DEEP REINFORCEMENT LEARNING FOR TRADING

Master of Science

A Capstone Project Paper Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Data Science

Gautam Bhowmick
bhowmg91@uwosh.edu

For my Father,

Who believes in Karma

Acknowledgement

This document provides details about make predictions for potential investors who might be willing to invest in shares.

The target group for this article is potential investors, both individual and institutional investors who are planning to invest in shares. The report provides details to conduct successful trading, accurate predictions of price movement to take BUY, SELL, HOLD decisions.

Potential investors can read this article and make informed decisions about whether to invest in shares.

This article is mainly for potential investors who have basic knowledge of financial markets. Investors who do not have a basic understanding of financial markets should contact their financial advisors.

It is a financial document and the writing design intended for financial advisors or individual investors who have some basic knowledge of financial markets. It cannot be written to the general audience because some financial terms must be used to explain the shares.

This document is not for investment advice, and the AI tool I fabricated is just for educational purposes. All my positions in any stocks disclosed in this article, and I have no affiliation with any companies mentioned other than sometimes owning their stock.

Abstract

While there is significant activity in creating sophisticated AI and ML based solutions for many problems (from medical diagnostics to natural language interaction with a machine), there is relatively less movement in preparing an algorithmic pricing bot. Most important barrier to entry is that it takes significant knowledge to understand the inner workings of the financial markets (exchanges, liquidity, order book, bid-ask-spreads etc.). Secondly, to conduct successful trading, accurate predictions of price movement is necessary but not enough. Simple prediction strategies would be derailed by practical considerations like market liquidity, network latency and of course transaction fees,

An agent would have to be trained to take BUY, SELL, HOLD decisions. Along with that, the agent would have to determine position sizing and ramp-up and ramp-down strategies which typical traders do with years of experience. Some researchers think using a Reinforcement Learning technique is the best method so that an agent can make decisions and get rewards and punishments. The objective would be to optimize the discounted reward over a finite time horizon. This can be accomplished by formulating the problem as a multi-player reinforcement learning problem, with agents acting on an environment, with a system to track rewards for each action taken. Based on all this an agent learns a policy for trading to optimize the best outcome.

Cryptocurrencies present a whole new level of challenge. Not only all the knowledge of traditional financial and currency exchange markets is important (e.g. technical indicators like MACD, RSI and Bollinger Bands based on minute by minute historical data sets) but a whole new set of fundamental indicators (transactions, addresses, NVT based on blockchain data) needs to be included. Additionally, sentiment analysis from social media and productivity metrics from GitHub can be incorporated.

The beauty of cryptocurrencies is all these data is easily available from public ledgers. That too for free! But the biggest reason is crypto markets never sleep. So crypto currencies are the best candidates for automating AI based learning bots. That way a human does not have to wake up in the middle of the night to watch or execute her trade!

Contents

1. Introduction

Developing trading algorithms makes predictions for the shares which investors can use for trading shares. Based on the forecast, investors can sell the (predicted) worst-performing stock that they own (no shorting) in the portfolio and buy the (predicted) best-performing share. The scope of this project is only for Crypto Trading, which can extend in the future for Equity trading.

1.1 Background of the Project

Use machine learning models running on a cloud environment to develop an autonomous trading bot. I'm going to use Deep Reinforcement Learning techniques for agents to trade in the financial (and cryptocurrency) markets can be an exciting topic for commercial investors.
The finance domain is vast and complex, so you can easily spend several years learning something new every day. In this project, I'll scratch the surface a bit with Reinforcement Learning (RL) tools, and the problem will be formulated as simple as possible, using price as an observation. I will investigate whether it will be possible for an agent to learn when the best time is to buy one single share and then close the position to maximize the profit. The purpose of this project is to show how flexible the RL model can be and what the first steps are that you usually need to take to apply RL to a real-life use case.

To formulate RL, three things are needed: **observation** of the environment, possible **actions,** and a **reward** system.

1.2 Purpose of the Project

The stock market is random; I mostly believe that it's a random walk (or would be in an ideal world, but markets are sadly not efficient). Despite thinking I know everything about hardware and software companies, I have found that actively trading shares is quite tricky. I have been consistently losing money week-on-week. First, you must find the time to be on top of the news about your portfolio companies, their industry segments, and market fundamentals. Second, having friends at those companies would probably help. And third, it all just seems a bit random.

Instead of doing all that, which sounds painful, I decided I'd do what I know how to build an AI to trade.

1.3 Basic concepts of Trading

I expect most readers to have no background in trading, just like I didn't, so I will start by covering some of the basics. I will use cryptocurrencies as a running example in this post, but the same concepts apply to most of the financial markets.

1.3.1 Stocks & Trading

When a company wants to grow and undertake new projects or expand, it can issue stocks to raise capital. A stock represents a share in the ownership of a company and is issued in return for money. Stocks are bought and sold buyers and sellers trade existing, previously issued shares. The price at which stocks are sold can move independent of the company's success: the prices instead reflect supply and demand. This means that whenever a stock is considered as 'desirable', due to success, popularity, the stock price will go up.

Stock trading is then the process of the cash that is paid for the stocks is converted into a share in the ownership of a company, which can be converted back to cash by selling, and this all hopefully with a profit. Now, to achieve a profitable return, you either go long or short in markets: you either by shares thinking that the stock price will go up to sell at a higher price in the future, or you sell your stock, expecting that you can buy it back at a lower price and realize a profit. When you follow a fixed plan to go long or short in markets, you have a trading strategy.

1.3.2 Broker

A person who buys or sells an investment for you in exchange for a fee (a commission).

1.3.3 Bid

The bid is the amount of money a trader is willing to pay per share for a given stock. It's balanced against the ask price, which is what a seller wants per share of that same stock, and the spread is the difference between those two prices.

1.3.4 Close

The NYSE and Nasdaq close at 4 p.m., with after-hours trading continuing until 8 p.m. The close simply refers to the time at which a stock exchange closes to trading.

1.3.5 Trading

The practice of buying and selling within the same trading day, before the close of the markets on that day, is called day trading. This is my primary trading strategy, although I have a long-term portfolio, as well. Traders who participate in day trading are often called "active traders" or "day traders."

1.3.6 Dividend

A portion of a company's earnings that is paid to shareholders, or people that own that company's stock, on a quarterly or annual basis. Not all companies pay dividends. For instance, if you trade penny stocks, you're likely not after dividends.

1.3.7 Exchange

A place in which different investments are traded. The most well-known exchanges in the United States are the New York Stock Exchange (NYSE) and the Nasdaq.

1.3.8 High

A high refers to a market milestone in which a stock or index reaches a greater price point than previously. Record highs can signal that a stock or index has never reached the current price point, but there are also time-constrained highs, such as 30-day highs.

1.3.9 Index

A benchmark that is used as a reference marker for traders and portfolio managers. A 10 percent return may sound good, but if the market index returned 12 percent, then you didn't do very well since you could have just invested in an index fund and saved time by not trading frequently. Examples are the Dow Jones Industrial Average and Standard & Poor’s 500.

1.3.10 Low

Low is the opposite of high. It represents a lower price point for a stock or index.

1.3.11 Margin

A margin account lets a person borrow money (take out a loan, essentially) from a broker to purchase an investment. The difference between the amount of the loan and the price of the securities is called the margin.
Trading on margin can be dangerous because, if you’re wrong about the direction in which the stock will go, you can lose significant cash. You must often maintain a minimum balance in a margin account.

1.3.12 Open

In the United States, the stock market opens at 9:30 a.m. Eastern time every day. It’s based on the trading hours of the Nasdaq and NYSE. Pre-market trading hours begin at 4:30 a.m. Eastern, but most traders don’t begin paying attention until about 8 a.m. Essentially, open refers to the time at which people can begin trading on a exchange.

1.3.13 Order

An investor’s bid to buy or sell a certain amount of stock or option contracts constitutes an order. You have to put an order in to buy or sell 100 shares of stock, for instance.

1.3.14 Quote

Information on a stock’s latest trading price tells you its quote. This is sometimes delayed by 20 minutes unless you’re using an actual broker trading platform.

1.3.15 Rally

A rapid increase in the general price level of the market or of the price of a stock is known as a rally. Depending on the overall environment, it might be called a bull rally or a bear rally. In a bear market, upward trends of as little as 10 percent can qualify as a rally.

1.3.16 Share Market

Any market in which shares of a company are bought and sold. The stock market is an example — and probably the most significant example — of a share market.

1.3.17 Spread

This is the difference between the bid and the ask prices of a stock, or the amount for which someone is willing to buy it and the amount for which someone is willing to sell it. For instance, if a trader is willing to trade XYZ stock for $10 and a buyer is willing to pay $9 for it, the spread is $1.

1.3.18 Stock Symbol

A stock symbol is a one- to four-character alphabetic root symbol that represents a publicly traded company on a stock exchange. Apple’s stock symbol is AAPL, while Walmart’s is WMT.

1.3.19 Volatility

The price movements of a stock or the stock market. Highly volatile stocks are those with extreme daily up and down movements and wide intraday trading ranges. This is often common with stocks that are thinly traded or have low trading volumes.

I'm a big fan of high-volatility stocks because I can make a big profit off spikes or dips, depending on how I'm trading, in a short period of time. High volatility often makes trading more exciting, but it's also risky if you're inexperienced.

1.3.20 Volume

The number of shares of stock traded during a time period, normally measured in average daily trading volume. Volume can also mean the number of shares you purchase of a given stock. For instance, buying 2,000 shares of a company is a higher-volume purchase than buying 20 shares.

1.3.21 Sharpe Ratio

The Sharpe Ratio measures the excess return per unit of risk you are taking. It's basically your return on capital over the standard deviation, adjusted for risk. Thus, the higher the better. It takes into account both the volatility of your strategy, as well as an alternative risk-free investment.

1.3.22 Order Book

An order book is an electronic list of buy and sell orders. An order book lists the number of shares being bid or offered at each price point, or market depth. It also identifies the market participants behind the buy and sell orders.

2. Data Collection

Cryptocurrencies data is public, free, and easy to obtain. Most exchanges have streaming APIs that allow you to receive market updates in real-time. Following exchanges are sources of Data collection of this project:

- Karken
- Coinbase
- Bitstamp
- Itbit
- Poloniex

2.1 Cryptocurrency

Cryptocurrency is a digital currency built with cryptographic protocols that make transactions secure and difficult to fake. Crypto assets are block-chain based asset to become a new asset class for investors. Why Cryptocurrency is the next magical thing the earthlings are going to embrace:

- Prevention of scams and hacks
- No more human involvements
- More employment opportunities
- Blockchain in financial institutions
- Other industries adopting Blockchain
- Blockchain in building technology infrastructure
- Easy to use and secure
- Transactions are faster and private
- Productive and efficient

2.2 Bitcoin

Bitcoin is a cryptocurrency. It is a decentralized digital currency without a central bank or single administrator that can be sent from user to user on the peer-to-peer bitcoin network without the need for intermediaries.

2.3 Altcoin

An altcoin, or alternative coin, is self-explanatory. An altcoin is every cryptocurrency alternative to Bitcoin – the first one. Altcoins may differ from Bitcoin in every possible way, such as mining mechanisms, coin-distribution methods or the ability to create decentralized applications.

2.4 Currencies Symbol

Name	Symbol	Type
Bitcoin	BTC	Bitcoin
Litecoin	LTC	Altcoin
Ethereum	ETH	Altcoin
Ripple	XRP	Altcoin
Dash	DASH	Altcoin
Monero	XMR	Altcoin

2.4 Data Structure

JSON Structure - Download Bitcoin data from *Quandl*	**JSON Structure- Download Altcoin Data from *Poloniex***

`Date",`
`"Open",`
`"High",`
`"Low",`
`"Close",`
`"Volume (BTC)",`
`"Volume (Currency)",`
`"Weighted Price"`
	`"date",`
	`"high",`
	`"low",`
	`"open",`
	`"volume",`
	`"quoteVolume",`
	`"weightedAverage"`

2.5 Setup and Data Fetch

- Import the required dependencies.
- Import Plotly.
- Setup Quandl API and get an API key.
- Define a function to download and cache datasets from Quandl.
- Use pickle to serialize and save downloaded data as a file, avoid re-downloading the same data.
- The function will return the data as a Pandas dataframe. These are spreadsheets on steroids!

