Abhi Uppal
AIML 168
Homework 1

Feel free to work with other students, but make sure you write up the homework and code on your own (no copying homework *or* code; no pair programming). Feel free to ask students or instructors for help debugging code or whatever else, though.

Note: You need to create a Github account for submission of the coding part of the homework. Please create a repository on Github to hold all your code and include your Github account username as part of the answer to the coding problems.

1 Follow the tutorial in the following link and run all the code in the tutorial on your own machine and submit the code and output as homework 1.
`https://www.datacamp.com/community/tutorials/finance-python-trading`

■

1 Reading Summary: Algorithmic Trading

The first question that the article answers is: *why algorithmic trading?* The answer is quite obvious — computers can use their speed and computational power to gain an edge over human traders. The obvious drawback is that, when creating an algorithm, we need to create a strict set of rules to follow. When we do this, often times we make oversights and bugs that could lead to problems, especially when money is involved. As such, we want to make sure we're as thorough as possible in creating our algorithms, and this article discusses some steps towards that.

The first thing the article discusses is important definitions. Stocks are shares of companies that are traded in markets (though algorithmic trading is not just limited to stocks), and trading these to make a profit happen in two ways: *long* and *short*. Going long requires a trader to buy stocks and sell at a higher price in the future. Going short requires a trader to sell their stock, hoping to earn a profit by buying it back for a lower price in the future. Either way, in order to do this, we will need to build predictive models and test them against historical data. In order to do this, we need to understand time series analysis.

The core packages we'll be utilizing are Pandas, NumPy, SciPy, Matplotlib, and others. We'll need the pandas-datareader package to read in financial data (though other packages exist). These packages will allow us to manipulate and analyze our data easily. In Pandas, we can store data in a *DataFrame*, which holds a collection of 1-dimensional *series*. These can be imported from or exported to CSV files.

After discussing various methods for manipulating, plotting, and modeling time series (using OLS), the article then moves on to building a specific trading strategy. The commonly used *momentum strategy* assumes that stocks will follow their current direction (i.e. upward trends will stay upward, downward trends will stay downward). There are various ways of determining how to figure out when the trend switches (and therefore when to enter and exit the market), three of which are covered by the article. These are the moving average crossover, the dual moving average crossover, and turtle trading. The other strategy is known as the *reversion strategy*. This assumes that stock prices will revert back towards their mean shortly after deviating from them. An example of this is pairs trading mean-reversion, which takes advantage of highly correlated stocks in order to create long positions for one stock while producing a long position for the other stock. Beyond these, one could try to simply predict the *direction* of a stock's price, or exploit the structure of the market on incredibly short time scales. The latter strategy is known as *high-frequency trading (HFT)*.

After devising a trading strategy, it's advisable to backtest it against historical data before bringing it live on the market. This will allow us to get a rough idea of how well our algorithm will perform in the market. However, it's important to realize that it is nowhere near perfect: a myriad of biases can creep in, including not being able to model major macroeconomic events, overfitting, lookahead bias (using data that should not have been available at the time), as well as disobeying the rules of the algorithm to produce better results. Some of the major components of a backtest are data handlers, which allows access to the correct set of data, a strategy that generates long and short signals, a portfolio that tracks holdings and profits, and an execution handler which executes trades based on the signals. More components can be added, though. We can backtest using Pandas or use Quantopian, which is an open-source backtesting platform. This is based on the zipline library, which we can import to run locally.

In order to improve our trading strategy, we may want to move to complicated modelling strategies for prediction. Additionally, the example in the article looks only at one stock — using multiple to mitigate risk will almost certainly yield better performance. It's important to understand how we can measure performance as well. The *Sharpe ratio* is the ratio of the mean of returns to the standard deviation of returns. Higher values of this are, as expected, better than lower ones. Additionally, the maximum drawdown measures the largest drop from peak to trough in a portfolio, which can be used to quantify the risk of a strategy. Another metric is the *Compound Annual Growth Rate (CAGR)*, which measures what you actually have at the end of an investment period. There are many other metrics that can be used, as well.