CRYPTO PREDICTOR

1. Introduction

The aim of this project is to create a model that can predict whether the closing price of Bitcoin will be higher or lower the following day and that, by acting on these predictions, the model can get higher returns on investment than simply buying and holding Bitcoin.

I. Why is it relevant?

Bitcoin and other cryptocurrencies have established themselves as relevant inversion vehicles, attracting the interest of mainly small investors, but also some investment funds and regulators.

Cryptocurrencies are highly volatile, very often suffering big swings in prices, which in turn makes them potentially highly profitable investments.

Cryptocurrencies are still relatively new and have a much higher ratio of non-professional investors than stocks. This might mean that, unlike stocks, they might be somewhat predictable.

Potential predictability and high profitability make Bitcoin and other cryptos very interesting targets for machine learning models.

II. Previous works – state of the art

Most existing works about using machine learning for Bitcoin price prediction try to predict the actual future price:

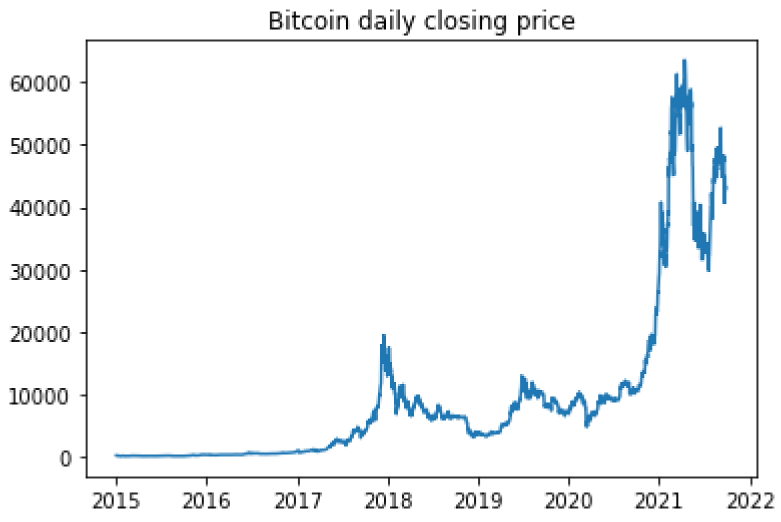
- Short-term bitcoin market prediction via machine learning
- Bitcoin Price Prediction Using Recurrent Neural Networks and LSTM
- Bitcoin price prediction using Machine Learning
- BITCOIN PRICE PREDICTION USING MACHINE LEARNING
- Automated Bitcoin Trading via Machine Learning Algorithms

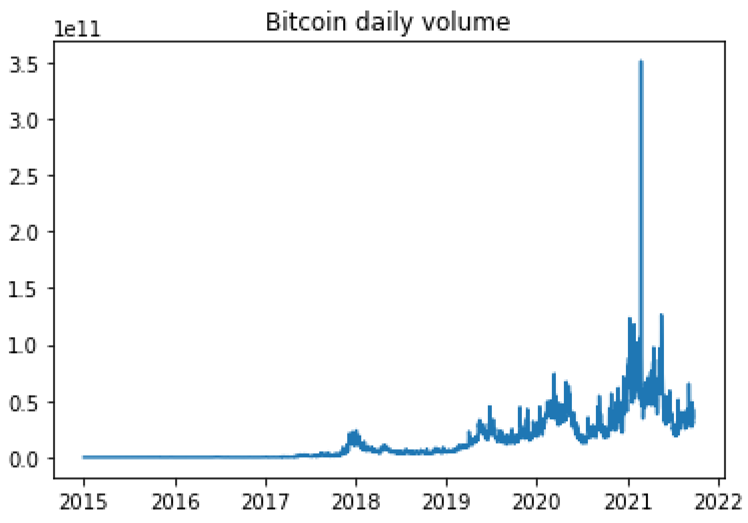
Very few of these works offer results of using the models to try to beat the market. When they do, the results are negative, like in this case: Using machine learning to predict future bitcoin prices