2.6 Data Collection Steps

Bitcoin	**Altcoin**
• Quandl API Key - To work with any version of the Quandl APIs you must first ensure that you have a Quandl API client key. • Define Quandl API Helper Functions • to download and cache JSON data from this API. • **get_qualdal_data**, which will download and cache JSON data with Quandl APY key. • Call **get_qualdal_data** function to save the resulting data. • Download Bitcoin Data from Quandl	• Retrieve Altcoin Pricing Data • Define Poloniex API Helper Functions • Define two helper functions to download and cache JSON data from this API. • **get_json_data**, which will download and cache JSON data from a provided URL. • Format Poloniex API HTTP requests and call **get_json_data** function to save the resulting data. • Download Trading Data from Poloniex • We'll download exchange data for five of the top cryptocurrencies

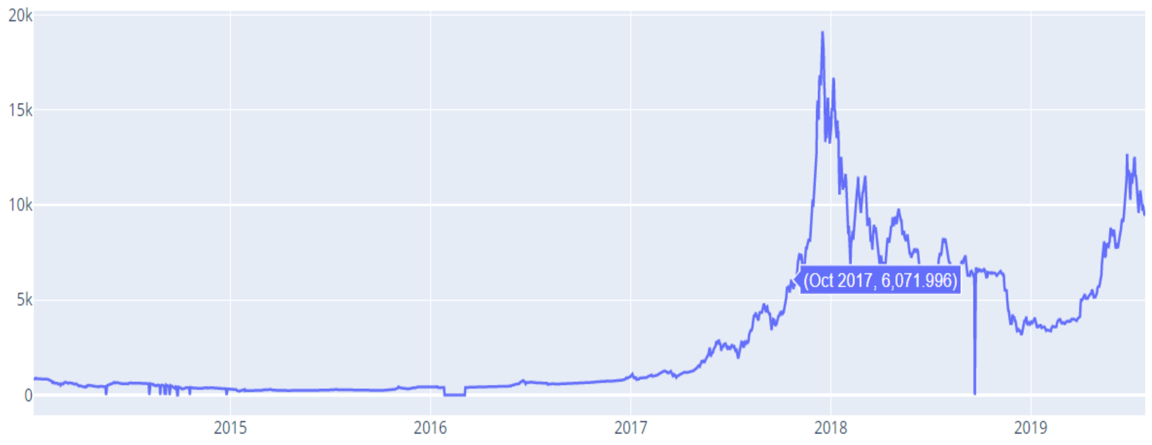
3. Data Analysis

(*Charts*, *Correlation*, *Distribution*) - You perform exploratory data analysis to find trading opportunities. You may look at various charts, calculate data statistics, and so on. The output of this step is an “idea” for a trading strategy that should be validated.

3.1 Kraken Bitcoin exchange – Download and Visualization

- Historical Bitcoin exchange rate from the Kraken Bitcoin exchange.
- Inspect the first 5 rows of the dataframe using the tail () method.
- Visually verify that the data looks correct.
- Plotly is used for visualizations. Produces fully interactive charts using D3.js.
- These charts have attractive visual defaults, are easy to explore, and are very simple to embed in web pages.
- Compare the generated chart with publicly Bitcoin prices (e.g. Coinbase)

Date	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
2019-07-25	9770.0	10187.4	9734.6	9878.0	5604.361026	5.611320e+07	10012.416316
2019-07-26	9878.0	9911.3	9625.2	9852.0	4322.503164	4.225617e+07	9775.856187
2019-07-27	9850.9	10204.6	9269.1	9471.8	6349.438830	6.137714e+07	9666.544853
2019-07-28	9472.8	9630.0	9120.0	9531.7	3862.187969	3.644929e+07	9437.472739
2019-07-29	9536.2	9709.7	9375.0	9510.4	3934.166699	3.752293e+07	9537.708035

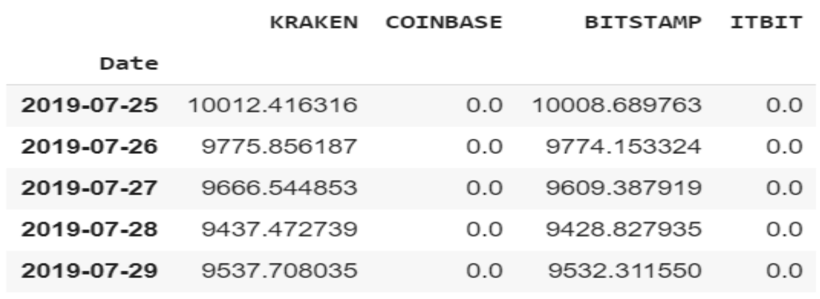


3.2 Other Bitcoin exchange (Coinbase, Bitstamp and Itbit)– Download & Visualization

- Pull Pricing Data from various BTC Exchanges. There is a reason!
- Kraken has a glitch in dataset - few notable down-spikes, late 2014 and early 2016.
- Distributed nature of Bitcoin exchanges means no single exchange contains a true "master price" of Bitcoin.
- Pull data from three major exchanges and calculate an aggregate Bitcoin price index.
- Download the data from each exchange into a dictionary of dataframes.
- Combine Average Price of each individual Bitcoin exchange to verify prices.

- Preview and verify last five rows the result using the tail () method
- Visualize the Pricing Datasets to compare.

3.2.1 Verify Bitcoin -Weighted Price- across Exchanges

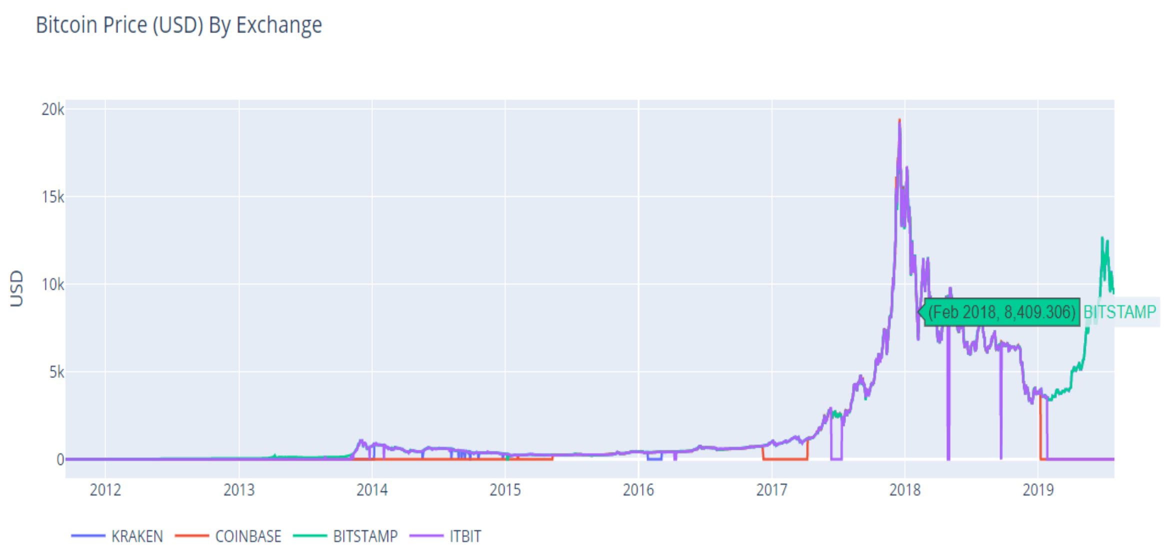


	KRAKEN	COINBASE	BITSTAMP	ITBIT
Date				
2019-07-25	10012.416316	0.0	10008.689763	0.0
2019-07-26	9775.856187	0.0	9774.153324	0.0
2019-07-27	9666.544853	0.0	9609.387919	0.0
2019-07-28	9437.472739	0.0	9428.827935	0.0
2019-07-29	9537.708035	0.0	9532.311550	0.0

The prices look to be as expected: they are in similar ranges, but with slight variations based on the supply and demand of each individual Bitcoin exchange.

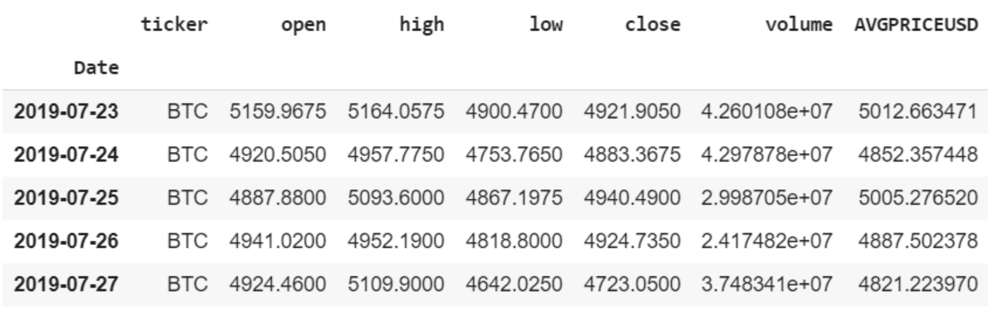
3.2.2 Visualize - Bitcoin -Weighted Price- across Exchanges

- View how these pricing datasets compare.
- Define a helper function to provide a single-line command to generate a graph from the dataframe.



3.2.3 Compute average of Bitcoin pricing data

- Calculate average of open, high, low, close and volume.
- A new column, containing the daily average Bitcoin prices.
- Merge All of The Pricing Data into A Single Dataframe.
- Create a new Colum 'ticker' to identify currency Symbol



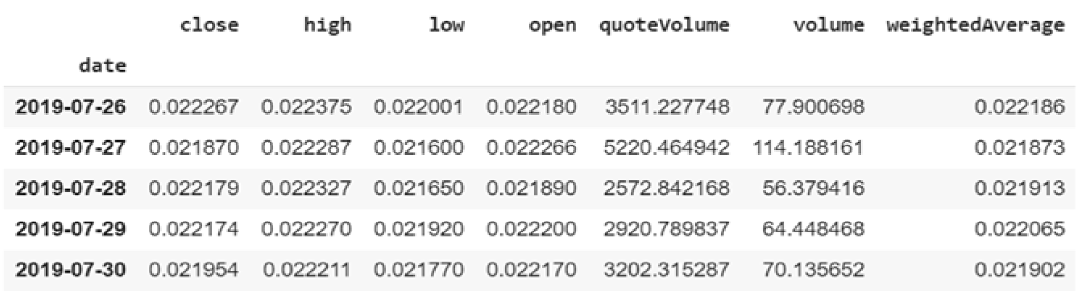
Date	ticker	open	high	low	close	volume	AVGPRICEUSD
2019-07-23	BTC	5159.9675	5164.0575	4900.4700	4921.9050	4.260108e+07	5012.663471
2019-07-24	BTC	4920.5050	4957.7750	4753.7650	4883.3675	4.297878e+07	4852.357448
2019-07-25	BTC	4887.8800	5093.6000	4867.1975	4940.4900	2.998705e+07	5005.276520
2019-07-26	BTC	4941.0200	4952.1900	4818.8000	4924.7350	2.417482e+07	4887.502378
2019-07-27	BTC	4924.4600	5109.9000	4642.0250	4723.0500	3.748341e+07	4821.223970

3.3 Altcoin – Download and Visualization

- Download Trading Data from Poloniex

  * Most altcoins cannot be bought directly with USD; to acquire these coins individuals often buy Bitcoins and then trade the Bitcoins for altcoins on cryptocurrency exchanges. *

- Download the exchange rate to BTC for each coin, and then use existing BTC pricing to convert this value to USD.
- We'll download exchange data for five of the top cryptocurrencies
  - Ethereum
  - Litecoin
  - Ripple
  - Dash
  - Monero
- Now we have a dictionary of 5 dataframes, each containing the historical daily average exchange prices between the altcoin and Bitcoin. We can preview the last few rows of the Ethereum price table to make sure it looks ok.
- Dictionary with 5 dataframes, each containing the historical daily average exchange prices between the altcoin and Bitcoin.
- Preview the last few rows of the Ethereum price table to visually verify



