Predicting Bitcoin Price Variations using Bayesian Regression

Reference paper: http://arxiv.org/pdf/1410.1231.pdf

In this project, you will be tasked with predicting the price variations of bitcoin, a virtual cryptographic currency. These predictions could be used as the foundation of a bitcoin trading strategy. To make these predictions, you will have to familiarize yourself with a machine learning technique, Bayesian Regression, and implement this technique in Python.

Submission Instructions:
Make your changes directly to the **bitcoin.py** file and submit ONLY this file on Moodle.

Project Setup

The Python packages that you will need for this project are `numpy`, `pandas`, `sklearn`, and `statsmodels`. To install these, simply use the pip installer `sudo pip install X` or, if you are using Anaconda, `conda install X`, where X is the package name.

Datasets

We have provided the datasets you are to use in the *data* folder. For your convenience, we have already done some cleaning of the data. The original raw data can be found here: http://api.bitcoincharts.com/v1/csv/. The datasets from this site have three attributes: (1) time in epoch, (2) price in USD per bitcoin, and (3) bitcoin amount in a transaction (buy/sell). However, only the first two attributes are relevant to this project.

To make the data to have evenly space records, we took all the records within a 20 second window and replaced it by a single record as the average of all the transaction prices in that window. Not every 20 second window had a record; therefore those missing entries were filled using the prices of the previous 20 observations and assuming a Gaussian distribution. The raw data that has been cleaned is given in the file `dataset.csv`.

Finally, as discussed in the paper, the data was divided into a total of 9 different datasets. The whole dataset is partitioned into three equally sized (50 price variations in each) subsets: `train1`, `train2`, and `test`. The train sets are used for training a linear model, while the test set is for evaluation of the model. There are three csv files associated with each subset of data: `*_90.csv`, `*_180.csv`, and `*_360.csv`. In `_90.csv`, for example, each line represents a vector of length 90 where the elements are 30 minute worth of bitcoin price variations (since we have 20 second intervals) and a price variation in the 91st column. Similarly, the `*_180.csv` represents 60 minutes of prices and `*_360.csv` represents 120 minutes of prices.

Project Requirements

You are expected to **implement the Bayesian Regression model** to predict the future price variation of bitcoin as described in the reference paper. You should first read the reference paper. If you cannot understand all of the math in the beginning, that is ok. The main parts to focus on are Equation 6 and the *Predicting Price Change* section.

You will be provided a basic stub for the project in **`bitcoin.py`**. Look for the tag **`# YOUR CODE GOES HERE`** and fill in the missing parts to complete the code. You are tasked with the following:

1. Compute the price variations (Δp^1, Δp^2, and Δp^3) for train2 using train1 as input to the Bayesian Regression equation (Equations 6). Make sure to use the similarity metric (Equation 9) in place of the Euclidean distance in Bayesian Regression (Equation 6).
2. Compute the linear regression parameters (w_0, w_1, w_2, w_3) by finding the best linear fit (Equation 8). Here you will need to use the `ols` function of statsmodels.formula.api. Your model should be fit using Δp^1, Δp^2, and Δp^3 as the covariates. Note: the bitcoin order book data was not available, so you do not have to worry about the rw_4 term.
3. Use the linear regression model computed in Step 2 and Bayesian Regression estimates, to predict the price variations for the test dataset. Bayesian Regression estimates for test dataset are computed in the same way as they are computed for train2 dataset – using train1 as an input.
4. Once the price variations are predicted, compute the mean squared error (MSE) for the test dataset (the test dataset has 50 vectors => 50 predictions).

When run, your code should print two things to the console. These are already added into the code. Therefore, the output of running **`python bitcoin.py data/`** should produce:

```
Intercept    -0.307294
deltaP90     14.254037
deltaP180   -25.938119
deltaP360    13.255519
dtype: float64
The MSE is 0.953199
```

IMPORTANT HINTS:

- The equation 6 for computing price variation.
  - The call of equation 6 returns one value of the Δp^i series each time.
- Since there is no randomness in the model to be constructed, the final results should be at least close to those above.