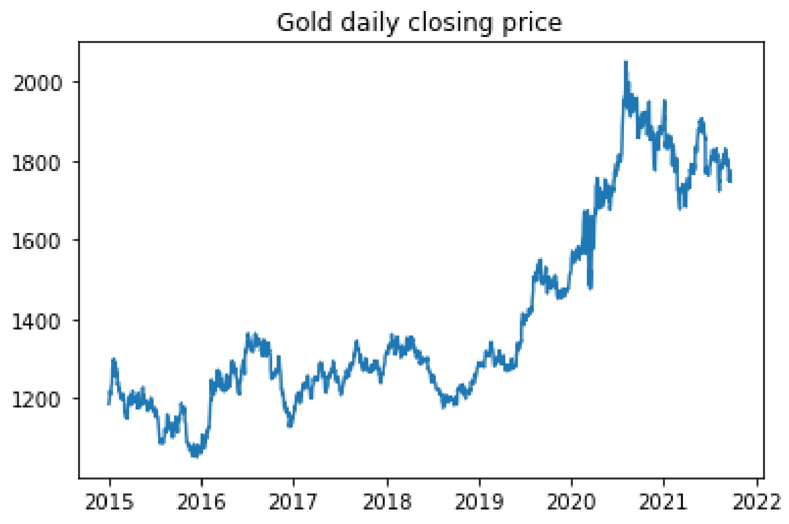
2. Raw data description

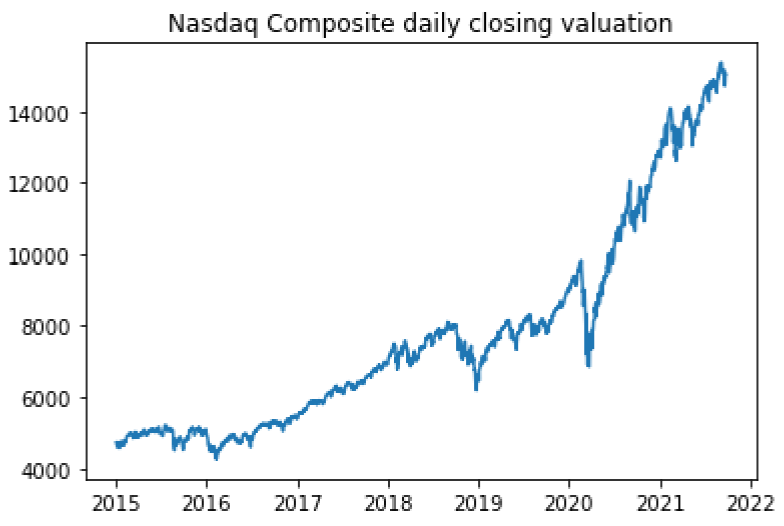
The data used in this project is very simple:

- Daily Bitcoin closing price and volume traded from 01/01/2015 to 26/09/2021. Note that Bitcoin exchanges are always open, so the closing price is just the last price of the day (using UTC time).
- Gold closing price daily from 01/01/2015 to 24/09/2021.
- Nasdaq Composite closing valuation daily from 01/01/2015 to 24/09/2021.









3. Methodology

I. Approach

The whole project follows the approach that in trading is called "technical analysis". This means that the model will try to predict future movements of prices using only past prices. Technical analysis never takes into account the underlying causes of price movements (for example investor sentiment, published articles or the global economy).

In order to assess the models I use a naive model as reference. The naive model is just buying Bitcoin at the beginning of the evaluated period and holding it until the end of that period. The performance of the naive model is the same as the price evolution of Bitcoin.

All the evaluated models use the following assumptions:

- If the model indicates that I should buy or sell, the price is the closing price for that day (UTC time).
- Each transaction (buying or selling) means a fee of 0.15% (a little bit higher than the minimum transaction fee at Binance – 0.1%).

II. Feature engineering

For ARIMA models, no feature engineering is needed, only daily closing prices of Bitcoin.

For classification models the feature engineering used is based on variations:

- For Bitcoin, and for every day, I compute the variation of closing price and volume since the previous day, the previous week (7 days) and the previous month (30 days).
- For Gold and Nasdaq Composite I calculate the same variations as for Bitcoin for closing price. But Gold and Nasdaq Composite don’t have closing prices for every day (because of weekends and holidays), so previously I fill the missing values with the first previous real closing value, i. e. the values for Saturdays and Sundays will be the value for the previous Friday.

I wanted to take into account variations over the previous year for each date, but I also wanted to keep the number of features for each date to a minimum. Therefore, each date/row for classification models includes:

- Variation from the previous day for each of the last 30 days.
- Variation from the previous week for days 30, 37, 44, 51, 58, 65, 72, 79, 86 (counting backwards from the date of the row).
- Variation from the previous month for days 93, 123, 153, 183, 213, 243, 273, 303, 333 (counting backwards from the date of the row).

For Bitcoin, the variations in closing price and volume are included, for Gold and Nasdaq Composite only the variations in closing price are included. In all cases the date of the first row is 01/01/2016, as 1 year of previous data is needed to fill the features for the first row.

III. Model selection

a. ARIMA

As previously stated, for the ARIMA models I use only the closing price of Bitcoin.

The results of running the Augmented Dickey-Fuller test and checking Autocorrelation and Partial Autocorrelation suggest that the best [p, d, q] values are [10, 1, 10].

I divide the data into training, cross validation and testing and use a walk forward validation approach. This means that, every time the model is run, it only predicts the following day (d+1). For predicting day d+2, the real data from day d+1 is added to the training set and the model is trained again.

I try two different decision-making algorithms for the model:

- 2-options algorithm: if the model predicts an increase in price, Bitcoin is bought or held. If the model predicts a decrease in price, Bitcoin is sold or not bought.
- 3-options algorithm: if the increase predicted by the model is higher than the transaction fee (>0.15%), then Bitcoin is bought or held. If the model predicts an absolute variation less or equal to the transaction fee (>=-0.15% and <=0.15%), Bitcoin is held or not bought (ie I maintain the same position). If the decrease predicted by the model is higher than the transaction fee (<-0.15%) , Bitcoin is sold or not bought.

The results for the cross validation set for the [10,1,10] model are negative: the naive model (just hold model) obtains a 63% return, the 2-options algorithm obtains a -33% return and the 3-options algorithm obtains a -28% return.

After that, I decided to try changing the [p,d,q] values. I tried all combinations of p=[9,10,11], d=[0,1] and q=[9,10,11]. The best result was a -5% return on the cross validation set (vs 63% of the naive model). These results indicated that ARIMA models cannot beat the market using this approach, so I decided to discard them.

b. Classification models

For classification models I use variations in price and volume, as explained in 3.II.

I divide the data into training, cross validation and testing and use a walk forward validation approach. This means that, every time the model is run, it only predicts the following day (d+1). For predicting day d+2, the real data from day d+1 is added to the training set and the model is trained again.