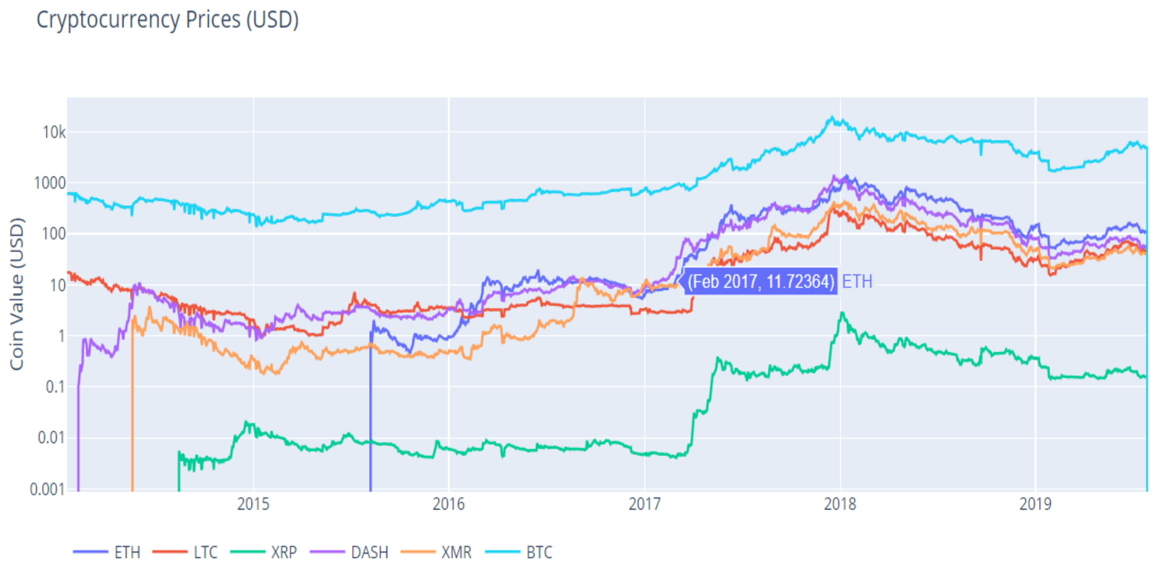
date	close	high	low	open	quoteVolume	volume	weightedAverage
2019-07-26	0.022267	0.022375	0.022001	0.022180	3511.227748	77.900698	0.022186
2019-07-27	0.021870	0.022287	0.021600	0.022266	5220.464942	114.188161	0.021873
2019-07-28	0.022179	0.022327	0.021650	0.021890	2572.842168	56.379416	0.021913
2019-07-29	0.022174	0.022270	0.021920	0.022200	2920.789837	64.448468	0.022065
2019-07-30	0.021954	0.022211	0.021770	0.022170	3202.315287	70.135652	0.021902

3.3.1 Convert Altcoins Prices to USD

- Since we now have the exchange rate for each cryptocurrency to Bitcoin, and we have the Bitcoin/USD historical pricing index, we can directly calculate the USD price series for each altcoin.
- Here, we've created a new column in each altcoin dataframe with the USD prices for that coin.
- Next, we can re-use our merge_dfs_on_column function from earlier to create a combined dataframe of the USD price for each cryptocurrency.

- Now let's also add the Bitcoin prices as a final column to the combined dataframe.

date	ETH	LTC	XRP	DASH	XMR	BTC
2019-07-26	108.434959	46.468027	0.157915	56.657638	40.003327	4887.502378
2019-07-27	105.406439	45.033205	0.154593	55.851292	39.810632	4818.983193
2019-07-28	103.355632	44.013900	0.153760	53.154859	40.021933	4716.575168
2019-07-29	105.196998	45.064792	0.154419	53.243590	39.557609	4767.504896
2019-07-30	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000



3.4 Compute Correlation Values of The Cryptocurrencies

- Despite their wildly different values and volatility, seem to be slightly correlated.
- Especially since the spike in April 2017, even many of the smaller fluctuations appear to be occurring in sync across the entire market.
- Computes a Pearson correlation coefficient for each column in the dataframe against each other column.
- Computing correlations directly on a non-stationary time series (such as raw pricing data) can give biased correlation values. We will work around this by using the pct_change () method, which will convert each cell in the dataframe from an absolute price value to a daily return percentage.

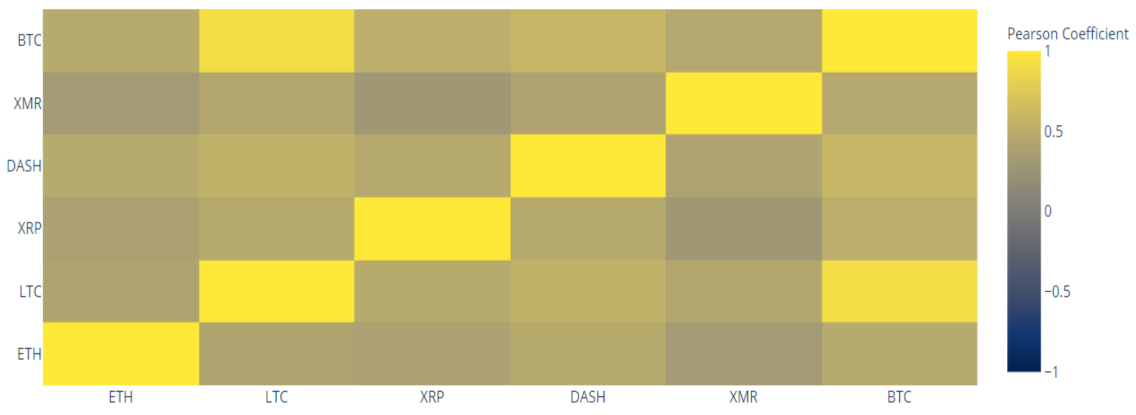
3.4.1 Pearson correlation coefficients for altcoins in 2016

- Calculate correlations for 2016.
- These correlation coefficients are all over the place. Coefficients close to 1 or -1 mean that the series are strongly correlated or inversely correlated respectively, and coefficients close to zero mean that the values tend to fluctuate independently of each other.

- Here, the dark yellow values represent strong correlations (note that each currency is, obviously, strongly correlated with itself), and the dark blue values represent strong inverse correlations. All of the light colors in-between represent varying degrees of weak/non-existent correlations.
- What does this chart tell us? Essentially, it shows that there was very little statistically significant linkage between how the prices of different cryptocurrencies fluctuated during 2016.

	ETH	LTC	XRP	DASH	XMR	BTC
ETH	1.000000	0.405814	0.382104	0.466996	0.323052	0.468396
LTC	0.405814	1.000000	0.467954	0.543109	0.436735	0.910912
XRP	0.382104	0.467954	1.000000	0.455206	0.288466	0.503339
DASH	0.466996	0.543109	0.455206	1.000000	0.400458	0.584253
XMR	0.323052	0.436735	0.288466	0.400458	1.000000	0.450863
BTC	0.468396	0.910912	0.503339	0.584253	0.450863	1.000000

Cryptocurrency Correlations in 2016



3.4.2 Pearson correlation coefficients for altcoins in 2017

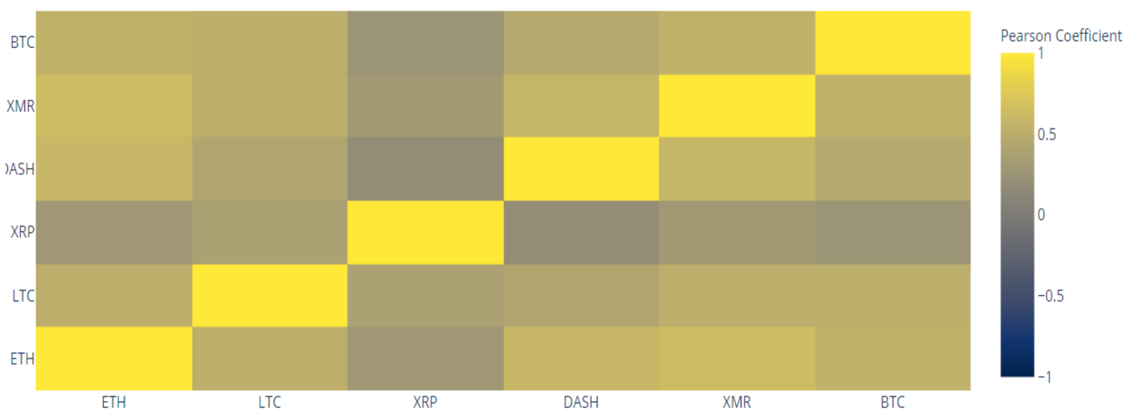
- To test hypothesis that cryptocurrencies are more correlated in recent months, let's repeat the same test using data from 2017.

	ETH	LTC	XRP	DASH	XMR	BTC
ETH	1.000000	0.510320	0.294375	0.587935	0.632141	0.544067
LTC	0.510320	1.000000	0.379068	0.422068	0.508928	0.507069
XRP	0.294375	0.379068	1.000000	0.182599	0.306323	0.250615
DASH	0.587935	0.422068	0.182599	1.000000	0.580199	0.455414
XMR	0.632141	0.508928	0.306323	0.580199	1.000000	0.543759
BTC	0.544067	0.507069	0.250615	0.455414	0.543759	1.000000

✓ Explanations

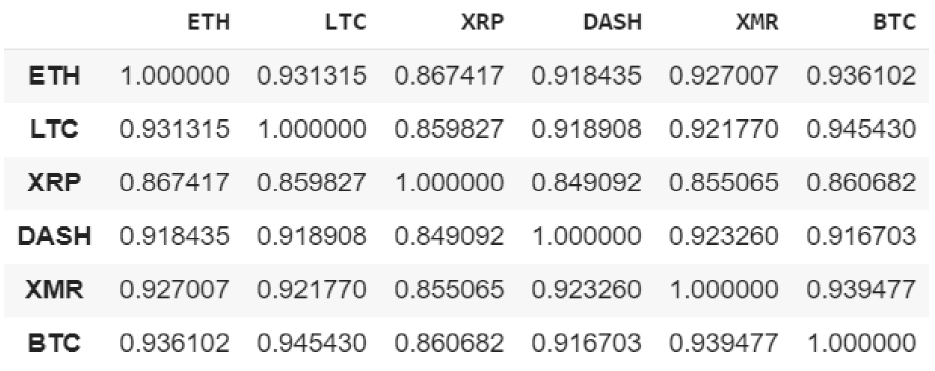
- The most immediate explanation that comes to mind is that hedge funds have recently begun publicly trading in crypto currency markets12.
- These funds have vastly more capital to play with than the average trader, so if a fund is hedging their bets across multiple cryptocurrencies, and using similar trading strategies for each based on independent variables (say, the stock market), it could make sense that this trend would emerge.
- In-Depth - XRP, for instance, one noticeable trait of the above chart is that XRP (the token for Ripple), is the least correlated cryptocurrency.
- What is interesting here is that Ripple are on fintech platforms aimed at reducing the friction of international money transfers between banks.
- It is conceivable that some big-money players and hedge funds might be using similar trading strategies for their investments in Ripple, due to the blockchain services that use each token.

Cryptocurrency Correlations in 2017



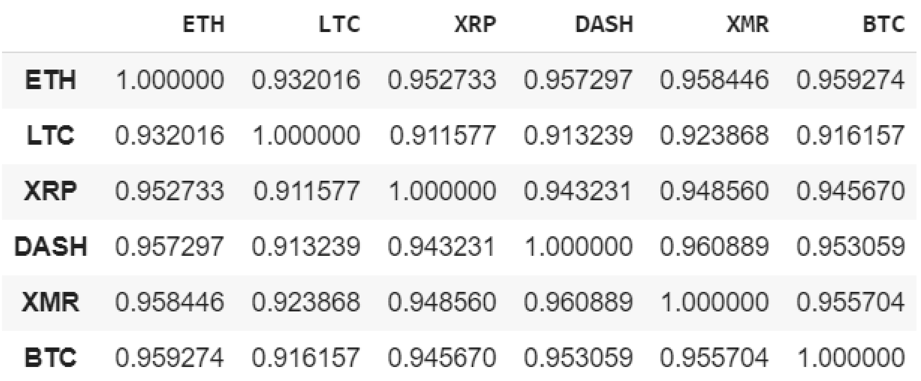
3.4.3 Pearson correlation coefficients for altcoins in 2018 and 2019

- What does this chart tell us? Essentially, it shows that there was very highly statistically significant linkage between how the prices of different cryptocurrencies fluctuated during 2018 and 2019.



