Machine Learning Engineer Nanodegree

Capstone Project

Predicting Bitcoin Prices
Lukasz Tymoszczuk
November 8th, 2018

Definition

Project Overview

Bitcoin is decentralised crypto-currency which nobody physically owns. It exists only virtually and with security by design requires no intermediary institution to secure the transactions. Due to its nature, it is free from any bank and government actions. The most recent popularity raises a question when is a good time to buy Bitcoins [1].

High popularity causes that the currency is very volatile and unpredictable. The hype created by bitcoin is visible in social media. There are some examples of how sentiment analysis can be used to predict Bitcoin prices [2].

In this project, I simplify the Bitcoin Price model by assuming that information about future prices can be deducted from the past prices. The historical data is taken from kaggle.com [3]. The model will use only the past performance in order to predict future prices.

Problem Statement

The chosen data (bitcoin prices) is so-called time series data. There are only two features: `Close Price` and 'Volume' values which are considered. The goal is to create a model which used on unseen data predicts prices as close as

possible to actual prices. Such a model even if not perfect, can be used to indicate moments when to sell and when to buy Bitcoins.

The steps for building and testing the model:

1. Prepare the normalised data
2. Train the model by using LSTM (Long-Short Term Memory) type of RNN (Recurrent Neural Network).
3. Prove that the historical prices are not random so that they may be predicted. This step will compare test month of unseen prices against completely random prices to see the model predicts prices much better on the real data.
4. Test the LSTM-model against different models. Calculate the root squared mean error (RMSE) between predicted and actual prices. The models to compare:
    - Random walk
    - Linear regression (two variants)
    - Moving Average (two variants)
5. It is expected that the LSTM model will have the smallest RMSE among the above models.
6. The winner model can be then used on Production - in real-life trading, for example, to determine if a long or short position should be opened in the next day.

Metrics

Root square mean error (RMSE) for all days between predicted and actual prices shows us how close the model predicts prices:

```
RMSE = for all days (i): sqrt(sum((predicted_price_day_i - actual_price_day_i)^2))
```

The experiment tests whether RMSE for LSTM model is the smallest:

```
RMSE_LSTM = min(RMSE_LSTM, RMSE_RANDOM_WALK,
RMSE_LINEAR_REGRESSION, RMSE_MOVING_WINDOW)
```

Analysis

Data Exploration

`coinbaseUSD_1-min_data_2014-12-01_to_2018-06-27.csv` contains 1819074 time-series data. Every minute a transaction with following features is recorded:

- Open - The very first price in dollars for this period
- High - The highest price in dollars for this period
- Low - The lowest price in dollars for this period
- Close - The last recorded price in dollars in this period
- Volume (BTC & Currency) - How many bitcoins have been bought/sold in this period

Fig. 1 The sample input of the data.

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1417411980	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000
1	1417412040	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000
2	1417412100	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000
3	1417412160	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000
4	1417412220	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000
5	1417412280	300.00	300.00	300.00	300.00	0.010000	3.000000e+00	300.000000

There are no missing values for raw data (for any columns), but there are some missing days in the time series data. The number of missing days is not significant so it can be easily ignored.

Exploratory Visualization

Fig. 2 The "close" price data presents Bitcoin price between 01/12/2014 and

27/06/2018. It contains the 17th December 2017 peak of $19891.99.

