Hopefully you get the idea.

2. Collaborative

Easily collaborate with others by breaking your program into areas of interest. I guess any project can be collaborative, but given the complexity of this challenge, people can choose their interest and simply tackle those challenges. It's also helpful that the overall objective of this project is clear, **make money**. Sometimes its hard to collaborate because people get lost in the details and forget the overall objective. Whether you are building a data pipeline, creating dashboards, or building some machine learning model, the objective is clear.

3. Clear Measure of Success: $$$

Sometimes its hard to measure success but with this project, knowing how much money the program has made or loss is the ultimate indicator. Frankly, from a learning standpoint, its a win-win. If you make money, think how you can make more money or lose less. If you are losing money, think how you can make more money or lose less. That's right, there is no difference. Always room for improvement and all effort amounts to the same measure. Sometimes with other projects, its hard to know whether your changes are beneficial. Success may rely on user feedback or just a matter of opinion. This project, in contrast, is great because success and failure is clear. Nevertheless, its important to remember that markets are infinitely complex. *Though it is easy to measure progress, this doesn't mean that it is easy to make progress*.

4. Easily understood by recruiters/future employer/grandma

It's good to have a project that everyone can understand. Most everyone invests in the stock market. Its easy understanding the goal of buying low and selling high. When employers ask about projects, most ramble on about some technology stack they've implemented and lose the recruiter or hiring manager in the details. This isn't to suggest that the technology isn't important, but that the ultimate goal of this project makes it easier to explain the technology. For example, "I wanted to limit the scope of stock tickers so I used K-Means

Clustering to cluster all my successful trades and find similar stocks" versus "I implemented a K-Means Clustering". Even if you don't know what K-Means Clustering is, you can understand its purpose.

5. Potentially Lucrative (but unlikely)

You can make money building a successful algorithm that trades for you. When I first started on this project a few months ago, I was convinced that building something lucrative is mostly luck and chance. I still mostly feel that way, but I believe *it is possible*to build something profitable. I would never risk money I can't afford to lose, and I strongly recommend others adopt the same mindset. Making money should not be the goal of this project, but its a nice side-benefit and aspirational goal.

6. Infinitely Complex

This ties in with my earlier point about easily incorporating any technology. Infinite complexity means you will never be done. (I am viewing this as glass half full). There will always be new strategies, technology, indicators, and metrics to incorporate and test. It's a never ending game. Interpret that how you will, but I find that interesting and exciting.

How to get started?

Define a strategy

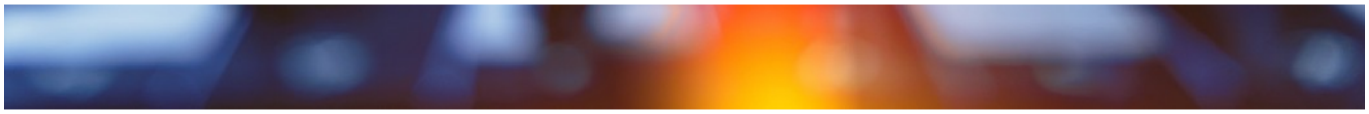
Photo by JESHOOTS.COM on Unsplash

Develop a strategy that works. Notice I didn't say to create a profitable strategy. Fact is that you will iterate. Defining a strategy will help provide some framework that can be improved upon. This is an example strategy that I created for trading stocks:

I explained this strategy in another article I wrote about the initial performance of my stock trading algorithm. Now iterate upon this strategy and provide more details. Once you feel like you have something you can easily implement, start thinking about how to implement. This is when I felt I could start implementing:

Note: notice that include API calls that I expect to use during implementation. This might be more specific than is currently needed during the pseudocode phase.

How will you implement your strategy?

What APIs, packages, and other resources help or are necessary to implement this pseudocode? Is this even possible? This is when I did some research and found that there are stock APIs for Robinhood (NOT RECOMMENDED), TD Ameritrade, and Alpaca that can execute buy and sell orders. For news stories, I am thinking about doing some web scraping using Python modules Beautiful Soup and Selenium and Scrapy. I can use Yahoo Finance API for getting moving averages and tracking volume. For building some machine learning models I might use scikit-learn, pandas, and numpy (I don't recommend focusing on machine learning too much in the beginning). I wrote my initial program in Jupyter Notebook and used Github as my repo.

Now at this point, you may be tempted to start delving into new technologies and platforms such as incorporating some cloud or using airflow or kubeflow, but I recommend focusing on implementing as quickly as possible. If are interested in incorporating other technology and are a bit more experienced, you can do Proof of Concepts (POCs) for the technologies you plan to use *before* incorporating the technology. If you are working in a group, some might focus on just doing POCs and seeing what works best.

Iterate: fail-fast

Simply going through the process of implementing the psuedocode will teach you so much. You begin to understand where the bottlenecks are and improvements in design. And perhaps have learned about new APIs, packages, or frameworks that can be useful. I also recommend doing some paper trading (simulated trading) to test your program's performance. Keep in mind, nothing is like trading live. When you live trade there are a variety of factors that will impact your program's performance. Read one of my earlier articles which describes some of the challenges I faced when building my algorithm.

Resources

Photo by Fikri Rasyid on Unsplash

Below are some of the resources that helped me get started and might help you too. Again, some might provide specific tangible advice while others provide you some domain knowledge and expertise.

Note: I will continuously update the list below with useful resources.

Github Repos

1. A curated list of insanely awesome libraries, packages and resources for Quants (Quantitative Finance)

2. News API

3. Trump Tweet to Cash

4. Scraping News Articles

Other Medium Articles

1. Stock Market API

2. Stock Market Strategies (High Level)
3. Indicators and more High Level
4. How to get indicators values (from Finviz using Scraping)
5. Example of Using Indicators
6. Scalping Strategy implementation that inspired my algorithm
7. Financial Web Scraping Example
8. Building a bitcoin streaming backend
9. Streaming Stock Data
10. Elon Musk Tweets to Cash

Websites:

1. finviz: click on a ticker to see all the indicators and filters professional traders use. This might inspire which indicators to use for your own filtering algorithm.
2. MarketWatch: a popular news source for day traders. (might be interesting to scrape these stories and do some sentiment analysis).
3. SEC Filings: if you enter a ticker, you will see all the official filings for the company. (We are thinking about scraping some of these filings to inform our stock trading algorithm).
4. Subreddit AlgoTrading: subreddit for algorithmic trading. Some great resources here too. Here is a discussion about the best API trading platforms. Introduction to Python for trading algorithms.

Youtube

1. Bear Bull Traders: Training video for beginner traders
2. Forrest Knight: Resources for building day trading algorithm
3. Interesting TED Talk about AI in FOREX

Books

1. Flash Boys: the reason why everyone should be skeptical about making money in the stock market.

Podcasts:

These are some podcast that describe high level strategies. (You can also just google “day trading podcast” — there are many).

1. Desire To Trade: interview with a day trading system developer
2. Top Dog Trading