	ETH	LTC	XRP	DASH	XMR	BTC
ETH	1.000000	0.931315	0.867417	0.918435	0.927007	0.936102
LTC	0.931315	1.000000	0.859827	0.918908	0.921770	0.945430
XRP	0.867417	0.859827	1.000000	0.849092	0.855065	0.860682
DASH	0.918435	0.918908	0.849092	1.000000	0.923260	0.916703
XMR	0.927007	0.921770	0.855065	0.923260	1.000000	0.939477
BTC	0.936102	0.945430	0.860682	0.916703	0.939477	1.000000

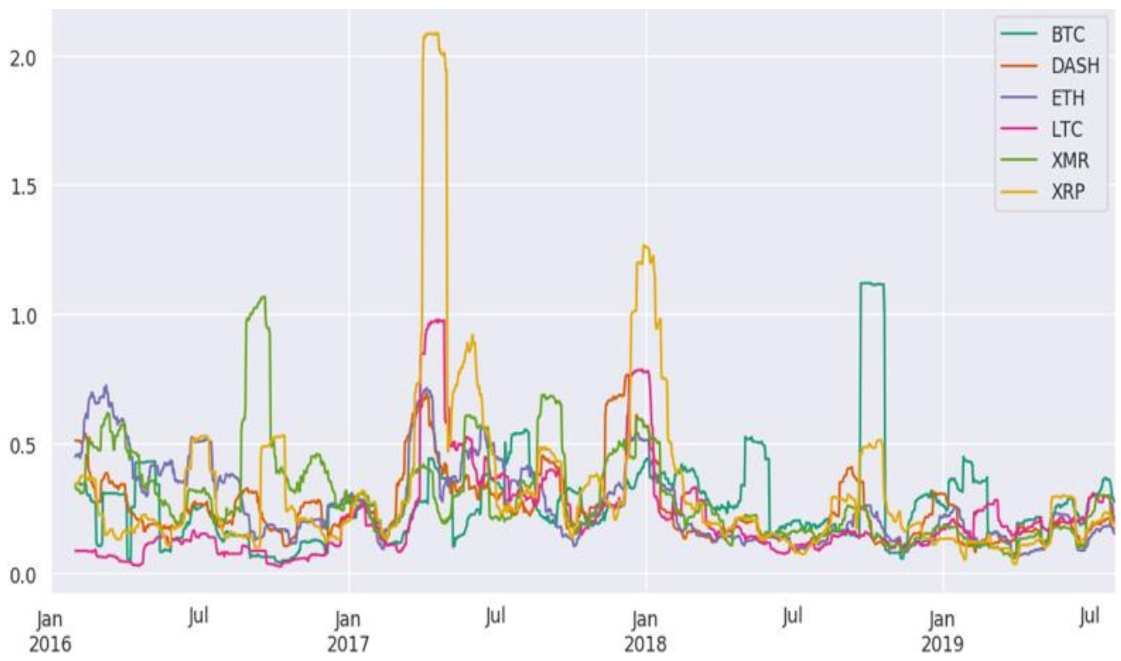
2
0
1
8

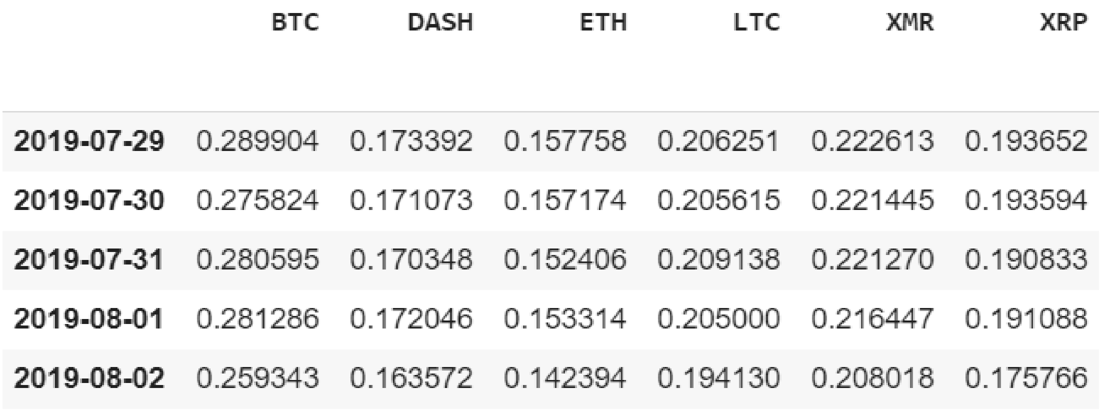


	ETH	LTC	XRP	DASH	XMR	BTC
ETH	1.000000	0.932016	0.952733	0.957297	0.958446	0.959274
LTC	0.932016	1.000000	0.911577	0.913239	0.923868	0.916157
XRP	0.952733	0.911577	1.000000	0.943231	0.948560	0.945670
DASH	0.957297	0.913239	0.943231	1.000000	0.960889	0.953059
XMR	0.958446	0.923868	0.948560	0.960889	1.000000	0.955704
BTC	0.959274	0.916157	0.945670	0.953059	0.955704	1.000000

3.5 Volatility Calculation

The volatility of a stock is a measurement of the change in variance in the returns of a stock over a specific period of time. It is common to compare the volatility of a stock with another stock to get a feel for which may have less risk or to a market index to examine the stock’s volatility in the overall market. Generally, the higher the volatility, the riskier the investment in that stock, which results in investing in one over another. The volatility is calculated by taking a rolling window standard deviation on the percentage change in a stock.





	BTC	DASH	ETH	LTC	XMR	XRP
2019-07-29	0.289904	0.173392	0.157758	0.206251	0.222613	0.193652
2019-07-30	0.275824	0.171073	0.157174	0.205615	0.221445	0.193594
2019-07-31	0.280595	0.170348	0.152406	0.209138	0.221270	0.190833
2019-08-01	0.281286	0.172046	0.153314	0.205000	0.216447	0.191088
2019-08-02	0.259343	0.163572	0.142394	0.194130	0.208018	0.175766

4. Trading Strategy

Developing a trading strategy is something that goes through a couple of phases, just like when you, for example, build machine learning models: you formulate a strategy and specify it in a form that you can test on your computer, you do some preliminary testing or backtesting, you optimize your strategy and lastly, you evaluate the performance and robustness of your strategy.

4.1 Backtesting

Trading strategies are usually verified by backtesting: you reconstruct, with historical data, trades that would have occurred in the past using the rules that are defined with the strategy that you have developed. This way, you can get an idea of the effectiveness of your strategy, and you can use it as a starting point to optimize and improve your strategy before applying it to real markets. Of course, this all relies heavily on the underlying theory or belief that any strategy that has worked out well in the past will likely also work out well in the future, and, that any strategy that has performed poorly in the past will probably also do badly in the future.

4.2 Portfolio

A collection of investments owned by an investor makes up his or her portfolio. You can have as few as one stock in a portfolio, but you can also own an infinite amount of stocks or other securities.

4.3 Moving Average

A stock's average price-per-share during a specific period is called its moving average. Some common time frames to study in terms of a stock's moving average include 50- and 200-day moving averages.

In detail, we have discussed about

- Relative and log-returns, their properties, differences and how to use each one,
- A generic representation of a trading strategy using the normalized asset weights $w_i\ (t)$ for a set of N tradable asset

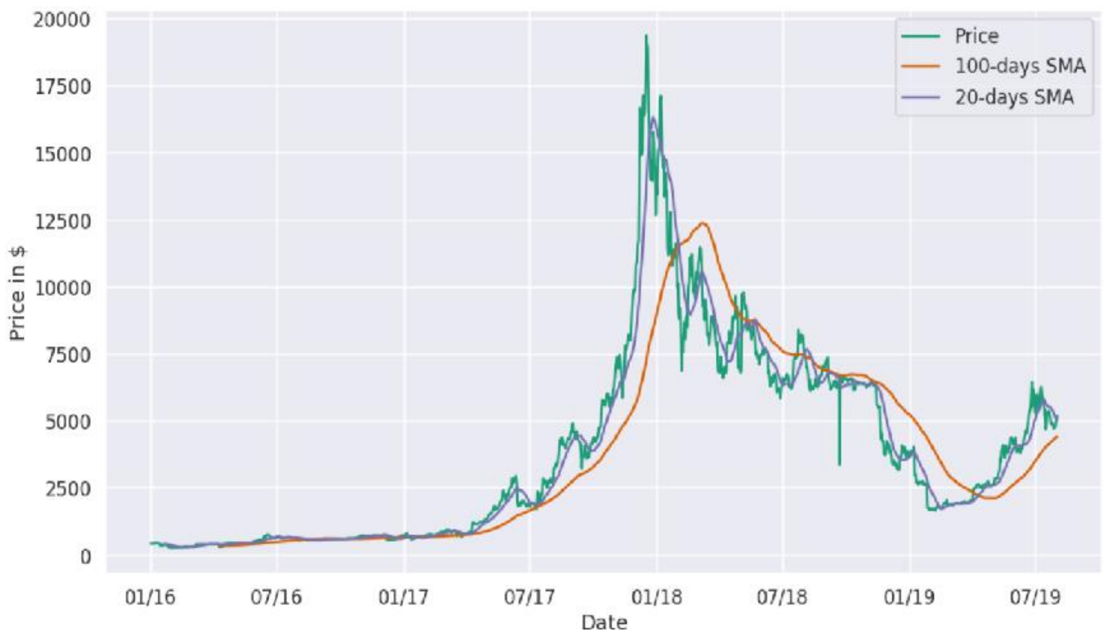
4.3.1 Simple Moving Average (SMA)

One of the oldest and simplest trading strategies that exist is the one that uses a moving average of the price (or returns) timeseries to proxy the recent trend of the price.

The idea is quite simple, yet powerful; if we use a (say) 100-day moving average of our price time-series, then a significant portion of the daily price noise will have been "averaged-out". Thus, we can observe more closely the longer-term behavior of the asset.

Let us, again, calculate the rolling *simple moving averages (SMA)* of these six timeseries as follows. Remember, again, that when calculating the M days SMA, the first M−1 are not valid, as M prices are required for the first moving average data point.

Let us plot the **SMA** for last 4 years for these six timeseries for **BITCOIN** stock, to get a feeling about how these behave.



It is straightforward to observe that SMA timeseries are much less noisy than the original price timeseries. However, this comes at a cost: SMA timeseries lag the original price timeseries, which means that changes in the trend are only seen with a delay (lag) of L days.

4.3.2 Exponential Moving Average (EMA)

For a SMA moving average calculated using M days, the lag is roughly M/2days. Thus, if we are using a 100 days SMA, this means we may be late by almost 50 days, which can significantly affect our strategy.

One way to reduce the lag induced using the SMA is to use the so-called Exponential Moving Average (EMA), defined as:

$$\begin{aligned} \mathrm{EMA}(t) &= (1-\alpha)\,\mathrm{EMA}(t-1) + \alpha\, p(t) \\ \mathrm{EMA}(t_0) &= p(t_0) \end{aligned}$$

where p(t) is the price at time t and α is called the decay parameter for the EMA. α is related to the lag as

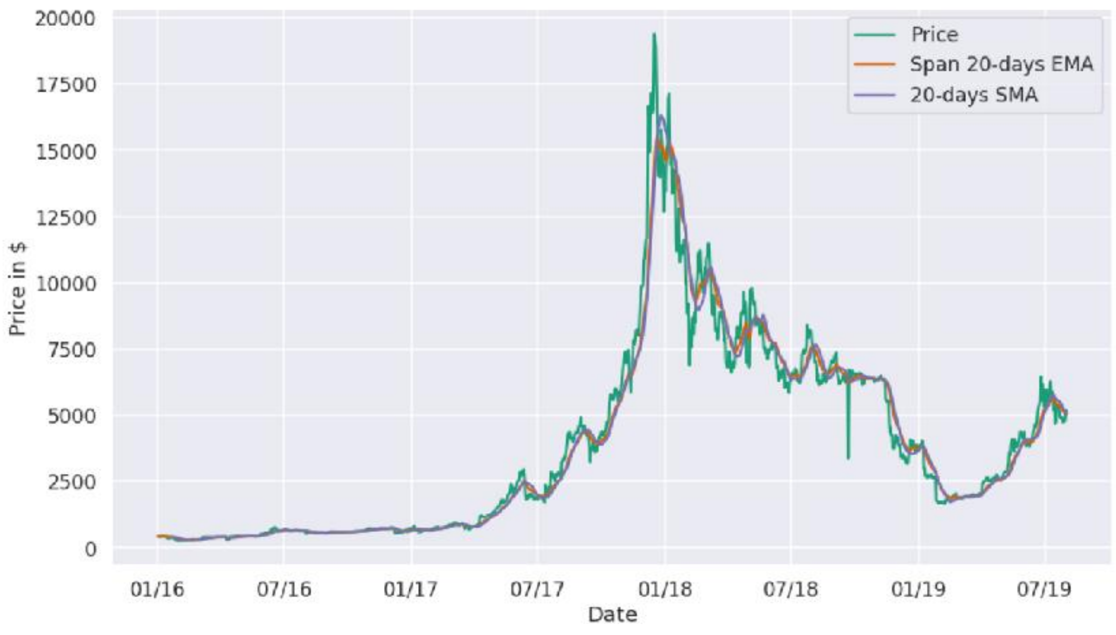
$$\alpha = \frac{1}{L+1}$$

and the length of the window (span) M as

$$\alpha = \frac{2}{M+1}$$

The reason why EMA reduces the lag is that it puts more weight on more recent observations, whereas the SMA weights all observations equally by 1/M. Using Pandas, calculating the exponential moving average is easy. We need to provide a lag value, from which the decay parameter α is automatically calculated. To be able to compare with the short-time SMA we will use a span value of 20.