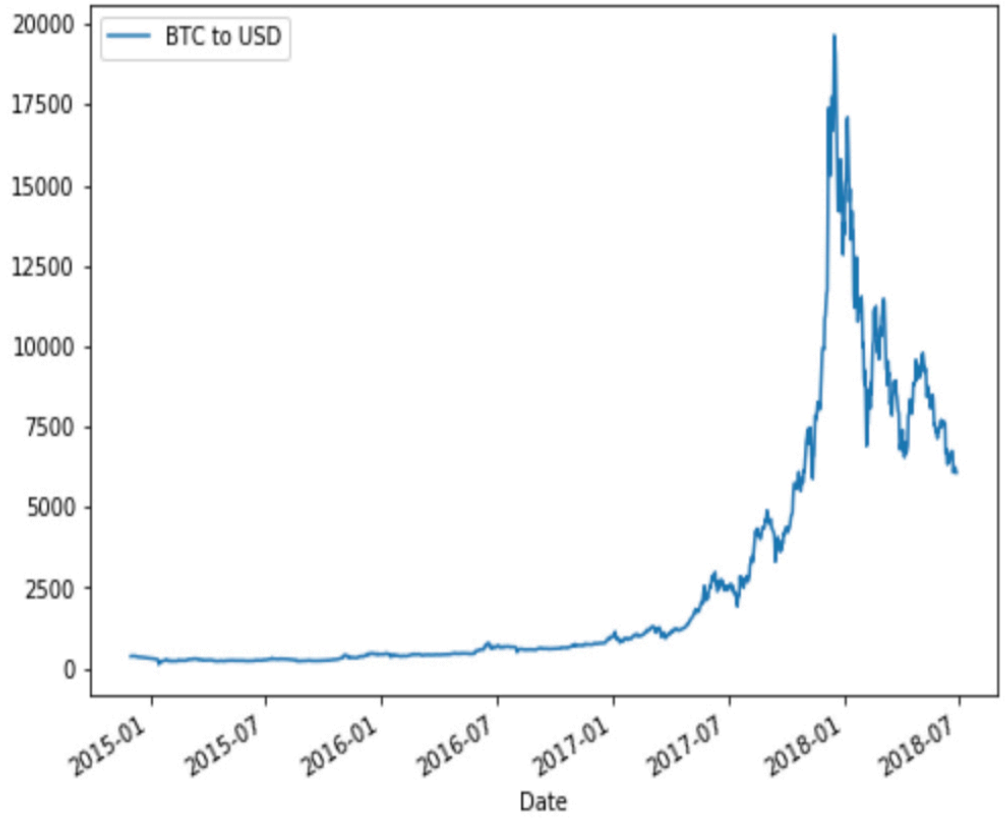
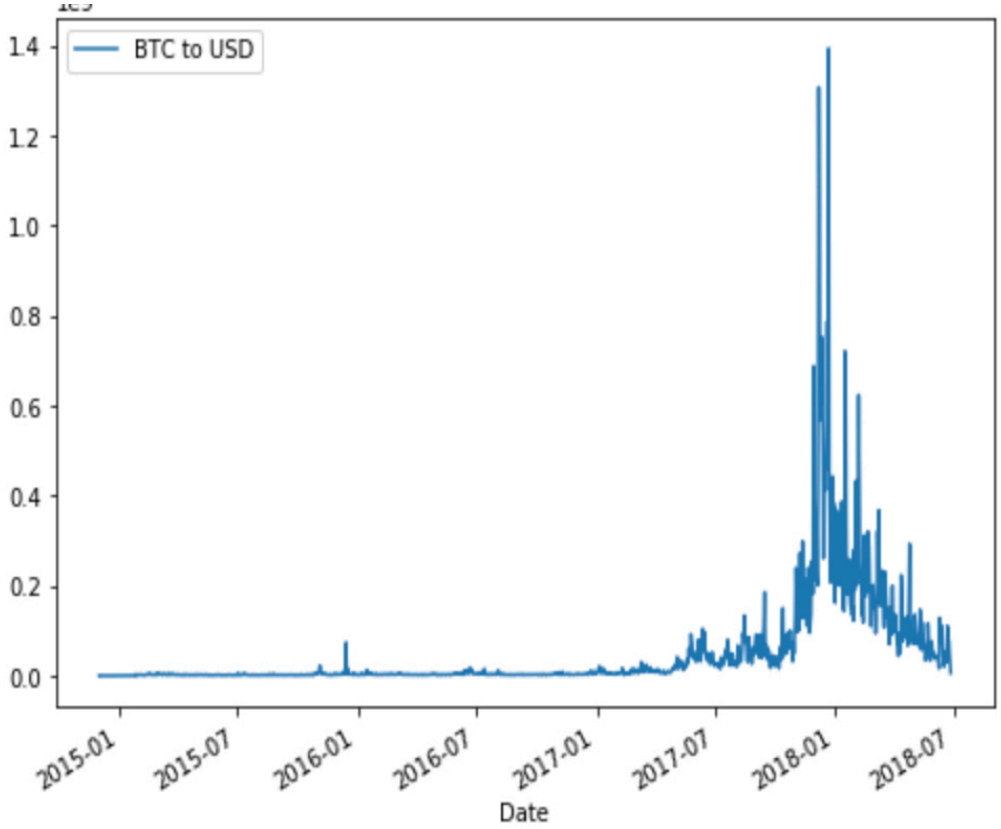


