Data Streaming project: Trading Data with Kafka

Yao Pacome KOUAME , Pierre LOVITON and Angie MÉNDEZ-LLANOS

MASTER DS IP-PARIS

January 16, 2022

Overview

1. Goals
2. Data
3. Batch models
4. Online models
5. Application

Goals

Objectives

- Using online learning to predict the future value of a given cryptocurrency using Kafka to process the data.
- Comparing the result of the online learning models with a similar batch version.

Data

One observation per minute, retrieved making a request from the **Binance API** by blocks. There are 13 features per observation:

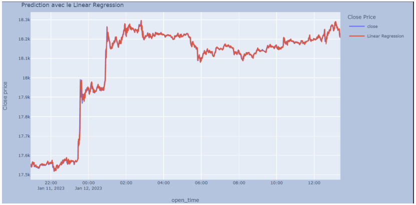
Features

'open time', 'open', 'high', 'low', 'volume', 'close time', 'quote asset volume', 'nb trades', 'Taker buy base asset volume', 'Taker buy quote asset volume', 'Ignore'

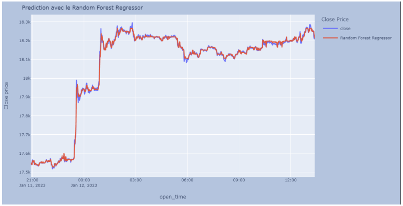
The models were built using the '**close**' variable as target.
Chosen cryptocurrency: **BTCUSDT**.

Batch Models

The data was split in **75-25 train-test** and evaluated on the test data for two different models:



Linear Regression



Random Forest

The performance was evaluated via the **RMSE** and **MAE**:

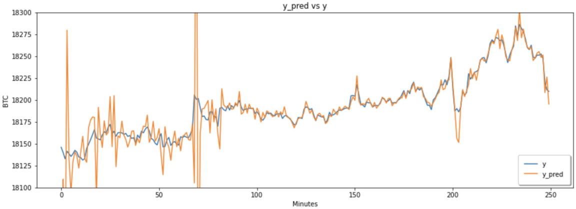
Model	Linear Regression	Randon Forest
RMSE	7.99	9.73
MAE	5.50	7.60

Online models

1. Linear Regression

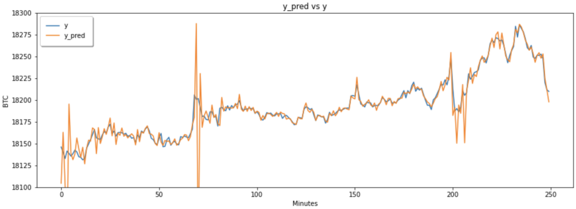
Classic

```
MAE: 3.055636519804319 ; SMAPE: 0.016764278383043646
```



Tuned (intercept lr = 0.25)

```
MAE: 3.3712420839288093 ; SMAPE: 0.018501314848544396
```

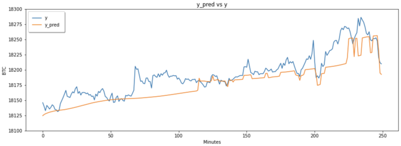


Online models

2. Hoeffding Tree Regressor

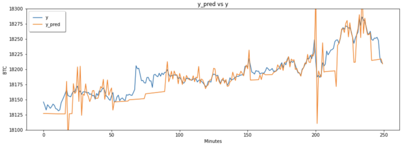
GP: grace period ; MSD: model selector decay

```
grace_period: 10 ; model_selector_decay: 0.5
MAE: 11.985337491365426 ; SMAPE: 0.06574979617678825
```



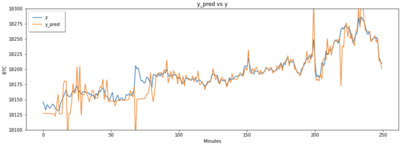
GP: 10 ; MSD: 0.5

```
grace_period: 100 ; model_selector_decay: 0.9
MAE: 12.099897521185268 ; SMAPE: 0.06637331391719749
```



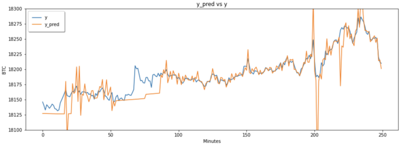
GP: 100 ; MSD 0.9

```
grace_period: 200 ; model_selector_decay: 0.5
MAE: 6.199117941220793 ; SMAPE: 0.03402880062139668
```