Let us plot the **EMA** for last 4 years for these six timeseries for **BITCOIN** stock, to get a feeling about how these behave.



4.3.3 Building Moving Average Trading Strategy

Let us attempt to use the moving averages calculated above to design a trading strategy. Our first attempt is going to be relatively straightforward and is going to take advantage of the fact that a moving average timeseries (whether SMA or EMA) lags the actual price behavior.

Bearing this in mind, it is natural to assume that when a change in the long-term behavior of the asset occurs, the actual price timeseries will react faster than the EMA one. Therefore, we will consider the crossing of the two as potential trading signals.

- When the price timeseries p(t) crosses the EMA timeseries e(t), we will close any existing short position and go long (buy) one unit of the asset.
- When the price timeseries p(t) crosses the EMA timeseries e(t), we will close any existing long position and go short (sell) one unit of the asset.

How about the weight $w_i\ (t)$?

Since, at this point, we are not interested yet in position sizing, we will assume that we use all our funds available to trade asset i. We will also assume that our funds are split equally across all 6 assets

Based on these assumptions, our strategy for each of the assets i, i=1,…,6 can be translated as follows:

- Go long condition: If $p_i(t) > e_i(t)$, then $w_i(t)$ =1/6
- Go short condition: If $p_i(t) < e_i(t)$, then $w_i(t)$ =−1/6

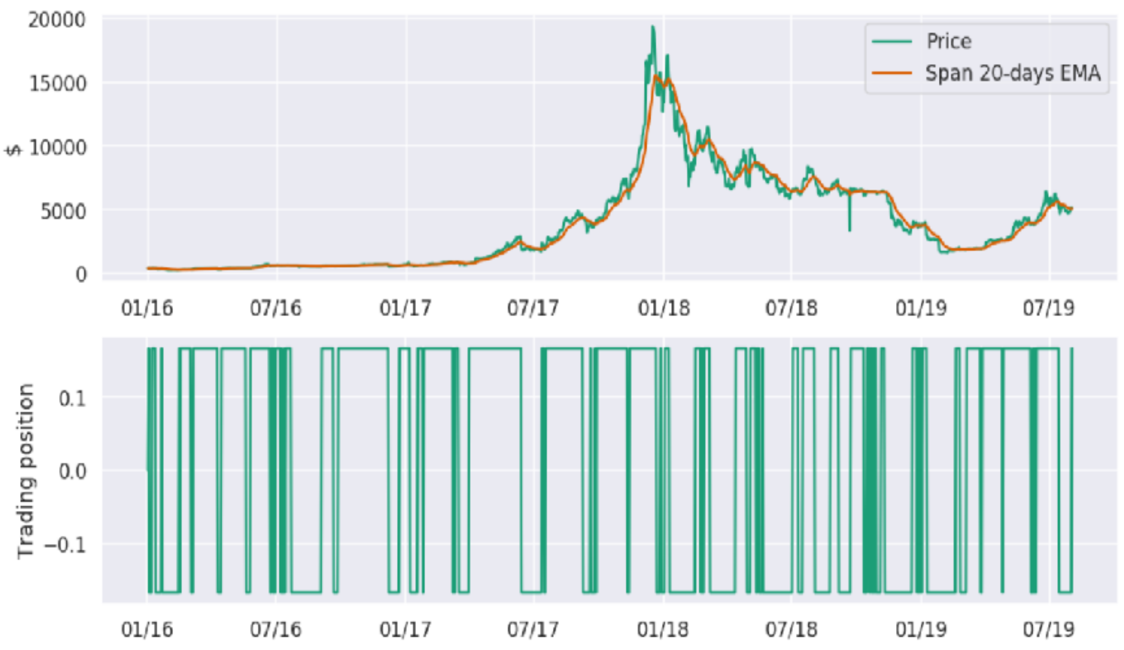
Whenever, the trade conditions are satisfied, the weights are 1/6 because 1/6 of the total funds are assigned to each asset and whenever we are long or short, all the available funds are invested.

Fix the caveat

Before seeing the performance of this strategy, let us focus on the first day t_o when the price timeseries p (t_o) crosses above and EMA timeseries $e_i(t_o)$. However, bear in mind that p (t_o) is the price of the asset at the close of day t_o. For this reason, we will not know that p (t_o) > e_i (t_o) until the close of the trading day. This is easily corrected for by lagging our trading positions by one day.

```
# Taking the difference between the prices and the EMA timeseries
trading_positions_raw = close - ema_short
# Taking the sign of the difference to determine whether the price or the EMA is greater and then multiplying by 1/6
trading_positions = trading_positions_raw.apply(np.sign) * 1/6
# Lagging our trading signals by one day.
trading_positions_final = trading_positions.shift(1)
```

Let us examine what the timeseries and the respective trading position look like for one of our assets, **BITCOIN**.



Now that the position our strategy dictates each day has been calculated, the performance of this strategy can be easily estimated.

4.3.3.1 Log-return strategy

Our strategy trades each asset separately and is agnostic of what the behavior of the other assets is. Whether we are going to be long or short (and how much) in BTC is in no way affected by the other five assets. With this in mind, the daily log-returns of the strategy for each asset i, $r_i^s(t)$ are calculated as:

$$r_i^s(t) = w_i(t) r_i(t)$$

where $w_i(t)$ is the strategy position on day t which has already been attained at the end of trading day t−1.

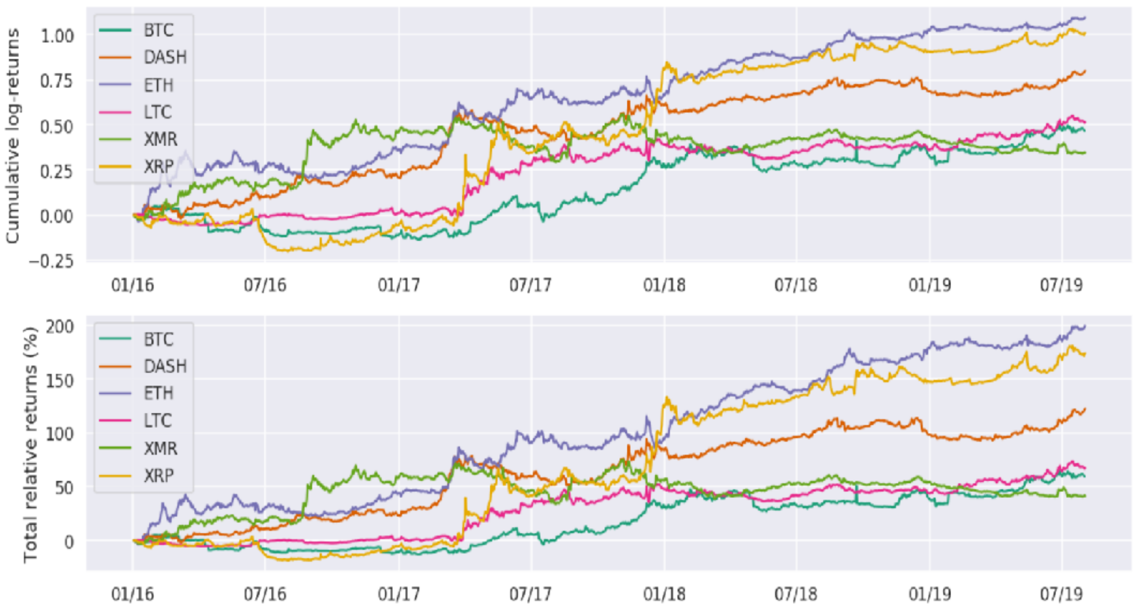
To get all the strategy log-returns for all days, one needs simply to multiply the strategy positions with the asset log-returns.

```
# Log returns - First the logarithm of the prices is taken and the the difference of consecutive (log) observations
asset_log_returns = np.log(close).diff()
strategy_asset_log_returns = trading_positions_final * asset_log_returns
```

Remembering that the log-returns can be added to show performance across time, let us plot the cumulative log-returns and the cumulative total relative returns of our strategy for each of the assets.

```
# Get the cumulative log-returns per asset
cum_strategy_asset_log_returns = strategy_asset_log_returns.cumsum()

# Transform the cumulative log returns to relative returns
cum_strategy_asset_relative_returns = np.exp(cum_strategy_asset_log_returns) - 1
```



What is the Total Return of the Strategy?

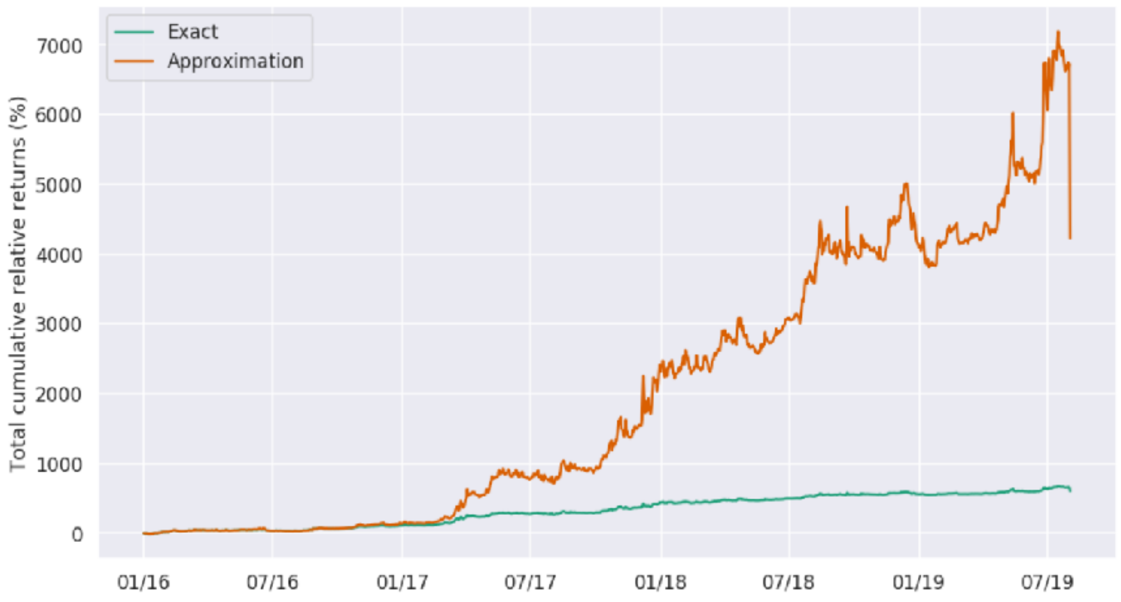
We can only add relative returns to calculate the strategy returns. Thus, an alternative way is to simply add all the strategy log-returns first and then convert these to relative returns.

```
# Total strategy relative returns. This is the exact calculation.
cum_relative_return_exact = cum_strategy_asset_relative_returns.sum(axis=1)

# Get the cumulative log-returns per asset
cum_strategy_log_return = cum_strategy_asset_log_returns.sum(axis=1)

# Transform the cumulative log returns to relative returns. This is the approximation
cum_relative_return_approx = np.exp(cum_strategy_log_return) - 1
```

Let us examine how good this approximation is:



As we can see, for relatively small time-intervals and as long the assumption that relative returns are small enough, the calculation of the total strategy returns using the log-return approximation can be satisfactory. However, when the small-scale assumption breaks down, then the approximation is poor. Therefore, what we need to remember the following:

- Log-returns can and should be added across time for a single asset to calculate cumulative return timeseries across time.
- However, when summing (or averaging) log-returns across assets, care should be taken. Relative returns can be added, but log-returns only if we can safely assume, they are a good-enough approximation of the relative returns.