Fig. 3 The volume of Bitcoin transactions.

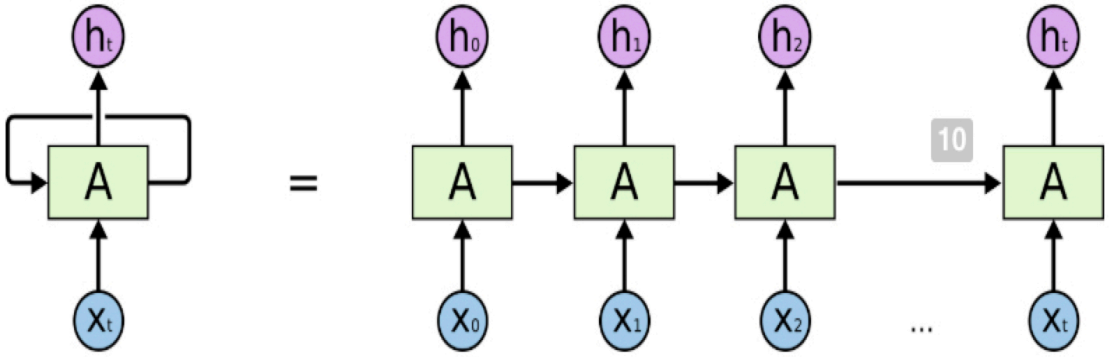


Algorithms and Techniques

Deep learning algorithms typically required normalised data, it's not different from LSTM. The very first step before we train model is to make all the prices and volumes between 0-1.

Recurrent Neural Network (RNN) introduces a possibility to learn from input and feedback from past experiences. The RNN is extremely powerful toolkit as it gives to neural network kind of memory. The concept of past experiences means that this Neural Network is perfect to model time series data like prices.

Fig. 4 The Recurrent Neural Network (picture from [4]).



Long Short-Term Memory networks are an extended version of RNN. They are capable of learning long-term dependencies.

LSTM cell has five components:

- Cell state - it is internal memory. It stores the long term and short term states.
- Hidden state - It is an output state which is used for returning the final output (predicting time-series prices)
- Input gate - chooses how much information is passed to cell state
- Forget gate - chooses how much information from the previous cell and current input is passed to the current cell state
- Output gate - Chooses how much information is passed to the hidden state

The following parameters can be tuned to optimise the algorithm:

Training:

- window-length - how many days the algorithm will have access to
- units - dimensionality of the output
- batch-size - the batch size used for the training sample. Each batch trains network in successive order, taking into account the updated weights coming from the appliance of the previous batch

Neural network architecture:

- Number of layers
- LSTM activation function, default: tanh
- Dropout size
- Optimiser and loss function
- Validation set size
- Number of iterations (epochs)

Benchmark

There are two group of benchmarks.

The first benchmark answers a question: Do past prices contain a value which can be used to predict future. It is a fundamental question. It checks if we can predict future prices knowing only the past performance, and at the same time ignoring all other features (news, gas prices, political situation etc.). In this tests, I used the trained LSTM model for predicting randomly generated prices. It is expected to have rather poor results here, as the `price_t+1` does not relate to `price_t`, there is no pattern the algorithm could learn from train data.

