Capstone Project

Data Scientist Nanodegree

13th February 2021

J.H

Build an automatica trading bot by leveraging ML techniques

Abstract

This project successfully created a simple automatic trading strategy using Python for the crypto currency Ethereum. This project has provided evidence for the ability to execute High-Low strategy on crypto exchanges to produce gains. The raw data use is for 1minute candlestick data for the ETH/USD pair from kaggle[1]. The method trained three XGBRegressor machine learning models on 15minutes intervals (but only two were used for this proof of concept) to provide the predicted future values. For the testing data, ETH/USD values for the month of December 2020 (1st - 27th December 2020) was used, an outcome of around 5% return was achieved.

This report will discuss the strategy used, evaluate the results, and provide advice on further fine tuning to the models and trading strategies for future improvements.

Project statement

Build a proof of concept for the ability to automatically execute High-Low strategy on crypto exchanges, leveraging machine learning models.

Introduction

The practice of using machines to provide and follow through with a trading strategy has been done for decades; with billions of investments into the technology to facilitate this, such as the example of the Hibernian Atlantic's $300Million contract to reduce the speed of data delivery by 5.3 milliseconds[2]. *(could the money be spent on something better? Who am I to judge…)*

Historically, projects of this nature can only be done in a research facility with access to a bloomberg terminal as well as the coding behind it to automatically place trades. However, the rise in crypto-currency as well as crypto-exchanges, facilitated by open source programming languages (e.g. Python) and increase in computing power; all of these elements have resulted in the possibilities of doing such a project, with nothing more than a good computer and an internet connection.

This project picks ETH/USD pair for no particular reason (other than it is always considered to be the second to Bitcoin, and in the author's perspective, better than Bitcoin); this project can easily be completed using other trading pairs as long as enough historical data is present for training models.

1 - Binance 1-min dataset, Kaggle, https://www.kaggle.com/jorijnsmit/binance-full-history

2 - Stock Trading Is About to Get 5.2 Milliseconds Faster, Bloomberg, https://www.bloomberg.com/news/articles/2012-03-29/stock-trading-is-about-to-get-5-dot-2-milliseconds-faster

The trading strategy is a High-Low strategy, to explain, at a specific interval, each execution on an exchange typically has a high, as well as a low price, which indicates, during that interval, what was the highest price or the lowest price.

Fig 1. Example of an image for the J D Wetherspoon stock on Yahoo finance showing prices for Open, High, Low, Close etc.

XGBRegressor machine learning models were used to train to predict the next X minutes Low and High, a Buy-Order is subsequently placed at the Low price, once executed, a Sell-Order is then placed at the High price (and hope it executes). The final amount will result in a small profit gain.

Data Analysis

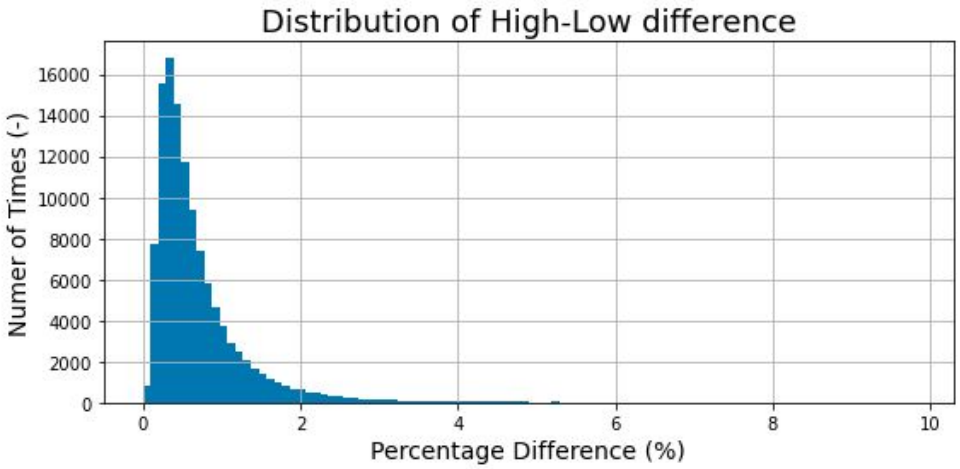

Fig 2. Plot of the percentage difference between high and low price point of 15-minute candlesticks.

Table 1. Basic statistics of the distribution

	count	mean	std	min	25%	50%	75%	max
high_low_dif	117466.0	0.710512	0.676803	0.0	0.324162	0.51434	0.851519	9.796008

The above analysis shows that it is possible to buy an asset at low price and sell for a small profit within 15minutes. The mean of the difference is 0.71%, with the median being 0.514%. One thing to consider, is that the statistics do not show whether it is a Low >> High, or High >> Low scenario (where the high price is hit before the low price, aka, when the price is dropping). Regardless, it is possible to reduce the risk of this happening during the execution strategy (more will be discussed in the methodology section).

Methodology

Data Processing

The processing of the data consists of:

1) Obtaining raw data with high, low, open and close value
2) Group by 15minuetes intervals
3) Calculate rolling averages
4) Pick a base value to normalise all the data on (for this project, the base value is 1Hour rolling average)

Data Training

Two machine learning models based on XGBoost Regression models are used to predict the high and low price given training data.

```
[26]: pipeline_xgb_low = Pipeline([('multi_clf', XGBRegressor(n_estimators=100,
                                                             learning_rate=0.1,
                                                             gamma=0,
                                                             subsample=0.75,
                                                             colsample_bytree=1,
                                                             max_depth=7) )
                                   ])

      pipeline_xgb_high = Pipeline([('multi_clf', XGBRegressor(n_estimators=100,
                                                               learning_rate=0.1,
                                                               gamma=0,
                                                               subsample=0.75,
                                                               colsample_bytree=1,
                                                               max_depth=7) )
                                   ])
```

Fig 3. Screenshot of the pipeline used to train the 2 models.