The overall, yearly, performance of the strategy (**Backtesting**):

```
Total portfolio return is: 604.91%
Average yearly return is: 47.34%
```

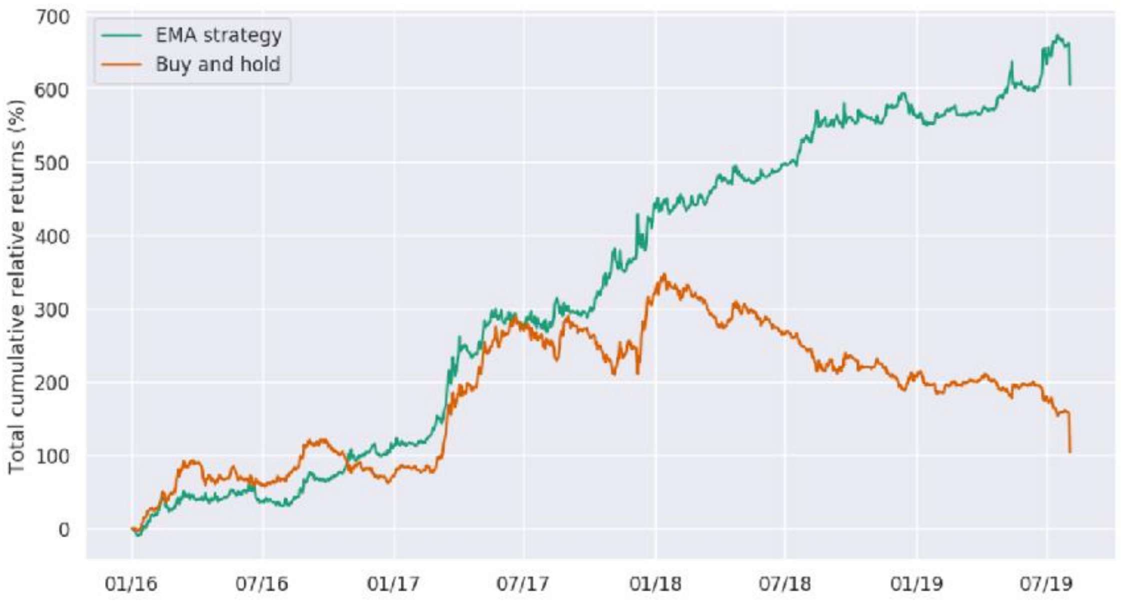
4.3.3.2 EMA strategy

The exponential moving average strategy is a classic example of how to construct a simple EMA crossover system. With this exponential moving average system, we're not trying to predict the market. We're trying to react to the current market condition, which is a much better way to trade.

The advantage of our trading strategy stands in the exponential moving average formula. It plots a much smoother EMA that gives better entries and exits.

The overall, yearly, performance of the strategy (**Backtesting**):

```
Total portfolio return is: 104.43%
Average yearly return is: 15.25%
```



4.3.3.3 Which Strategy is Better?

This is not a simple question for one to answer at this point. When we need to choose between two or more strategies, we need to define a metric (or metrics) based on which to compare them.

In addition, we observe in this last graph that the performance of the two strategies is not constant across time. There are some periods when one outperforms the other and other periods when it is not. So, a second question that naturally arises is how we mitigate the risk to be "tricked" by a good backtesting performance in a given period.

4.4 Deep Learning

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

4.4.1 How does deep learning attain such impressive results?

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

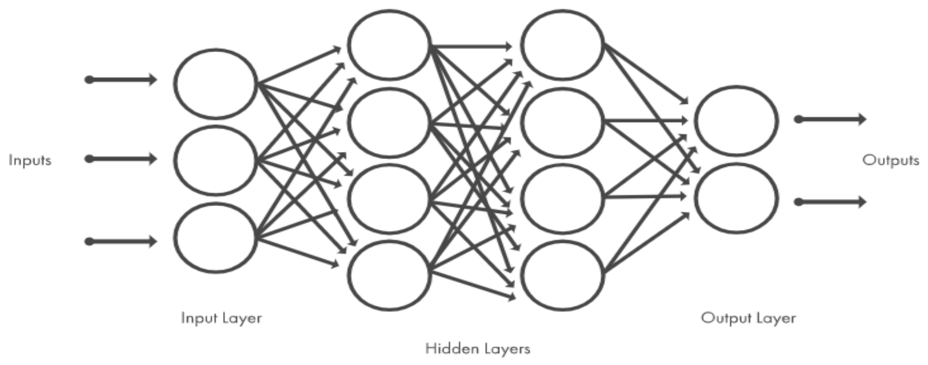
- Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.
- Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

4.4.2 How Deep Learning Works?

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.



4.4.3 What's the Difference Between Machine Learning and Deep Learning?

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs "end-to-end learning" – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network.

A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

4.4.4 Choosing Between Machine Learning and Deep Learning

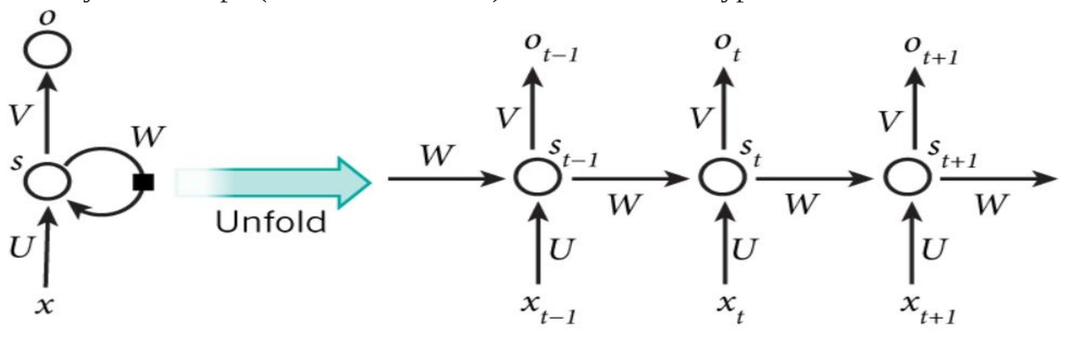
When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labeled data. If you don't have either of those things, it may make more sense to use machine learning instead of deep learning.

4.5 Recurrent Neural Networks (RNN) Architecture

Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies. These algorithms take time and sequence into account, they have a temporal dimension.

Research shows them to be one of the most powerful and useful type of neural network, alongside the attention mechanism and memory networks. RNNs are applicable even to images, which can be decomposed into a series of patches and treated as a sequence.

Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:



4.5.1 Recurrent neural network (RNN) - Trading Strategy

In this experiment, outcome will help investors to buy and sold shares on the predictions based on the AI made, for one week or so, to see if they can make a decent profit. To start with I put my portfolio (BTC and LTC) totally in the hands of the AI.

I made a basket out of shares that I either own, have owned, or which randomly popped up in my watchlist. The stock tickers are:

['BTC','LTC','ETH','XRP','DASH','XMR']

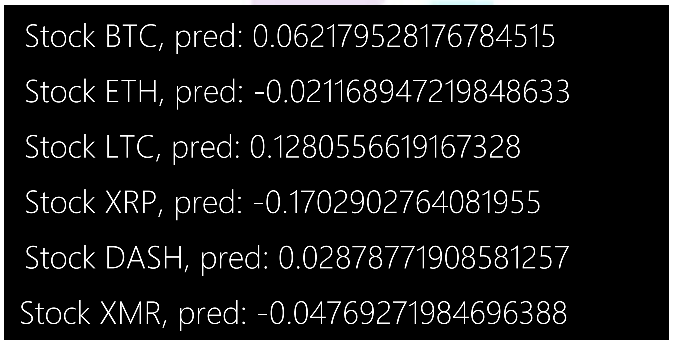
I'll evaluate performance both daily and by comparing the value of whatever I have for next day.

This AI tool only makes predictions for the shares in the basket, so that investors can predict and trading during the week. Additionally, this tool suggests sell the (predicted) worst performing share that you own in the basket and buy the (predicted) best performing share.

Day 1- Saturday 3 August 2019

Full disclosure: the experiment began for the third or fourth time today. In the previous six weeks, I started several times intrepidly, only to discover that, in varying ways, I was pre-processing the data wrong and feeding it to the AI wrong. Sometimes the error was obvious (such as it made the same predictions day after day, due to my reshaping the data wrong), and sometimes more secret.

The AI's predictions for tomorrow:

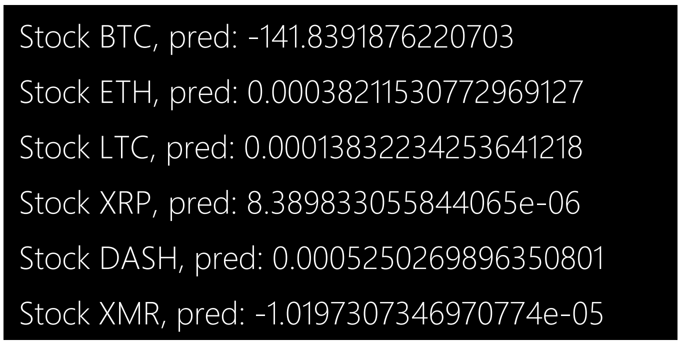


```
Stock BTC, pred: 0.0621795281767845l5
Stock ETH, pred: -0.021168947219848633
Stock LTC, pred: 0.1280556619167328
Stock XRP, pred: -0.1702902764081955
Stock DASH, pred: 0.028787719085812 57
Stock XMR, pred: -0.04769271984696388
```

Day 2- Sunday 4 August 2019

Bitcoin share performance (downwards) is weird to me. I think what most traders don't get is that they aren't a gaming hardware company, but rather, they own the deep learning market.

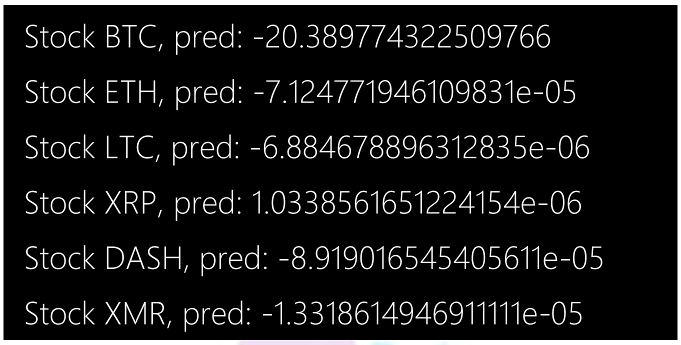
The AI’s predictions for tomorrow:



```
Stock BTC, pred: -141.8391876220703
Stock ETH, pred: 0.00038211530772969127
Stock LTC, pred: 0.00013832234253641218
Stock XRP, pred: 8.389833055844065e-06
Stock DASH, pred: 0.0005250269896350801
Stock XMR, pred: -1.0197307346970774e-05
```

Day 3- Monday 5 August 2019

The AI’s predictions for tomorrow:



```
Stock BTC, pred: -20.389774322509766
Stock ETH, pred: -7.124771946109831e-05
Stock LTC, pred: -6.884678896312835e-06
Stock XRP, pred: 1.0338561651224154e-06
Stock DASH, pred: -8.919016545405611e-05
Stock XMR, pred: -1.3318614946911111e-05
```

4.5.1.1 How I built the AI tool?

You can view what I did as a Jupyter notebook on GitHub . The high level is:

- Mangle the data (I used training data from QUANDL and POLONIEX, which gives 6 years of historical data to train on when you use their private API — the free public API only gives 1 year.
- Add a few features. What this AI tool is ultimately predicting is a transformed version of the stock’s gain between today and tomorrow.
- Build a Keras deep recurrent network to be the predictor. I’m back in to Tensorflow since my last article on how it can be deployed at the frontend, plus Keras has the benefit that you don’t need to spend so long reshaping data as PyTorch.
- Train the model on *all* of the data for the stocks in my basket (one-hot encoding the stock symbol). That makes it a general model, learning general stock market trends across all the stocks in my basket.
- Freeze some of the layers, pop off the network head, add a new head, and replicate that model for as many stocks as I have in my basket.
- Fine-tune each of those models, with its individual head, to the stock.
- Still working on creating a job to do this every midnight at East Coast time.