Prediction type	Error
255 days of real prices	RMSE = $633.30
255 days of made up (random) prices	RMSE = $1096.51

The second group of tests is a comparison of the LSTM to other models. Some of them like Linear Regression are only theoretical, as they are not designed for predicting time-series data. The model which has the smallest error is the best choice and potentially a good candidate for using this in real life scenarios.

The following algorithms were tested and compared with LSTM model.

Algorithm	Setup	RMSE

LSTM	**one layer**	**RMSE = $633.30**
Random Walk	+/- 20% per day	RMSE = $3184.26
Linear Regression	Continuous split	RMSE = $9185.11
Linear Regression	Random split	RMSE = $703.52
Moving averages	Long window = 100 days	RMSE = $3758.70
Moving averages	Short window = 20 days	RMSE = $1665.71

Methodology

Data Preprocessing

The input data has good quality, as they do not contain any incorrect values. Besides a few days missing - which can be ignored as it is statistically insignificant, the rest of the data are good and can be used in the algorithm.

In the beginning, the data is reduced to only three columns: Date, Close and Volume.

The second step is to normalise the input data to have values between 0.0 and 1.0 (Normalization).

The third step is to split the data into Train and Test. The train set is then split into Train and validation sets in the `fit` method. The train set contains prices between: `01/12/2014 – 15/10/2017`. The test set contains prices between: `16/10/2017 – 27/06/2018` (80% train, 20% test split).

Implementation

The implementation has four parts:

1. The model training stage
2. The model testing stage

3. Benchmark models creation
4. Validating the results against the benchmark models

The model testing stage uses previously split historical values and calculate the RMSE to determine how efficient is the prediction. There is also a visual representation of actual and predicted price.

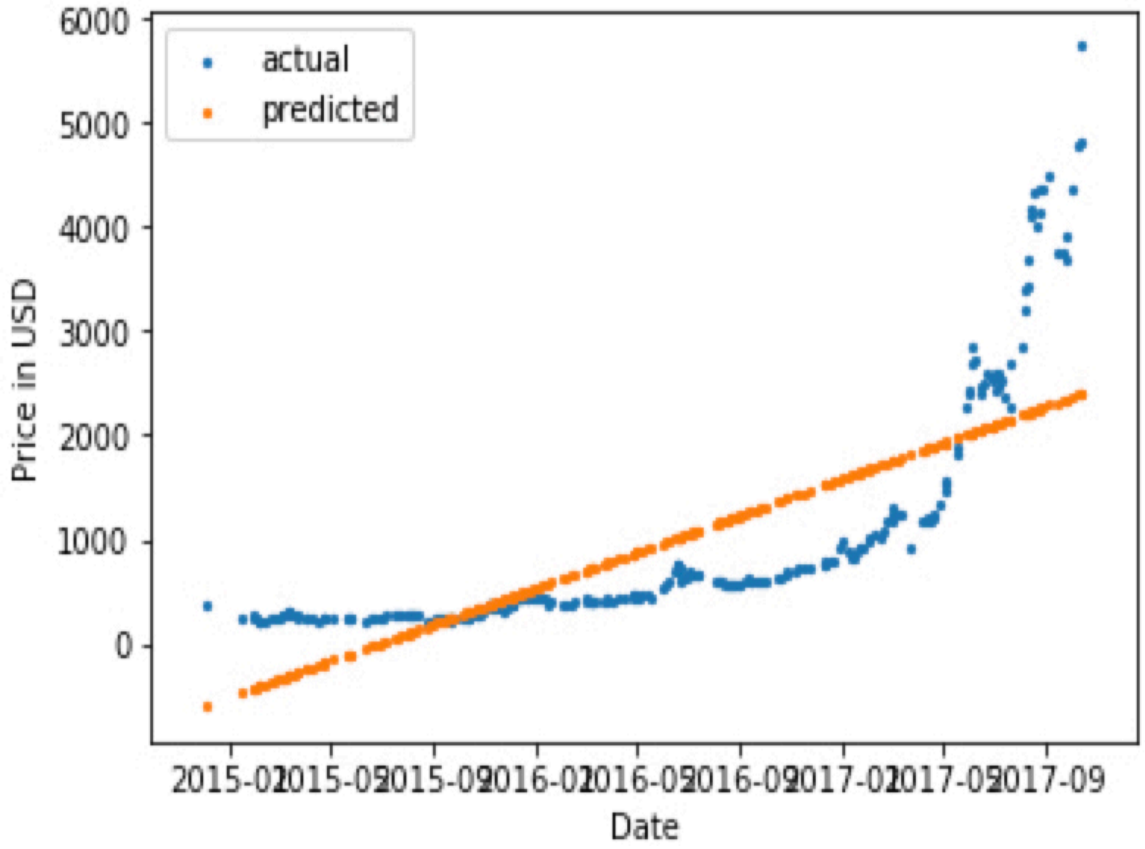
The additional benchmarks are calculated: Random prices to compare that the LSTM model won't predict these random data effectively.