I’ve tried a large number of models on the cross-validation sets. These models include all combinations of the following data, transformers and classifiers:

- Data:
  - data_for_use_w_vol: Bitcoin closing price and volume variations
  - data_for_use_w_gold: Bitcoin closing price and volume variations and Gold closing price variations.
  - data_for_use_w_nasdaq: Bitcoin closing price and volume variations and Nasdaq Composite closing price variations.
  - data_for_use_w_all: Bitcoin closing price and volume variations, Gold closing price variations and Nasdaq Composite closing price variations.
  - data_for_use_basic: Bitcoin closing price variations

    - data_for_use_basic_w_gold: Bitcoin closing price variations and Gold closing price variations.
    - data_for_use_basic_w_nasdaq: Bitcoin closing price variations and Nasdaq Composite closing price variations.
    - data_for_use_basic_w_all: Bitcoin closing price variations, Gold closing price variations and Nasdaq Composite closing price variations.
    - data_for_use_bone_deep: only Bitcoin closing price daily variations for the last 30 days.
- Transformers (there are only numeric columns in the data):
    - StandardScaler
    - MinMaxScaler
    - PowerTransformer, method='yeo-johnson'
    - QuantileTransformer, output_distribution='normal'
    - MaxAbsScaler
- Classifiers:
    - KneighborsClassifier, with n_neighbors between 1 and 20
    - DecisionTreeClassifier, with max_depth between 1 and 20
    - AdaBoostClassifier, with n_estimators between 1 and 20
    - RandomForestClassifier, with min_samples_split between 2 and 20

The results of running all these models on the cross validation set are stored in the file *all_models_CV_results.csv*. The inspection of this file reveals that only 482 out of 3556 models beat the naive model on the cross validation set and that, when results are different for different transformers, models that have used the PowerTransformer overwhelmigly obtain the best results.

In order to decide between the 107 models that beat the naive model and use PowerTransformer, I perform back-testing on the training data, using all those models to predict Bitcoin evolution since 2018. The results of the models that manage to beat Bitcoin evolution during the back-testing are stored in the file *back_testing_results.csv*.

44 of these models beat the naive model during back-testing. The most repeated classifier is KneighborsClassifier (43 out of 44) and the most repeated dataset is data_for_use_basic (18 out of 44).

The following table presents a summary of the results of the models that beat the naive model both in backtesting and cross validation using data_for_use_basic, PowerTransformer and KneighborsClassifier:

dataset	model	model_boost_bt	f1_score	accuracy	model_boost_cv
data_for_use_basic	KNeighbors_2	0.273368	0.296296	0.491071	0.038458
data_for_use_basic	KNeighbors_3	0.276004	0.486322	0.497024	0.17485
data_for_use_basic	KNeighbors_4	0.497313	0.369231	0.511905	0.195834
data_for_use_basic	KNeighbors_5	0.205694	0.468085	0.479167	0.381945
data_for_use_basic	KNeighbors_7	0.423383	0.449231	0.467262	0.010775
data_for_use_basic	KNeighbors_8	0.375538	0.416961	0.508929	0.427289
data_for_use_basic	KNeighbors_9	0.704103	0.504451	0.502976	0.673785
data_for_use_basic	KNeighbors_10	0.99751	0.450331	0.505952	0.537804
data_for_use_basic	KNeighbors_11	0.596276	0.530259	0.514881	0.463674
data_for_use_basic	KNeighbors_12	1.334393	0.471338	0.505952	0.127442
data_for_use_basic	KNeighbors_13	0.731507	0.538682	0.520833	0.277382
data_for_use_basic	KNeighbors_14	0.801408	0.503067	0.517857	0.198657
data_for_use_basic	KNeighbors_15	0.702536	0.561111	0.529762	0.23614
data_for_use_basic	KNeighbors_16	0.542663	0.511905	0.511905	0.072797
data_for_use_basic	KNeighbors_18	0.992195	0.520231	0.505952	0.029453
data_for_use_basic	KNeighbors_19	0.533433	0.55643	0.497024	0.060268
data_for_use_basic	KNeighbors_20	0.487056	0.548476	0.514881	0.165602

There is no clear winner between these models, it depends on the metric used. I think the chosen model should do better with more training data, so I choose to go with the model with the highest performance increase in cross validation (column *model_boost_cv*), and that is KNeighbors_9.

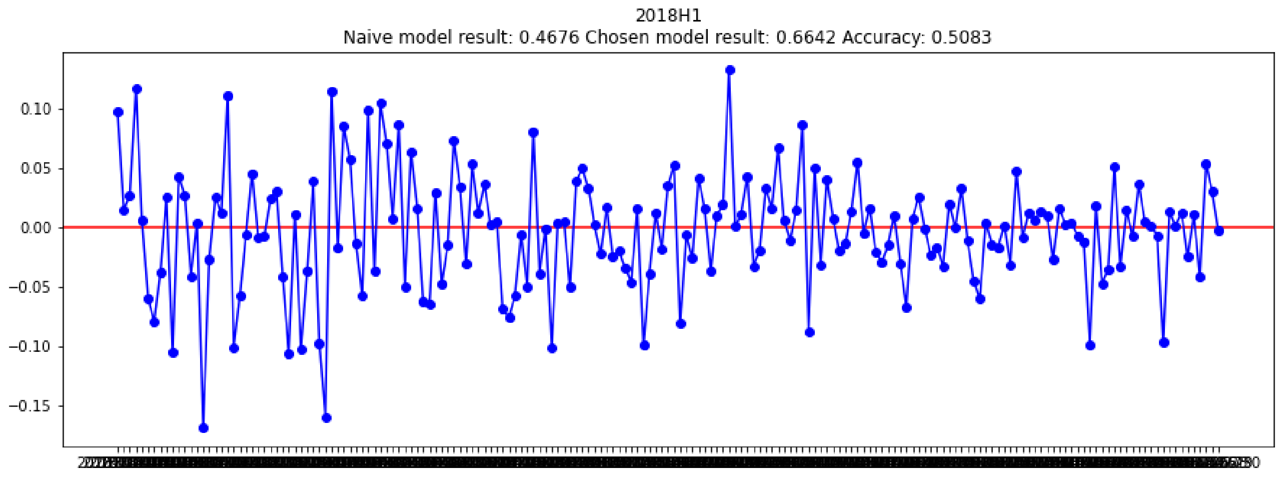
When I run this model on the test set the results are slightly negative, with a variation of 3,84 (284% return) using the naive model and a variation of 3,40 (240% return) using the chosen model.