4.5.1.2 High Level Learnings

- It seems that more, and bigger, fully connected layers are not required since RNN layer renders them somewhat unnecessary.
- Feature transformations are important. With extended feature engineering a neural network can learn to do that itself. However, using one-hot encoding for datetime features was very useful, as was taking the log of very large values.
- While creating features, we should make sure they are as different as possible and use those features extensively even if they are not highly correlated. For example, I was doing 2- and 3-day price windows; it was useless. 20- and 50-day windows worked much better.
- Keep batch size the same between training and inference. This is because the Keras LSTM layer is only stateful within a batch. Thus, if you use 32 batch size in training, it'll learn to predict outputs based on 32 history steps. Don't then thwart it by giving it fewer steps: it needs to 'warm up' the same amount, even if that intuitively makes you worry about overfitting.
- I had to admit there was probably data leakage happening — for example, somewhere I was probably taking the mean over the full range of data instead of just the training set.
- Better logging! I didn't save anything from day-to-day. I wish I had. It became challenging to create a combined picture after the experiment ended because I hadn't fully recorded each day's actions and prices.
- I ran out of memory issue with more than 9 cryptocurrencies. My recommendation would be run this AI in Google cloud with multiple cryptocurrencies.
- Based on last few days experiment, RNN didn't provide me expected result and I'd like to recreate this work without the non-parallelizability and vanishing gradients over long patterns that RNNs suffer from.

4.5.1.3 RNN a better strategy?

In my view, especially since doing this experiment — it's impossible to predict stocks accurately. High frequency trading might yield predictive signals from second to second, but from day to day. Moreover, I don't want to trade day-to-day even, may be bi-weekly or from month to month.

4.6 Learning – Supervised, Unsupervised, and Reinforcement

The name **supervised** comes from the fact that we learn from the known answers, which were obtained from some supervisor who has provided us with those labeled examples.

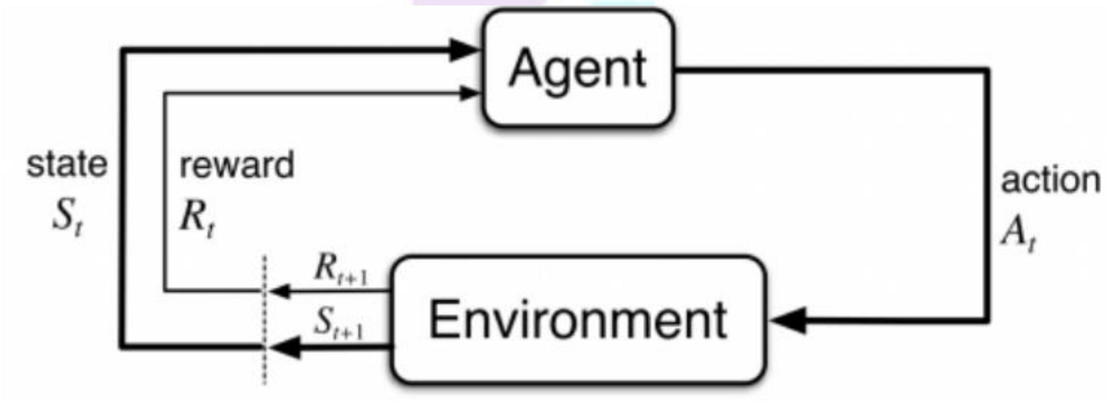
At the other extreme, we have the so-called **unsupervised** learning, which assumes no supervision that has no known labels assigned to our data. The main objective is to learn some hidden structure of the dataset at hand. One common example of such an approach to learning is the clustering of data.

RL (Reinforcement Learning) is the third camp and lays somewhere in between full supervision and a complete lack of predefined labels. On the one hand, it uses many well-established methods of supervised learning such as deep neural networks for function approximation, stochastic gradient descent, and backpropagation, to learn data representation. On the other hand, it usually applies them in a different way.

4.6.1 Deep Reinforcement Learning

RL, being a subfield of ML, borrows lots of its machinery, tricks, and techniques from ML. Basically, the goal of RL is to learn how an agent should behave when it is given imperfect observational data.

Reinforcement Learning problem can be formulated as a Markov Decision Process (MDP). We have an agent acting in an environment. Each time step t the agent receives as the input the current state S_t, takes an action A_t, and receives a reward R_{t+1} and the next state S_{t+1}. The agent chooses the action based on some policy π: $A_t = \pi(S_t)$. It is our goal to find a policy that maximizes the cumulative reward $\sum R_t$ over some finite or infinite time horizon.



Reward:

In RL, it's just a scalar value we obtain periodically from the environment. It can be positive or negative, large or small, but it's just a number. The purpose of reward is to tell our agent how well they have behaved. The purpose of a reward is to give an agent feedback about its success, and it's an important central thing in RL. Basically, the term reinforcement comes from the fact that a reward obtained by an agent should reinforce its behavior in a positive or negative way. Reward is local, meaning, it reflects the success of the agent's recent activity, not all the successes achieved by the agent so far.

Of course, getting a large reward for some action doesn't mean that a second later you won't face dramatic consequences from your previous decisions.

What an agent is trying to achieve is the largest accumulated reward over its sequence of actions. e.g.

- *Financial trading*: An amount of profit is a reward for a trader buying and selling stocks.

The agent:

An agent is somebody or something who/which interacts with the environment by executing certain actions, taking observations, and receiving eventual rewards for this. In most practical RL scenarios, it's our piece of software that is supposed to solve some problem in a more-or-less efficient way. e.g.

- *Financial trading*: A trading system or a trader making decisions about order execution

The environment:

The environment is external to an agent, and its communication with the environment is limited by rewards (obtained from the environment), actions (executed by the agent and given to the environment), and observations (some information besides the rewards that the agent receives from the environment). We discussed rewards already, so let's talk about actions and observations.

Actions:

Actions are things that an agent can do in the environment. In RL, we distinguish between two types of actions: discrete or continuous. Discrete actions form the finite set of mutually exclusive things an agent could do, such as move left or right. Continuous actions have some value attached to the action, such as a car's action steer the wheel having an angle and direction of steering.

Observations:

Observations are pieces of information that the environment provides the agent with, which say what's going on around them. It may be relevant to the upcoming reward (such as seeing a bank notification saying, you have been paid) or not. Observations even can include reward information in some vague or obfuscated form, such as score numbers on a computer game's screen. The reward is the main force that drives the agent's learning process. If the reward is made wrong, noisy, or just slightly off-course of the primary objective, then there is a chance that training will go in a wrong way.

- *Financial trading*: Here the environment is the whole financial market and everything that influences it. This is a huge list of things such as the latest news, economic and political conditions, weather, food supplies, and Twitter trends. Even your decision to stay home today can potentially indirectly influence the world financial system. However, our observations are limited to stock prices, news, and so on. We don't have access to most of the environment's state, which makes trading such a nontrivial thing.

4.6.2 Why RL for trading?

First let's create a profitable trading strategy using a supervised learning approach (e.g. moving average, RNN). Then we will see what the problems around this are, and why we may want to use Reinforcement Learning techniques.

First question, what price do we predict? As we have seen in data analysis, there is not a "single" price we are buying at. The final price we pay depends on the volume available at different levels of the order book, and the fees we need to pay. A naive thing to do is to predict the *mid-price*, which is the mid-point between the *best bid* and *best ask*.

The next question is time scale. Do we predict the price of the next trade? The price at the next second? Minute? Hour? Day?

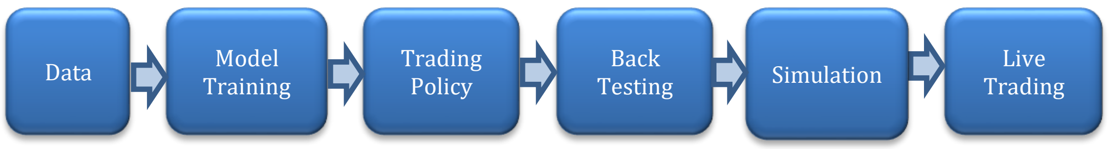
Let's assume the BTC price is $5,000 and we can accurately predict that the "price" moves up from $5,000 to $5,050 in the next minute. It looks like you can make $50 of profit by buying and selling? But it's not that easy. Let see why?

- If you plan to buy 1.0 BTC most likely we may be forced to buy 0.5 BTC at $5,000 and 0.5 BTC at $5,010, for an average price of $5,005 because the order book does not have the required volume. On GDAX, we also pay taker fee, which corresponds to roughly $30.
- Suppose we place the sell order at predicted price $5,050. Let's say it's now at $5,045 because of Market volatility or by the time the order is delivered over the network, the price has shifted already. Like above example, we most likely forced to sell 0.5 BTC are $5,045 and 0.5 BTC at $5,040, for an average price of $5,042.5. Then we pay another taker fee, which corresponds to roughly $30.

So, how much money have we made? -5005 - 30 - 30 + 5,042.5 = -$22.5. Instead of making $50, we have lost $22.5, even though we accurately predicted a large price movement over the next minute.

In the above example there were three reasons for this: No liquidity in the best order book levels, network latencies, and fees, none of which the supervised model could consider.

Here is a typical workflow for a trading system using **supervised learning**:



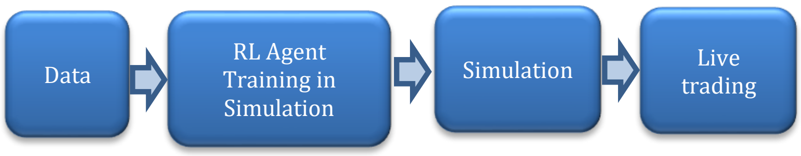
Why do I think this process in not effective?

- Iteration cycles are slow. Step 1-3 are largely based on intuition, and you don't know if your strategy works until the step 4 is done, force you to start from scratch.

In fact, every step comes with the risk of failing and forcing you to start from scratch.

- Simulation comes too late. You do not explicitly consider environmental factors such as latencies, fees, and liquidity until step 4. Shouldn't these things directly inform your strategy development or the parameters of your model?
- Policies are developed independently from supervised models even though they interact closely. Supervised predictions are an input to the policy.
- Policies are simple. They are limited to what humans can come up with.

The case for **Reinforcement Learning**



- In the traditional strategy development approach, we must go through several steps. We need to train supervised model, come up with a rule-based policy using the model, backtest the policy and finally assess its performance through simulation. Reinforcement Learning allows for end-to-end optimization and maximizes rewards.
- Instead of needing to hand-code a rule-based policy, Reinforcement Learning directly learns a policy. There's no need for us to specify rules and thresholds such as "buy when you are more than 80% sure that the market will move up".
- Reinforcement Learning agents are trained in a simulation, and that simulation can be as complex as you want, considering latencies, liquidity and fees, we don't have this problem.
- We could take this a step further and simulate the response of the other agents in the same environment, to model impact of our own orders.
- Typically, simulators ignore this and assume that orders do not have market impact. However, by learning a model of the environment and performing rollouts using techniques like a Monte Carlo Tree Search (MCTS), we could consider potential reactions of the market (other agents). By being smart about the data we collect from the live environment, we can continuously improve our model.

4.6.3 Deep Reinforcement Learning Trading Strategy

If there's a real trend in the numbers, irrespective of the fundamentals of a stock, then given a enough function approximator (like a deep neural network) reinforcement learning should be able to figure it out.

4.6.3.1 Setting up the problem

In RL important for success is framing the problem. This AI tool:

- Would be given a starting amount of cash
- Would learn to maximize the value of its portfolio (shares and cash) at the end of a given time interval
- At each time step, could buy or sell shares or do nothing

- If it buys more than the money it has available, the run ends; if it sells more than it has, the run ends. It must learn the rules of the game, as well as how to play it well.

To keep things manageable, I decided only to work with a pair of presumably somewhat correlated stocks: BTC and LTC. Initially I wanted the bot to be able to pick how many shares it buys or sells at each timestep, and this led me down into a situation of "continuous action space" reinforcement learning.

As usual, we have a neural network for the AI tool and an environment that reacts to the AI tool's actions, returning a reward each step depending how good the AI tool's action was.