There are also Random walk model, Linear regression and Moving average models computed. All of them present higher mistakes (in terms of RMSE) in comparison with LSTM model.

Refinement

The one-layer LSTM model presents a good result in terms of RMSE: Only $633 error in predicting 255 days is a promising result. The random walk benchmark and moving averages show that this result is good, as using any other approaches will not give the smaller error.

The Linear Regression with random split generates small error too, recorded at $703. Even though it has close RMSE to the LSTM model, but when we see the visual representation it will be clear that the Linear Regression is not pragmatic, as it minimizes the error but cannot be used to determine if Bitcoin should be bought or sold - the simple straight line does not help in trading strategy.



Results

Model Evaluation and Validation

The combination of layers and hyperparameters were chosen by trial and error. They generated the lowest error (RMSE)

The model training stage uses Sequential neural network model with the following architecture:

Layer (type)	Output Shape	Param #
lstm_89 (LSTM)	(None, 256)	265216
dropout_80 (Dropout)	(None, 256)	0
dense_57 (Dense)	(None, 1)	257

activation_56 (Activation)	(None, 1)	0

Total params: 265,473, trainable params: 265,473

Justification

Algorithm	Setup	Error in prediction *
LSTM	**one layer**	**Base model**
Random Walk	+/- 20% per day	+403%
Linear Regression	Continous split	+1350%
Linear Regression	Random split	+11%
Moving averages	Long window = 100 days	+493%
Moving averages	Short window = 20 days	+163%

* - Error in prediction in comparison to the base model (LSTM) is calculated in the following way:

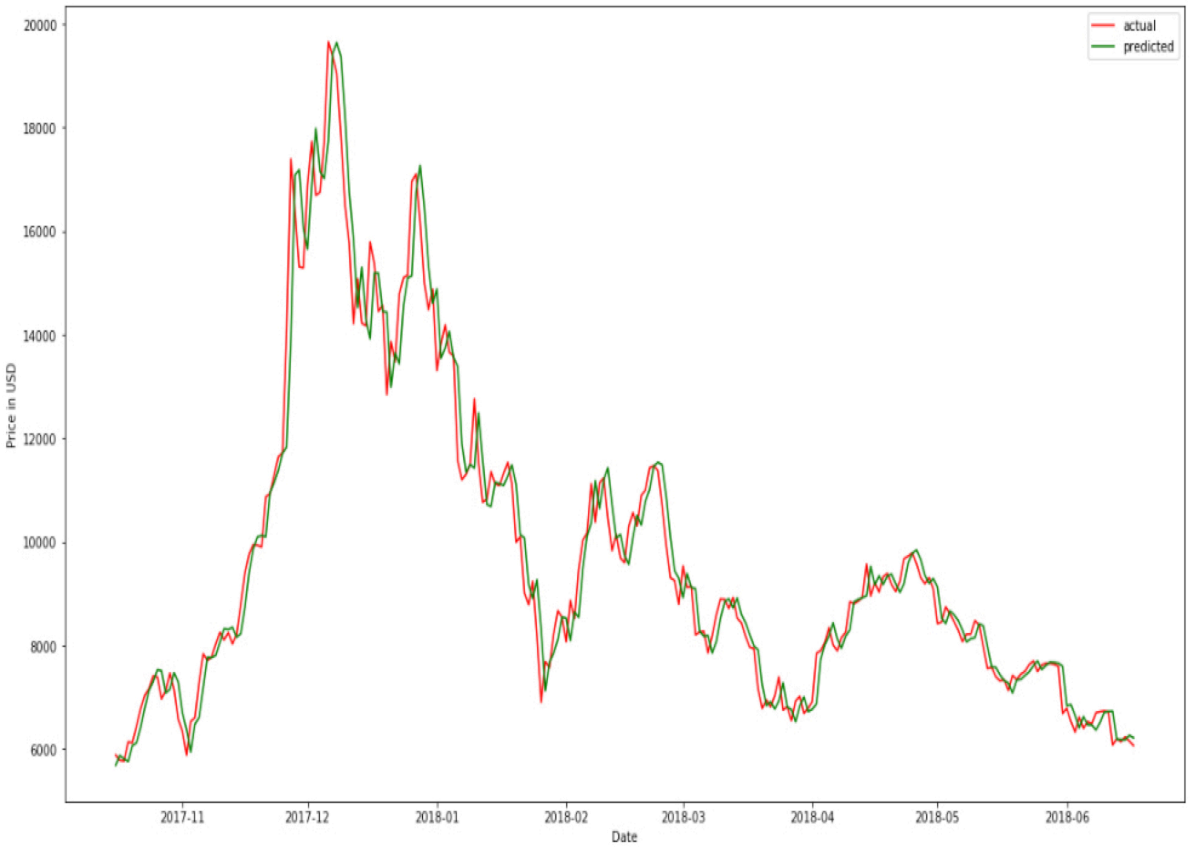
```
abs(base-model) / base * 100%
```

Trading randomly (flipping the coin) gives 403% worse results than using LSTM model. Using moving averages method generated 163% poorer results. Finally, the linear regression which cannot be used for any trading strategy gives 11% poorer portfolio than LSTM model.

Conclusion

Free-Form Visualization

As the model on test set showed, using the LSTM model for 255 days gives an error of only $633.30 which is relatively small values compared to the prices which are in the highest peak about $19,000.



The chart shows that the algorithm adjusts values whenever price changes the trend from short to long and from long to short position. In other words, the algorithm follows the real prices, and change the direction effectively.

Reflection

The process for predicting bitcoin prices can be summarised using the following steps:

1. Download the data, normalise them
2. Train the model until the error RMSE is relatively small (use other benchmarks to help to determine what does it mean small)
3. Use the model on real-life data, update the original data set every day.
4. If the model predicts the price going up, it's an excellent opportunity to buy. If it predicts the trend going down, it is probably an excellent opportunity to sell.

The most challenging part is that even if the algorithm on the chart looks good, the additional fees for buying/selling can impact our final performance. Additionally, some of the price movement is determined by factors like Government actions, social media news. The model cannot predict such events. It is doubtful that such a simple price model which ignores so many factors, and only learn from the historical prices can be a method used by professional Hedge fund companies or professional traders.

Improvemen

During this training, the data has been simplified to use only Closing day prices. The input data is a continuous time series data maybe LSTM could predict better price with larger data once the time series are not collapsed to per-day prices.

Additional layers of LSTM could improve the results, but it would impact the final run time.

Additional features like open prices, average prices could be taken into considerations and give more features to use in the algorithm.

Another approach for improving the results would be a combination of LSTM and sentiment analysis - to interpret social media noise in order to predict Bitcoin prices.

Resources

1. https://www.quora.com/What-is-bitcoin-Why-is-it-so-popular
2. https://hackernoon.com/sentiment-analysis-in-cryptocurrency-9abb40005d15
3. https://www.kaggle.com/mczielinski/bitcoin-historical-data
4. http://colah.github.io/posts/2015-08-Understanding-LSTMs/