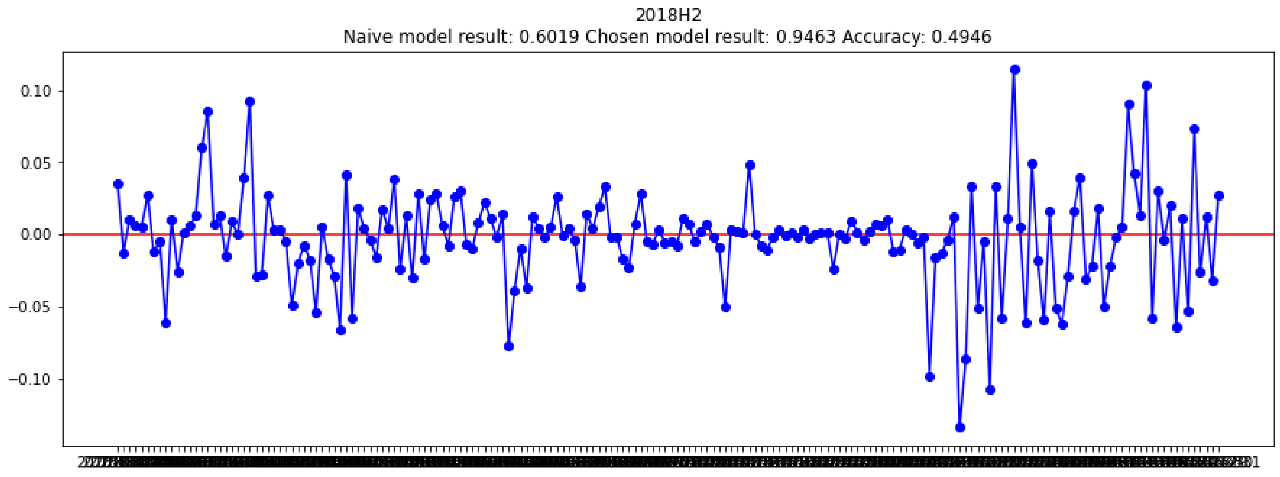
This doesn’t bode well for the model, but I believe the analysis of the results that is included in the following sections demonstrates that the negative result in the test set is due to a concrete period included in that set, when Bitcoin had a bull run (continuous and high increases in price) . Bull runs are situations in which the chosen model performs worse than the naive model.

