Sean Casey
Udacity Machine Learning Capstone Proposal – v2
9/4/17

Domain background

This project is going to look at a novel approach to gauging whether it is a 'good time to buy' for a particular asset in the cryptocurrency trading space. In 2017, cryptocurrency trading and investing has seen a massive influx in both capital and prospective commodities to invest in, thanks to stories of incredible gains and increasing mass media attention.

One of the biggest arenas in this space has been the ICO (initial coin offering) boom. The first rounds of ICOs were almost entirely unregulated and individual investment was uncapped, which led to a great majority of the funding coming from "whales" with deep pockets and access to develop the most efficient bots to be able to buy as much stake in an ICO as they want, before normal investors are able to. This led to some historic sellouts, such as the Basic Attention Token selling out $35m of tokens in 30 seconds. New ICOs are responding with different mechanisms for achieving a more well-distributed crowdsale, whether that means placing individual quotas on contribution, identity verification, or novel distribution practices. The EOS ICO is implementing the lattermost.

The EOS ICO model is truly novel: they are dividing their crowdsale into a mini-crowdsale each of 350 days. Every day, 2 million EOS tokens is up for grabs, but **the contribution amount is uncapped**. When the day's contribution period is finished, the number of tokens is divided proportionately based on contribution amounts by all participants. This means that on one day, 2 million EOS tokens might go for $2.7 million and the next day the same number of tokens might go for $2.1 million.

Problem statement

In this project, I am going to attempt to create a model that can predict whether or not it is a "good day" to contribute to the EOS crowdsale. This poses a

binary classification problem, i.e., is it a good day to invest, or isn't it? I want to generate this guess when there are 5 minutes left in the contribution period.

The dataset for this problem will be structured components from the contribution periods that have elapsed thus far in the crowdsale. I will curate the dataset using a couple APIs (from etherscan.io and coinmarketcap.com).

The raw output variable here will be a 'yes' or 'no.' This is essentially the answer to the question: 'will the value of the EOS token at the end of the crowdsale be at least 5% greater than its market value?'

Datasets and inputs

There is no existing dataset for this problem. I will create the dataset myself by 1) accessing etherscan.io's API to create a log of most of the primary data and 2) identifying the rough price of the EOS token and of Ether immediately before the crowdsale ended via coinmarketcap's API (https://coinmarketcap.com/). One datapoint will consist of: value of EOS token, value of Ether (the contribution is made with Ethereum), the # of contributed Ether 30 minutes, 15 minutes, 10 minutes, and 5 minutes before the end of a crowdsale period, and the # of contributed Ether at the end of the crowdsale period.

The problem is that right now, there are only 68 datapoints (i.e., the crowdsale has been going on for 68 days). The hope will be that as the crowdsale goes on (and there is more data available), the model's accuracy will improve.

Solution statement

Because of the lack of datapoints, I am going to do an exploratory analysis of what works with the limited data, including several types of models: Decision trees, logistic regression, random forest, and Long Short Term Memory neural network. In the long run, as more data becomes available, the LSTM model will potentially be

the most beneficial to making accurate predictions, because the model needs to adapt to new conditions, as other people who are contributing to the crowdsale deploy their capital more efficiently (e.g., improve machine learning algorithms of their own).

I will use these four methods to train models based on this data that I have described above. The y value (that we are hoping to predict) will be a Yes/No (1/0) indication of whether or not EOS is a 'good buy'. I'll define a 'good buy' as being as beating the market value by at least 5%.

The X values will be all the remaining values outlined in the section above. The idea with capturing contributions at different time intervals (30 min, 15 min, 10 min, 5 min) is that maybe we can take into account the 'velocity' of contribution during those last critical minutes at the end of the sale. I might add in an additional feature of 3 minutes to see how much that helps the model.

Rather than using simple cross validation, I will use sklearn's TimeSeriesSplit framework. It is important to evaluate relative time frames in their contexts, rather than blindly testing a highly offset cross validation data set.

Benchmark model

There isn't really a benchmark model (as in a model that another project has used to accomplish a similar result) that I am aware of. Instead, I will use a simple sanity check, a "NoChange" classifier, to establish a baseline precision metric (see section below about why I'm using precision). The NoChange classifier will simply take the previous item in the dataset's y value as the correct y value for the subsequent item. If yesterday was a "good day" to invest in EOS, tomorrow will be as well.

Evaluation metrics

It is especially important for this project that the model is excellent with precision (rather than recall). Because we're only looking for potential trading opportunities, we don't care if the model misses out on identifying some of them (false negatives), only that the ones it does identify are actually good buys (true positives) and not bad buys (false positives). For the sklearn-based models, I will get the precision_score from sklearn.metrics For the neural network, I will calculate precision in my code, as keras does not offer precision as a metric.

I will also use 'accuracy' of the models as an additional indicator (from sklearn.metrics and from keras' evaluate() method).

Project design

1) I'll curate the data sets using the etherscan.io and coinmarketcap APIs, retrieving:

- eos_market_value = value of EOS token
- eth_value = value of Ethererum
- eth_contributed_n (e.g., eth_contributed_30) = number of Ethererum contributed before the end of the crowdsale at:
    - 30 minutes
    - 15 minutes
    - 10 minutes
    - 5 minutes
    - 3 minutes
- eth_contributed_total = number of Ethererum contributed at the completion of the crowdsale

2) This data will be separated into dataset X and label Y, where Y is computed as follows:

value_per_token = eth_value * eth_contributed_total / 2000000

if eos_market_value / value_per_token >= 1.05, then y=1, else y=0

The data will then be split into training, testing, and cross validation datasets. I will use sklearn's TimeSeriesSplit to ensure that cross validation occurs in a way that respects the time series.

3) I will train a decision tree classifier using sklearn. I will measure the precision of the model using sklearn.metics.precision_score. I will tweak the parameters to achieve a maximal score. I will also take note of the recall score and accuracy.

4) I will repeat step 3 using logistic regression and random forest models.

5) I will build a Long Short Term Memory (LSTM) neural network with keras. I will use a Sequential model, starting with only 1 LSTM hidden layer, and will see how well the model can be trained before attempting to add in a second hidden layer.

6) I will calculate the precision of the model in code, and will implement keras' `accuracy` metric and its `evaluate()` method to get baseline evaluation metrics.

7) I will iterate through different options for hidden layers and activations, and try adding and removing the various time-dependent features (e.g., eth_contributed_30).

8) I will compare the results of the different models and offer an analysis of how they performed and why some performed better than others.