Names: Brian Kominick, Noah Foltz

Title: Swing Trading with Bitcoin using Random Decision Forests and K Nearest Neighbor

Aims: Classification to find a predicted price movement (up, down, or neutral) of bitcoin based upon our selected attributes. We will use real-time data from the cryptocurrency exchange Bitmex and trade on this exchange as well. They have a testnet, so it is perfectly suited to our aims.

Data: We will be using price, time, exchange volume, exchange position (shorts versus longs) and order book data. This is all real-time data and comes from Bitmex's API.

We will be using random decision forests to predict price rise or fall based on the attributes listed above. We will be using the scikit learn randomForestClassifier in order to predict price change. In addition to random forests, we'll make use of sklearn.neighbors as another model for predicting the price movement. These two methods differ in their classification processes and one or the other may have a higher accuracy.

We will evaluate both algorithms on only one thing: the class attribute accuracy. If the price movement was predicted correctly, the algorithm's accuracy increases. Bitmex has the ability to short and long positions, which means we can take advantage of both sides of price movement and the algorithm's ability to predict it.

Our theoretical framework is weak from the beginning because if machine learning could easily and accurately predict price, everyone would be making money off of trading. Also, we will need to learn about the scikit random forest module and its corresponding parameters, as well as fit our data into the scikit learn data classification template. In addition to analyzing the data, we will be collecting real-time prices from Bitmex and will need to decide how and when to incorporate them into decision making. This data will need to be preprocessed to filter out noise from many small transactions and those that are quickly removed after being posted. In our nearest neighbor classification, we need to be wary not only about marking sure data is correctly scaled but also in selecting/weighting our attributes.

Ref:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
http://scikit-learn.org/stable/modules/neighbors.html
http://scikit-learn.org/stable/modules/ensemble.html#forest
https://www.blopig.com/blog/2017/07/using-random-forests-in-python-with-scikit-learn/
https://www.bitmex.com/app/apiOverview
http://scikit-learn.org/stable/modules/neighbors.html#unsupervised-nearest-neighbors
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors
https://tampub.uta.fi/bitstream/handle/10024/96376/GRADU-1417607625.pdf?sequence=1