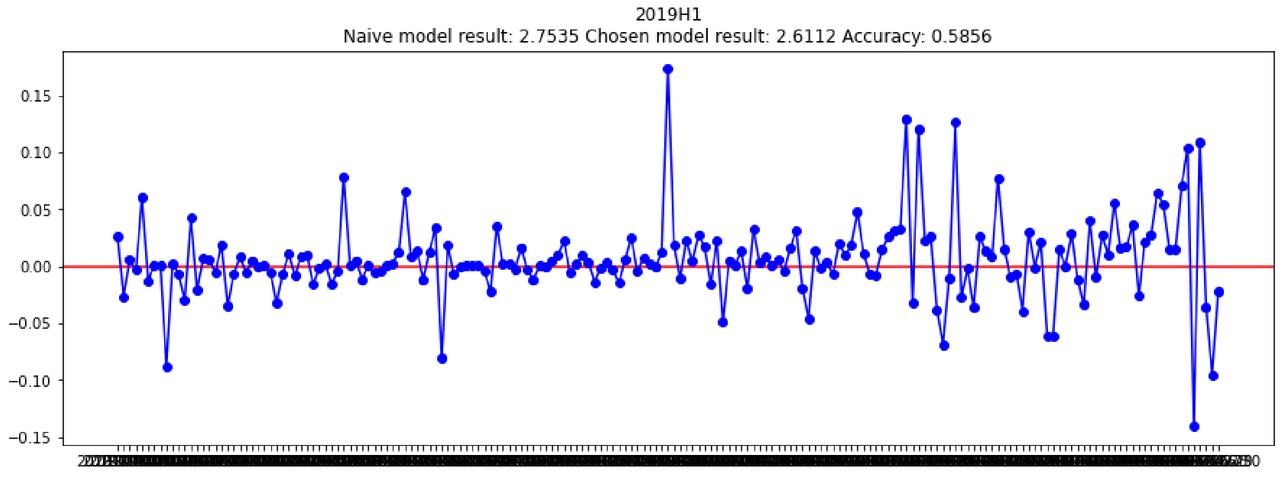
4. Summary of main results

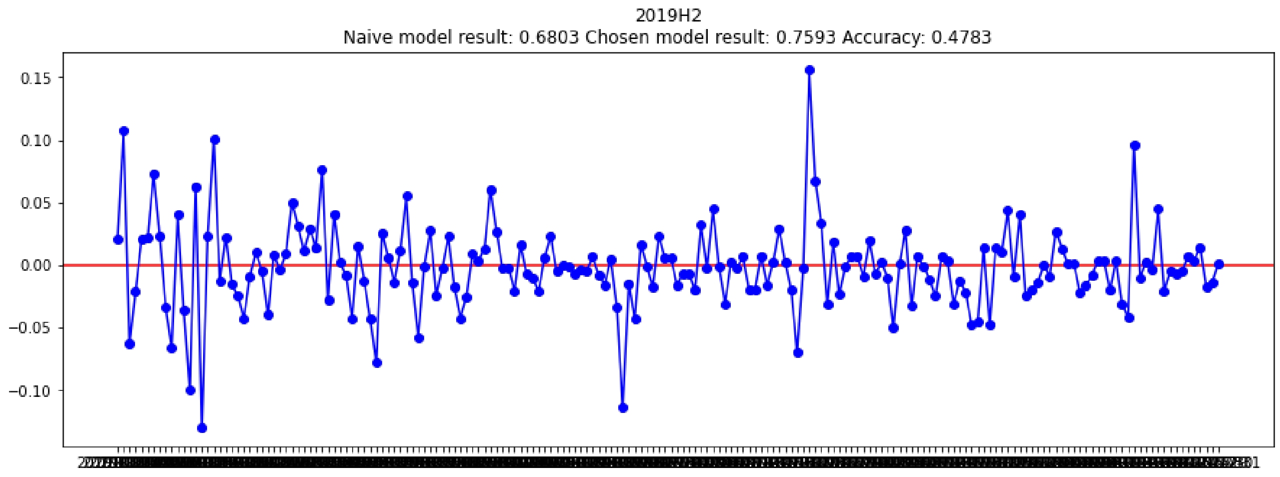
As stated above, the chosen model is Kneighbors_9 with transformer PowerTransformer, for the data data_for_use_basic (Bitcoin closing price variations).

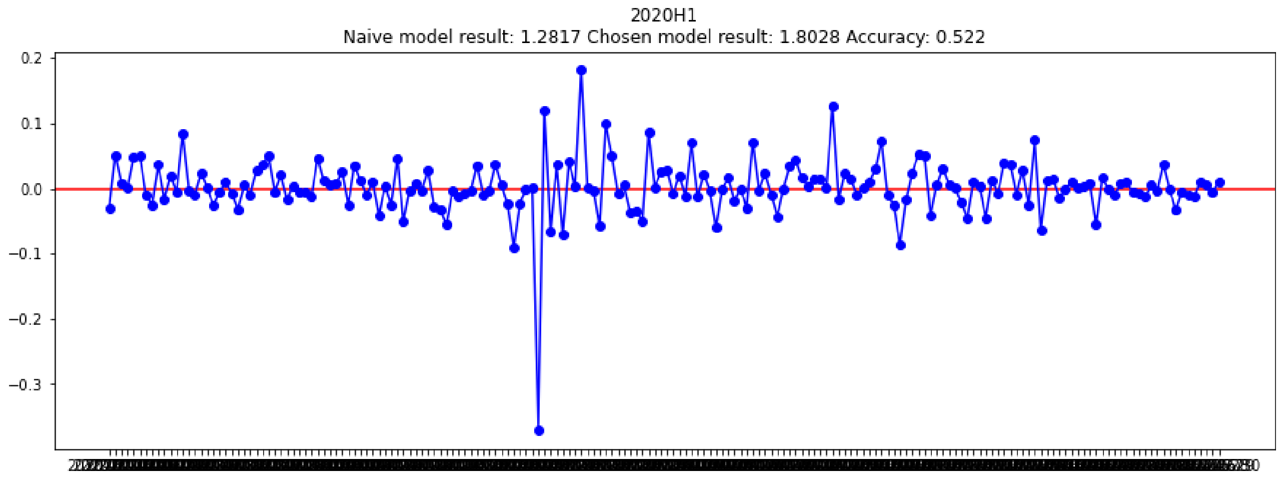
The results of running this model since 2018, separated by half-years, are shown below, with the plotting of the daily price variations during those periods.

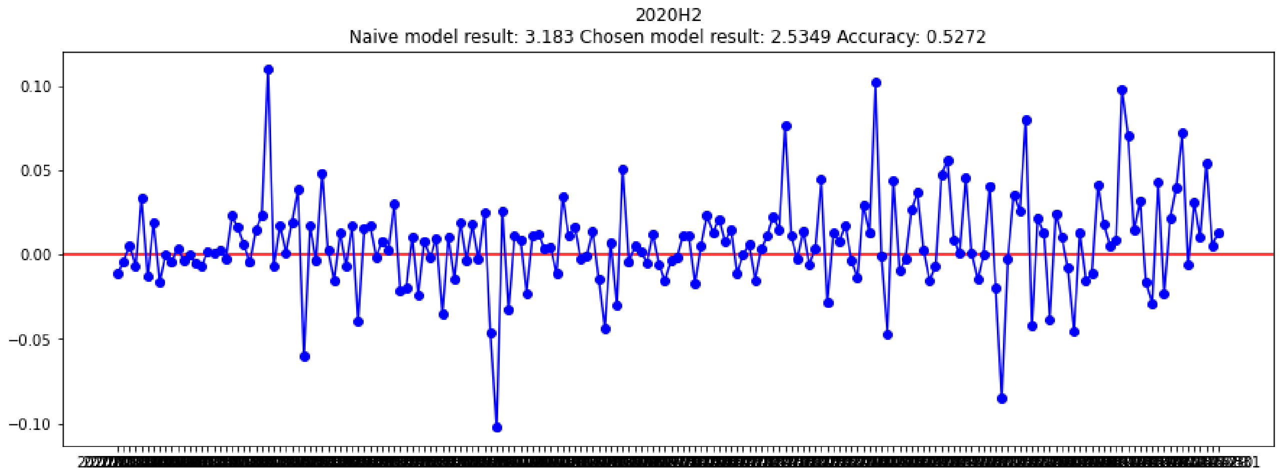

2018H1
Naive model result: 0.4676 Chosen model result: 0.6642 Accuracy: 0.5083
0.10
0.05
0.00
−0.05
−0.10
−0.15