Pseudocode like this:

```
For 1..n episodes:
  Until environment says "done":
    Run the current state through the network
    That yields:
      a set of "probabilities" for each action
      a value for how good the state is

    Sample from those probabilities to pick an action

    Act upon the environment with that action
    Save the state, action, reward from environment to a
    buffer

 Adjust each saved reward by the value of the terminal
 state discounted by gamma each timestep

 Loss for the episode is the sum over all steps of:
    The log probability of the action we took * its reward
    The value of that action compared to what was expected

 Propagate the loss back and adjust network parameters
```

You can see that it does not attempt to optimize for the choice of action — we do not ever know what the objectively correct action would have been — but rather:

- The degree of certainty about each action, given its resulting reward
- The degree of surprise about each reward, given the state

As the network becomes more certain about a given action (*p -> 1*), loss decreases (*ln(p) -> 0*) and it learns slower. And on the flipside, unlikely actions that were sampled and which led unexpectedly to big rewards, produce a much bigger loss. Which, in backprop/gradient descent, causes the optimizer to increase the probability of that unlikely action in future.

The environment code and much of the training loop code is not very interesting; if you are interested, it's in the notebook on GitHub.

State: The state I finally opted for was a vector consisting of:

```
[BTC holdings,
LTC holdings,
cash,
current timestep's BTC opening price,
current timestep's LTC opening price,
current portfolio value,
past 5-day average BTC opening price,
past 5-day average LTC opening price]
```

The AI to learn the fundamentals of a BUY signal and a SELL signal, rather than just learning to repeat the time series. Each time the environment reset, set the environment to start at a different random timestep. Also, use a different random stride through the data and defined "done" as a different random number of steps in the future.

While I think this is the right approach, unfortunately, it made it basically impossible for the RNN to learn anything useful because it had no idea where it was in the series each time. So, I abandoned the varying start point, stride, and sequence length, and held them the same during all the training episodes.

4.6.3.2 Train and Test split

At this point, everything looks great, and you want to test the model on a different time range of stock data to ensure it generalized well. After all, we'd like to be able to deploy this on our trading accounts in the future and make lots of money.

What I did next was to go back earlier, grab the stock data from then to present; I then split it into roughly 2/3 train, and reserved the last 1/3 for the test.

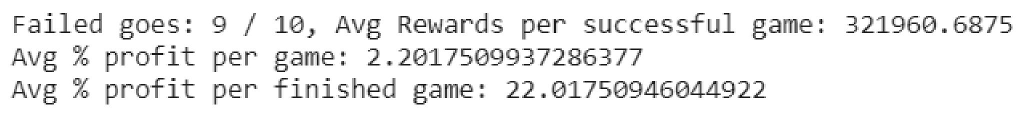
To aid the AI, I set the stride to 1 and sequence length to the length of the training data, so it had as much sequential data as possible to train. Now, enough action-reward pairs were processed so that it was too slow to continue CPU, so by adding 5.cuda() statements I was able to move much of the work to the GPU.

Now, you can imagine the chances of the AI getting to the end of a sequence of ~1000 moves to get a meaningful reward. Starting random, it must play follow rules it doesn't even know yet and not make a single false move. Even if it does that, it's useless to us unless it made some trades and a profit during that time. Some hours' wasted training time confirmed this not to be a viable way to proceed.

The environment I've made here was not too difficult to build. I just had to start the agent off with lots of BTC and LTC shares and a significant amount of cash. Train it for a while, then reduce the starting shares/cash, train it again, and so on. Eventually, it learned to respect the rules.

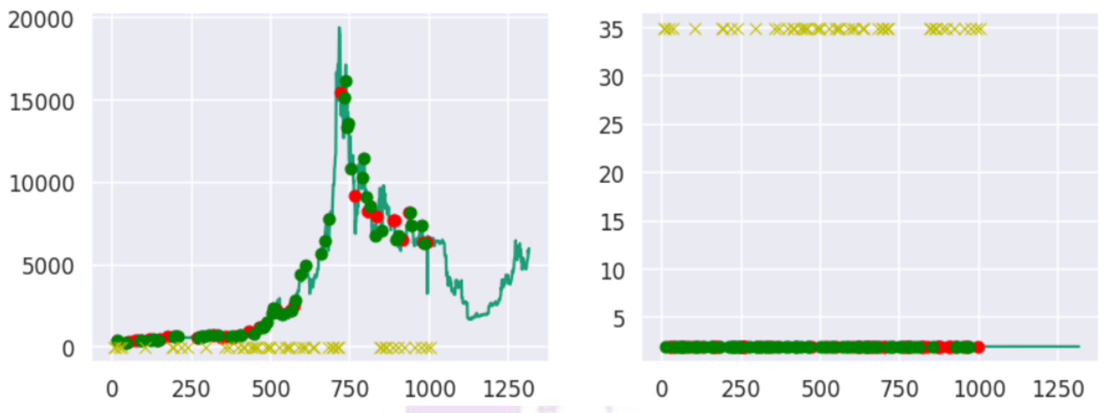
4.6.3.3 RL Result

Results on the test data, which the bot had not trained on, looks great. The bot learned to not cheat or bankrupt itself, most of the time, and it was able to make a significant profit: here you can see 220 % returns.

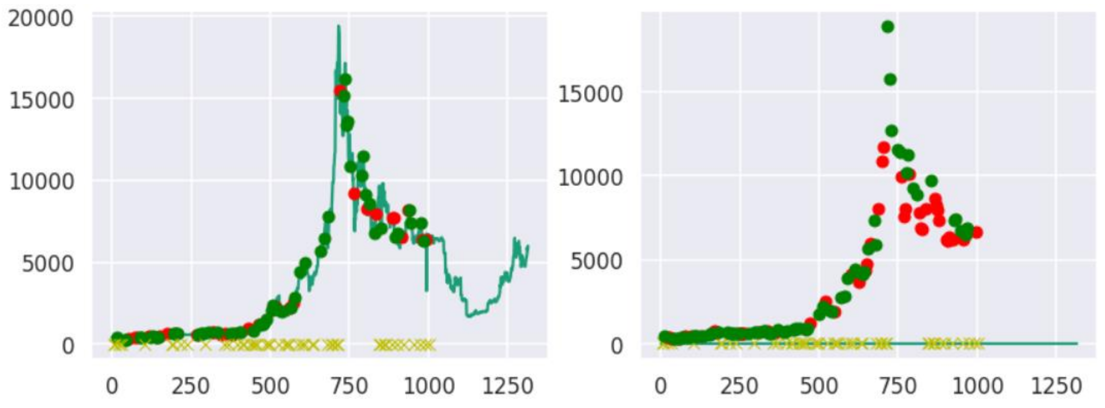


```
Failed goes: 9 / 10, Avg Rewards per successful game: 321960.6875
Avg % profit per game: 2.2017509937286377
Avg % profit per finished game: 22.01750946044922
```

Here's one of its "trading strategies", graphed. Red means buy, green means sell, and left is BTC and right LCT. The yellow crosses are timesteps where the bot chose to take no action.



And here's a longer run, showing buys and sells in the context of the original, non-transformed share prices.



4.6.3.4 Conclusion

I'm overwhelmed by RL's potential as a universal optimizer that can learn to function by itself, but not sure to get working well — in high-entropy environments like online trading.

We may get rich or lose money, but we did figure out a state-of-the-art and AI technique, and surely those are two things that will prove correlated over time

5. Improvement

- Add data from more cryptocurrencies to the analysis.
- Adjust the time frame and granularity of the correlation analysis, for a more fine- or coarse-grained view of the trends.
- Search for trends in trading volume and/or blockchain mining data sets. The buy/sell volume ratios are likely more relevant than the raw price data if you want to predict future price fluctuations.
- Add pricing data on stocks, commodities, and fiat currencies to determine which of them correlate with cryptocurrencies (but please remember the old adage that "Correlation does not imply causation").
- Quantify the amount of "buzz" surrounding specific cryptocurrencies using Event Registry, GDLELT, and Google Trends.
- Use your analysis to create an automated "Trading Bot" on a trading site such as Poloniex or Coinbase, using their respective trading APIs.
- Be careful: a poorly optimized trading bot is an easy way to lose your money quickly.
- Run this AI in Google Cloud.

6. Source Code

- You can view what I did as a Jupyter notebook on GitHub

7. Plotly Charts

- Cryptocurrency Correlations in 2019
  https://plot.ly/~gautam_bhowmick/422/
- Cryptocurrency Correlations in 2018
  https://plot.ly/~gautam_bhowmick/420/
- Cryptocurrency Correlations in 2017
  https://plot.ly/~gautam_bhowmick/418/
- Cryptocurrency Correlations in 2016
  https://plot.ly/~gautam_bhowmick/416/
- Cryptocurrency Prices (USD)
  https://plot.ly/~gautam_bhowmick/414/
- Bitcoin Price (USD)
  https://plot.ly/~gautam_bhowmick/414/

References

- Google Cloud Platform Big Data and Machine Learning Fundamentals
  https://www.coursera.org/learn/gcp-big-data-ml-fundamentals

- Neural Networks and Deep Learning
  https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning

- Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
  https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning

- Structuring Machine Learning
  https://www.coursera.org/learn/machine-learning-projects?specialization=deep-learning

- Convolutional Neural Networks
  https://www.coursera.org/learn/convolutional-neural-networks?specialization=deep-learning

- Sequence Models
  https://www.coursera.org/learn/nlp-sequence-models

- How Bitcoin Works in 5 Minutes (Technical)
  https://www.youtube.com/watch?v=l9jOJk30eQs&list=PLBQ0biANmxYJ9I1owRK8FOIDN5FP742TI&index=2&t=0s

- How Bitcoin Works Under the Hood
  https://www.youtube.com/watch?v=Lx9zgZCMqXE&list=PLBQ0biANmxYJ9I1owRK8FOIDN5FP742TI&index=3&t=0s

- TED Don Tapscott "The Blockchain will Change EVERYTHING!"
  https://www.youtube.com/watch?v=yK6Ldefgbl0&index=9&list=PLBQ0biANmxYJ9I1owRK8FOIDN5FP742TI

- Blockchain 101 - A Visual Demo
  https://www.youtube.com/watch?v=_160oMzblY8&list=PLBQ0biANmxYJ9I1owRK8FOIDN5FP742TI&index=3

- What is Bitcoin Mining and is it Profitable?
  https://99bitcoins.com/bitcoin-mining/

- What is the Bitcoin Mining Block Reward?
  https://www.bitcoinmining.com/what-is-the-bitcoin-block-reward/

- Introduction to Learning to Trade with Reinforcement Learning by Denny Britz
  http://www.wildml.com/2018/02/introduction-to-learning-to-trade-with-reinforcement-learning/

- Applying Deep Reinforcement Learning for Trading
  https://www.youtube.com/watch?v=Pka0DC_P17k&t=24s

- Crypto-assets are data science heaven
  https://www.slideshare.net/jesusmrv/crypto-assets-are-a-data-science-heaven-rev

- Turtle Trading: A Legend of Trading
  https://www.investopedia.com/articles/trading/08/turtle-trading.asp

- Python for finance: Algorithmic trading
  https://www.datacamp.com/community/tutorials/finance-python-trading

- Self-driving car of Finance
  https://iknowfirst.com/deep-reinforcement-learning-building-a-self-driving-car-in-financial-world3

- Trading crypto for beginners
  https://medium.com/@macropus/trading-crypto-for-beginners-10659165b44a

- MIT 6.S091: Introduction to Deep Reinforcement Learning (Deep RL)
  https://www.youtube.com/watch?v=zR11FLZ-O9M

- Lecture 14 | Deep Reinforcement Learning
  https://www.youtube.com/watch?v=lvoHnicueoE

- RL— Introduction to Deep Reinforcement Learning
  https://medium.com/@jonathan_hui/rl-introduction-to-deep-reinforcement-learning-35c25e04c199

- Using reinforcement learning to trade Bitcoin for massive profit
  https://towardsdatascience.com/using-reinforcement-learning-to-trade-bitcoin-for-massive-profit-b69d0e8f583b

- Stock Market Prediction by Recurrent Neural Network on LSTM Model
  https://blog.usejournal.com/stock-market-prediction-by-recurrent-neural-network-on-lstm-model-56de700bff68

- Predict Stock Prices Using RNN
  https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html

- Using the latest advancements in deep learning to predict stock price movements
  https://towardsdatascience.com/aifortrading-2edd6fac689d

- Implementing Moving Averages in Python
  https://towardsdatascience.com/implementing-moving-averages-in-python-1ad28e636f9d

- Building a Moving Average Crossover Trading Strategy Using Python
  https://www.jasonlee.mobi/projects/2018/5/18/building-a-momentum-trading-strategy-using-python

- Deep Reinforcement Learning Hands-on – ***Maxim Lapan***

- Crypto assets, The innovative Investor's Guide to Bitcoin and Beyond – ***Chris Bruniske & Jack Tatar***