Exchange Imitation

Two main Python classes were written to replicate the trading environment. First class imitates what an exchange would do, it stores information such as the current price, open orders etc. It also executes open orders. The second class is a decision class, it takes in and stores raw data, processes them, and then predicts the next low and high value from the historical data.

Both classes work together to automate the trading process. The historical data was from 17th August 2017 until 1st December 2020. The data used to perform trades on was from 1st December 2020 until 27th December 2020. For December 2020, roughly 5% gain was obtained.

Results

For the month of December, three successful trades were executed. The graph is shown below:

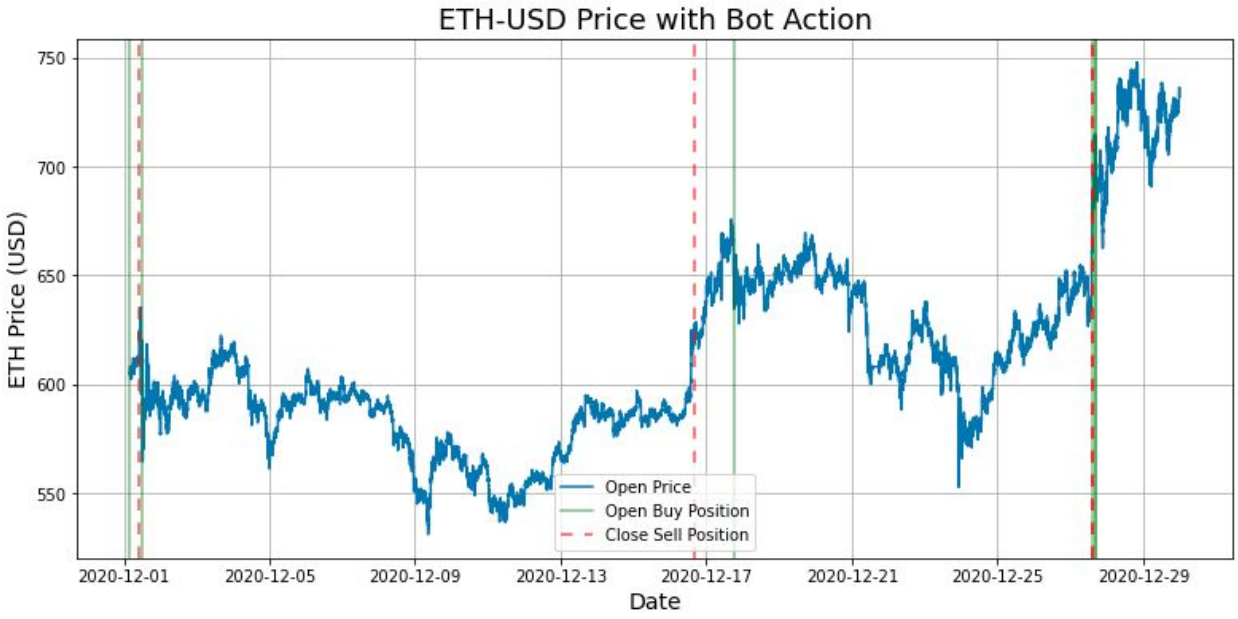

Fig 4. Shows the events of execution overlaying the price change for the month of December

It appears the model has successfully captured some opportunities. However, there are rather a few opportunities that have been missed. Further investigation is needed to understand if the missed opportunities were due to having an open trade at the time.

Discussion and Refinement

Due to the fact that this project only seeks to provide proof of concept, many steps were done to a crude quality. Therefore, much improvement can be done. Each step of the project will be broken down and suggest possible improvements.

1. Raw data

The raw data was obtained on Kaggle. In real situations, an API will be used to obtain data. The main thing needs to be considered here is the timing aspect. There will always be a slight delay between obtained value and real value. Therefore, any trading strategy needs to have enough protection against this. This factor was taken into consideration when chosen to have a 15-minute grouping rather than a more frequent strategy.

2. Trading Decisioning

This is perhaps the most important aspect. Given the fact that the percentage difference between high and low prices are roughly 1% for most of the time, as well as the volatility of the asset, a small error could hurt the end result given the frequency here. A few strategies should be looked into to protect against errors. One of which has already been implemented was the idea of a mark-down - for the price predicted, reduce the buying price by x percentage. This essentially reduces the effect of the error (of course, this also reduces the possible execution frequency).

3. Fees

One thing to note, is that there is a cost to buy-sell on crypto-exchanges. This fee varies from exchanges to exchanges (currently, 0.3% seems to be the standard). Therefore, any execution must have a raw return of at least 0.3%.

4. Trading Strategy

This perhaps is the most important part to refine, as the result shows, many opportunities (dips) seem to have been missed, due to having an order in place, therefore, perhaps a limit on the length of open position needs to be applied.

Conclusion

This project coded up a trading bot using ML techniques and tested its ability against 1 month of data. The result was not as good as expected due to many opportunities that seemed to have been missed. However, a framework to test and deploy different strategies have been successfully coded up. Therefore, proving a building ground for more advanced analytical techniques to be tested.