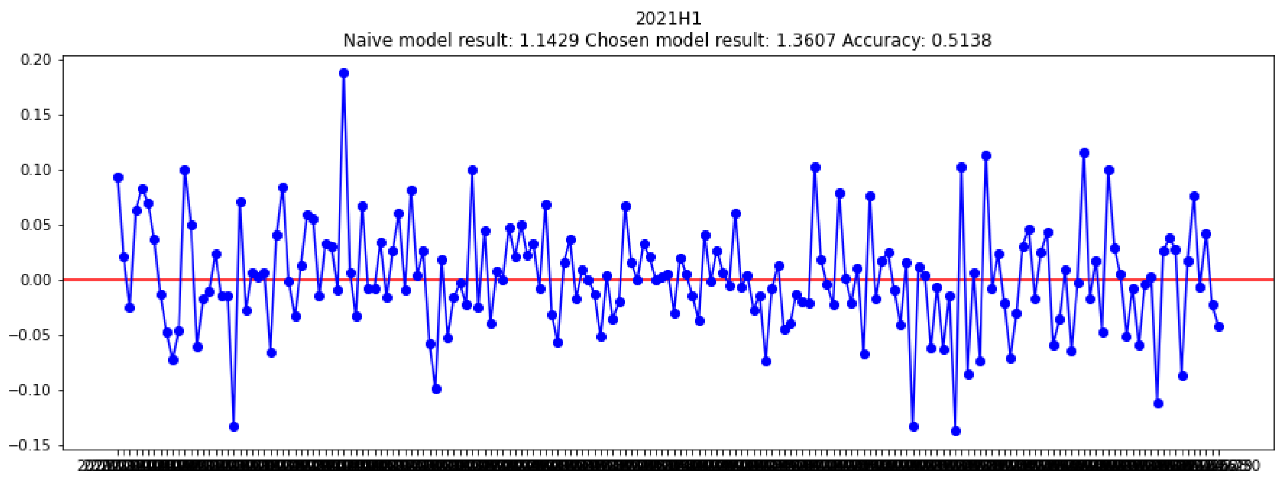

2018H2
Naive model result: 0.6019 Chosen model result: 0.9463 Accuracy: 0.4946
0.10
0.05
0.00
−0.05
−0.10

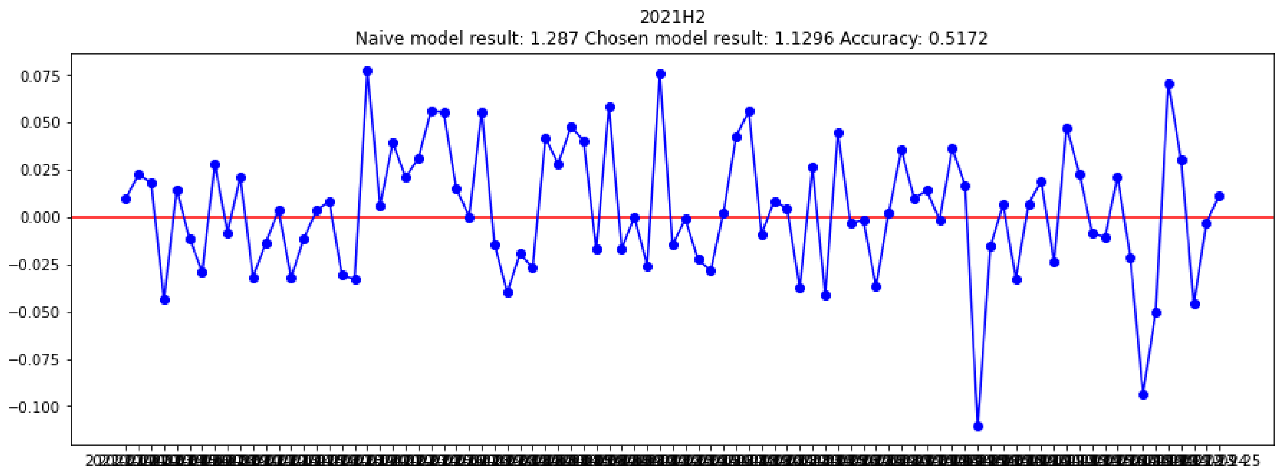

2019H1
Naive model result: 2.7535 Chosen model result: 2.6112 Accuracy: 0.5856
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15


2019H2
Naive model result: 0.6803 Chosen model result: 0.7593 Accuracy: 0.4783
0.15
0.10
0.05
0.00
−0.05
−0.10


2020H1
Naive model result: 1.2817 Chosen model result: 1.8028 Accuracy: 0.522
0.2
0.1
0.0
−0.1
−0.2
−0.3


2020H2
Naive model result: 3.183 Chosen model result: 2.5349 Accuracy: 0.5272
0.10
0.05
0.00
−0.05
−0.10





The following table presents the accumulated results for every half-year:

Year	Naive model	Chosen model
2018H1	-53%	-34%
2018H2	-40%	-5%
2019H1	+175%	+161%
2019H2	-32%	-24%
2020H1	+28%	+80%
2020H2	+218%	+153%
2021H1	+14%	+36%
2021H2	+29%	+13%

5. Conclusions

It seems that, up until now, Bitcoin evolution can be predicted well enough for beating the market using the assumptions indicated in section 3.I.

The model chosen in this project performs specially well during bear markets and loses to the market during bull runs (see the graphic above for 2019H1 and, specially, 2020H2), but over an extended period of time it seems to be able to beat the Bitcoin performance. This may change as a higher number of professional investors start trading bitcoin (see Bitcoin moves in lockstep with US stocks as big traders enter market).