GP: 200 ; MSD: 0.5

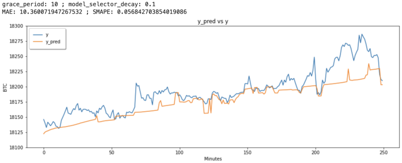
```
grace_period: 200 ; model_selector_decay: 0.9
MAE: 6.111240022371385 ; SMAPE: 0.03354638120709286 4
```



GP: 200 ; MSD: 0.9

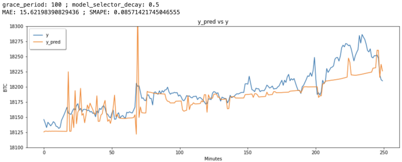
Online models

3. Hoeffding Adaptative Tree Regressor



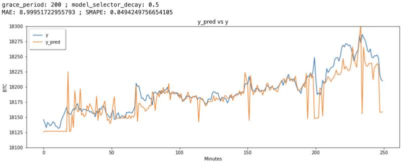
```
grace_period: 10 ; model_selector_decay: 0.1
MAE: 10.360071947267532 ; SMAPE: 0.056842703854019086
```

GP: 10 ; MSD: 0.1



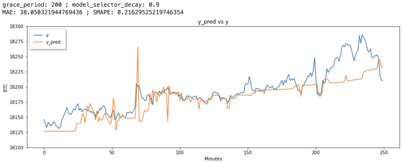
```
grace_period: 100 ; model_selector_decay: 0.5
MAE: 15.62198390829436 ; SMAPE: 0.08571421745046555
```

GP: 100 ; MSD 0.5



```
grace_period: 200 ; model_selector_decay: 0.5
MAE: 8.99951722955793 ; SMAPE: 0.0494249756654105
```

GP: 200 ; MSD: 0.5



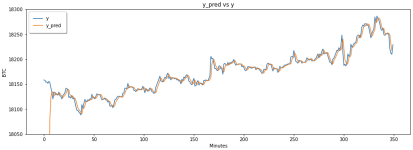
```
grace_period: 200 ; model_selector_decay: 0.9
MAE: 38.050321944769436 ; SMAPE: 0.21629525219746354
```

GP: 200 ; MSD: 0.9

Online models

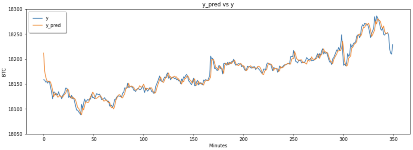
4. **SNARIMAX** horizon: 10 (horizon!= h)

```
h: 1 ; n_iter: 349
MAE: 6.912588 SMAPE: 0.038017
```



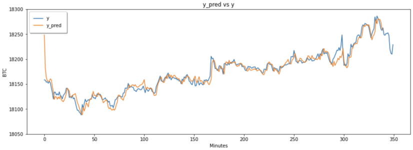
h: 1

```
h: 5 ; n_iter: 345
MAE: 11.651803 SMAPE: 0.064076
```



h: 5

```
h: 10 ; n_iter: 340
MAE: 14.42841 SMAPE: 0.079377
```



h: 10

Kafka

The data was processed using Kafka.

Topics

- Original data : **BTCUSDT-1m-raw**
- Clean data: **BTCUSDT-1m-clean**
- Model outputs
  - Linear model: **model-linear-BTCUSDT**
  - Hoeffding Tree Regressor: **model-HTreg-BTCUSDT**
  - Hoeffding Adaptive Tree Regressor: **model-HATReg-BTCUSDT**
  - SNARIMAX: **model-SNARIMAX-BTCUSDT**

Python scripts

One script for retrieving real-time data, one for cleaning it and one for inoutting the data to each model and saving the results.

Data flow

1. Retrieving the data **ingest-data-BTCUSDT.py**
   - Kafka producer - *BTCUSDT-1m-raw*
2. Clean raw data **clean-data-BTCUSDT.py**
   - Kafka consumer - *BTCUSDT-1m-raw*
   - Kafka producer - *BTCUSDT-1m-clean*
3. Model clean data (identical for each model) **model-linear-BTCUSDT.py**
   - Kafka consumer - *BTCUSDT-1m-clean*
   - Kafka producer - *model-linear-BTCUSDT*