Assignment 1

Develop a forecasting model of the bitcoin price that performs better than chance. The model can be either a regression or classification model.

Regression

The model should predict the price at the next timepoint

Classification

The model should predict whether the price will go, for example, up or down. However, it doesn’t have to be a binary prediction (i.e. up/down) but may also be a multi-class prediction (e.g. up/down/no_change). However many classes you chose, you will need to define the class boundaries.

Example of class boundaries:

class	interval
0	-infinity to -0.5%
1	-0.5% to 0.5%
2	0.5% to +infinity

Dataset

The dataset is available in `data.zip` on Canvas. The dataset consists of multiple CSV files with ohlcv (open, high, low, close, volume) data at different frequencies (e.g. every 30 min, 1 hour, 1 day or 1 week). You have to chose which frequency you want to use and whether to predict the opening, high, low or close price.

Validation

Partition the dataset into training and validation sets. Use a 20-point moving average as the baseline prediction to beat.

Regression

Regression will be evaluated with the Mean Squared Error (MSE) and you need to get lower MSE than the baseline prediction.

Classification

Classification will be evaluated with the F1-score and you will need to be a higher F1-score than the baseline prediction (partitioned according to the class boundaries you have defined).

Submission & test

Once submitted, your model will be tested against previously unseen data. Your goal is to beat the 20-point moving average on the test set. In your submission, you will need to specify with data freuqency you used to train your model (5 min, 15 min, 1 hour, …) and whether it predicts the open, high, low or close price. If you decide on a classification model, then you will also need to submit the class boundaries that you used.

The submission should be a `PY` file and the trained model weights. The `PY` file should have to functions: `preprocess` and `model`. If you're using Keras, then you can find information about how to save model weights in their FAQ.

Functions

- `preprocess()`
  - Description: loads and prprocesses the data.
  - Arguments: a list of names of the data (`csv`) files (the list can have length 1).
  - Returns: Numpy arrays `X` and `y`. Inputs and targets, respectively.
- `model()`
  - Description: loads the model weights and returns predictions given some inputs.
  - Arguments: filename of the model weights, and `X` (returned by `preprocess()`).
  - Returns: A numpy array of predictions.

The assignment is due at 2:30 pm on February 3.