An investor that had bought Bitcoin at the beginning of 2018 and held it until September 26th 2021 would have gotten a **216%** return. The same investor following the presented model's advice would have achieved a **775%** return.

6. User manual of the frontend

The frontend is divided into 3 sections:

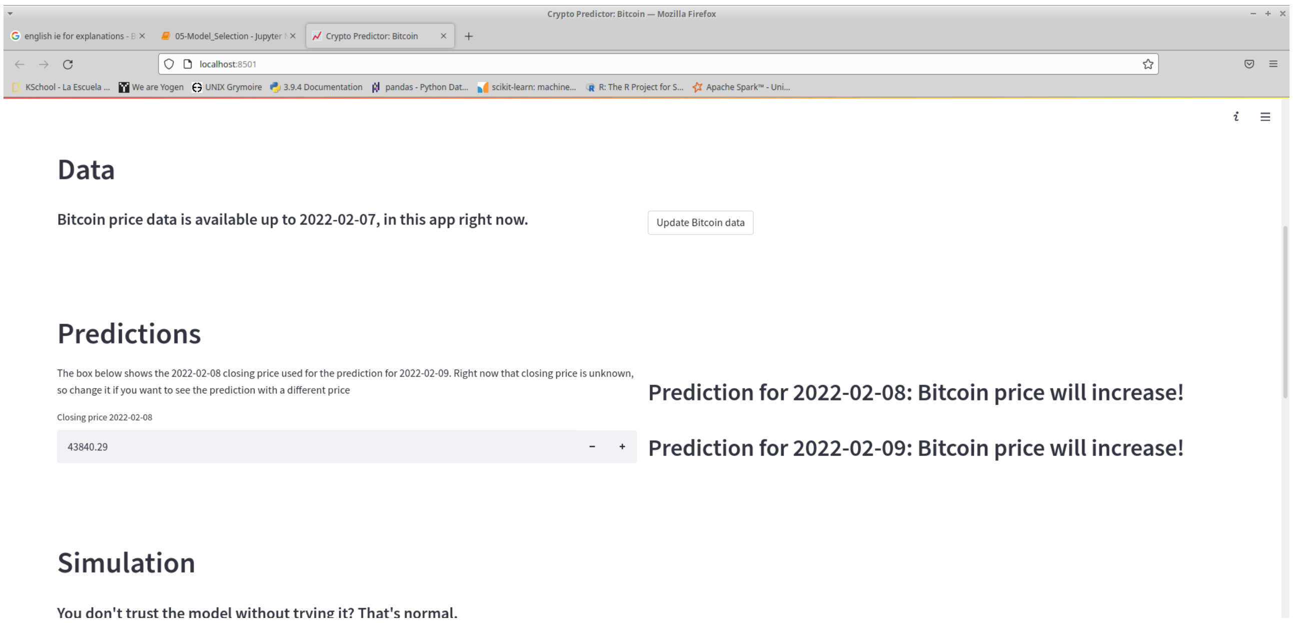
- Data
- Predictions
- Simulation

I. Data

In this section the user is informed of the last date of available data in the app right now.

The user can update this data by clicking in the "Update Bitcoin data" button. The update process uses Firefox web browser for web scraping, so it might not work in a machine without that browser.

Please note that the most up-to-date data that can be obtained is for yesterday, as the closing price for the present day is always unknown. Also remember that the app uses UTC time for closing prices and dates.


Data

Bitcoin price data is available up to 2022-02-07, in this app right now.

Predictions

The box below shows the 2022-02-08 closing price used for the prediction for 2022-02-09. Right now that closing price is unknown, so change it if you want to see the prediction with a different price

Closing price 2022-02-08

43840.29

Prediction for 2022-02-08: Bitcoin price will increase!

Prediction for 2022-02-09: Bitcoin price will increase!

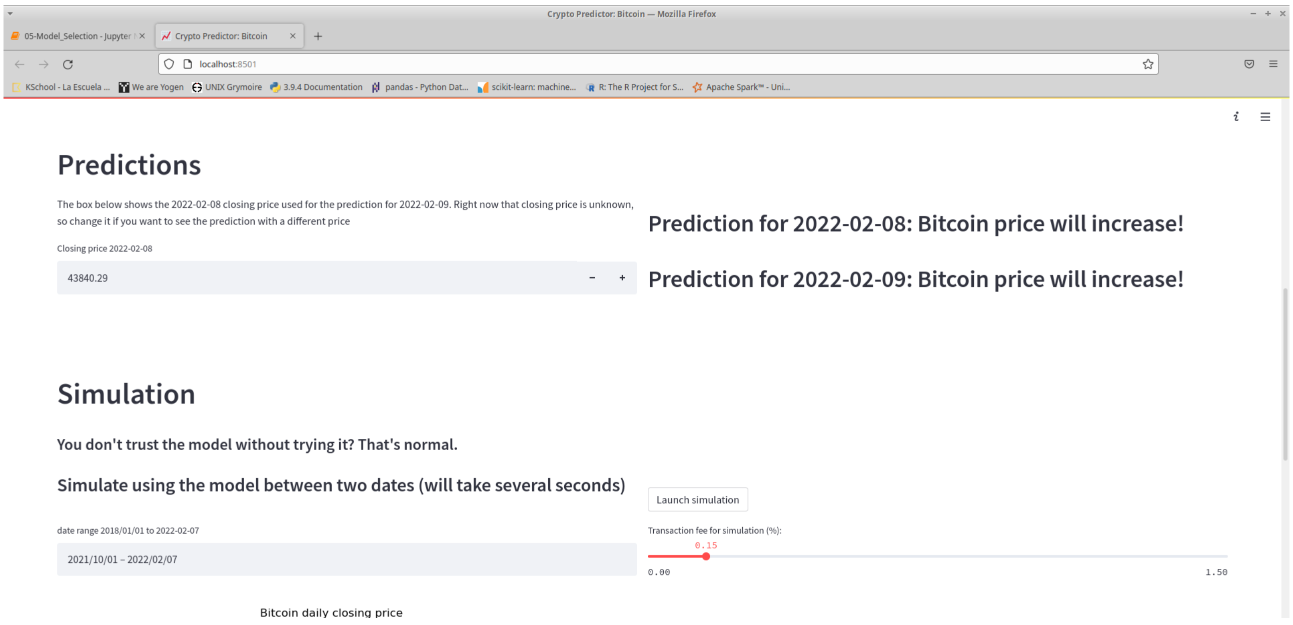
Simulation

You don't trust the model without trying it? That's normal.

II. Predictions

In this section the user can see the predictions of the model for the day after the data last available date and 2 days after.

The prediction for “2 days after” uses as input the closing price for the day after, which is unknown. This is for the use case when it’s 23:55 and you want to know if you should buy/sell Bitcoin. In that case you can input the current price in the box (as a proxy for the actual closing price, which will be known in 5 minutes) to get a prediction for the following day.


Predictions

The box below shows the 2022-02-08 closing price used for the prediction for 2022-02-09. Right now that closing price is unknown, so change it if you want to see the prediction with a different price

Closing price 2022-02-08

43840.29

Prediction for 2022-02-08: Bitcoin price will increase!

Prediction for 2022-02-09: Bitcoin price will increase!

Simulation

You don't trust the model without trying it? That's normal.

Simulate using the model between two dates (will take several seconds)

date range 2018/01/01 to 2022-02-07

2021/10/01 – 2022/02/07

Transaction fee for simulation (%): 0.15

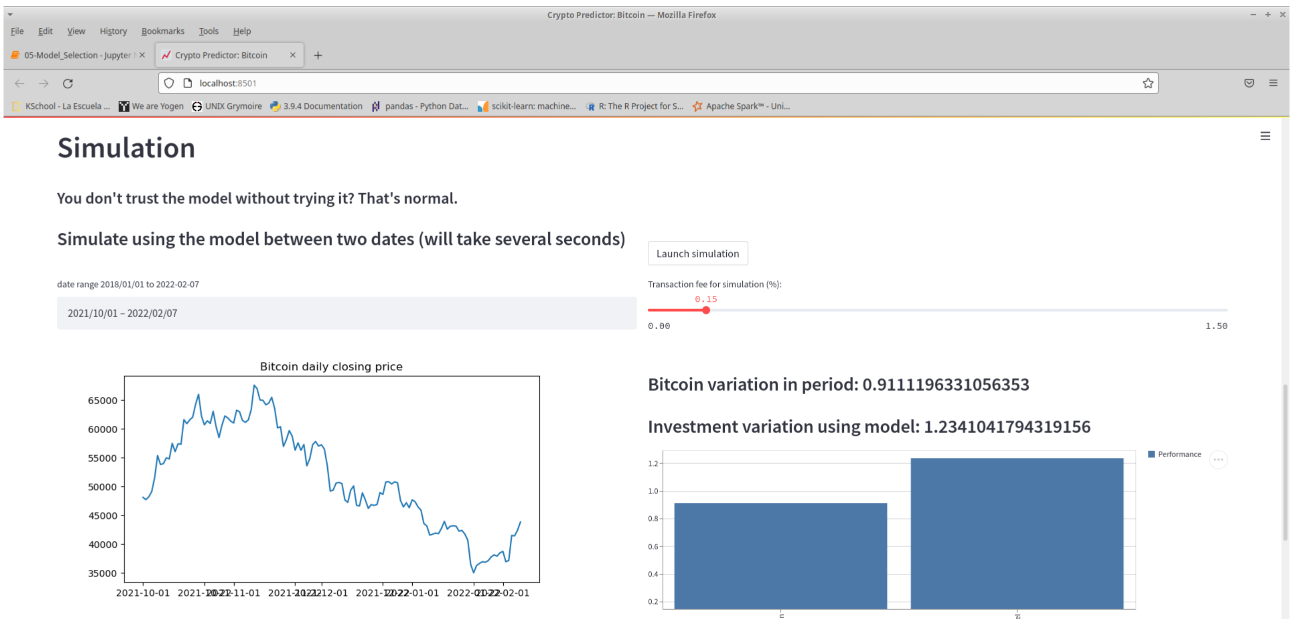
III. Simulation

In this section the user can simulate the model's performance during past periods.

Simulations take time, as they train the model and make a prediction for every day included in the period.

In order to launch a simulation, the user must:

- Select a period (2 dates). When the selection is completed, the price chart for Bitcoin during that period will be shown below.
- Select a transaction fee. This is the percentage paid to the exchange in every buy or sell operation.
- Press the "Launch simulation" button. Bitcoin variation (end price/starting price) and investment variation using the model will be shown for the input period, applying the input fee.



Simulation

You don't trust the model without trying it? That's normal.

Simulate using the model between two dates (will take several seconds)

date range 2018/01/01 to 2022-02-07

2021/10/01 – 2022/02/07

Transaction fee for simulation (%): 0.15 (0.00 – 1.50)

Bitcoin variation in period: 0.9111196331056353

Investment variation using model: 1.2341041794319156

References

- John J. Murphy (1986) Análisis técnico de los mercados financieros. Gestión 2000
- Francesca Lazzeri (2020) Machine Learning for Time Series Forecasting with Python. Wiley
- Jason Brownlee (2020) How to Create an ARIMA Model for Time Series Forecasting in Python https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
- Jason Brownlee (2016) How To Backtest Machine Learning Models for Time Series Forecasting https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/