Bitcoin Price Prediction Using LSTM NNs

1. Introduction

In my project I will try to predict the future price of Bitcoin with the use of Long Short Term Memory RNNs. Based on the results I will try to find out if it is really that hard to build a model (machine learning model, trading bot) that would help us make profits in crypto market.

2. Bitcoin (BTC) and Bitcoin Blockchain

Bitcoin (BTC) is a *decentralized digital asset*. It's transactions are recorded on an *electronic ledger* called Bitcoin Blockchain. This ledger is stored on many computers (nodes) around the world that are connected and communicate with each other, forming a *peer-to-peer network*. It is a first example of digital asset of value that we could transfer over the internet without any third party acting as an intermediary.

Bitcoin was introduced on 31st of October 2008 in Bitcoin's whitepaper written by anonymous Satoshi Nakamoto.

3. How Bitcoin Works

Users around the world make orders for transactions which are then put in the transaction pool (or *memory pool*). Specific nodes – called *miners* take transactions that have highest *transaction fees* (price – amount of BTC that is required by the network to make the transaction) out of the memory pool and build a packet – *block* out of them. Miners then have to solve a consensus mechanism called *Proof–of–Work* (PoW). They are repeatedly changing a specific number included in the block called *the nonce* and calculating the hash function of this number and all the transactions stored in the block (plus some other values). To solve the PoW correctly calculated hash has to be smaller than a *target number*. Target number is changing dinamically every 2016 blocks based on the *difficulty* (dependent on the number of nodes in the network). If the miner solves the PoW first, he broadcasts the block to other nodes. Others validate the block by specific rules and if the block is validated (formed correctly, PoW is solved correctly...) they upgrade their instance of Bitcoin Blockchain and start building the next block. Transactions in validated block are executed and the miner receives a *block reward* in the form of BTC. Each block contains a hash of the previous block and in that way the blocks are connected and are forming a *blockchain*.

4. LSTM

Long Short Term Memory (LSTM) is a type of recurrent neural network (RNN) capable of learning long-term dependencies (able to remember information for long periods of time).

RNNs are incapable of doing that because they only consist of reccuring modules of tanh layers which cannot retain information for a long time.

LSTM has a reccuring module with four layers interacting with each other. The LSTM *memory cell* has a *cell state* which enables the information to flow through the units without being altered. It also has three gates: forget gate, input gate and output gate. The *forget gate* uses sigmoid function to decide which previous information should be forgotten. *Input gate* uses multiplication of sigmoid and tanh to control the flow of information to the current cell state. And the *output gate* uses sigmoid function to decide which information is worth passing to the next hidden state.

5. Predicting the Price of Bitcoin (BTC)

While prediction of stocks is divided into two parts: *technical analysis* (historical price, volume, technical indicators,…) and *fundamental analysis* (revenue, net income, cash on hand, debt,…), prediction of cryptocurrencies is much different.

Technical analysis is pretty much the same. Traders are using historical price patterns and technical indicators to predict the future price. Compared to stocks, cryptocurrencies, including Bitcoin, do not earn any money or posses any cash on hand.

But there are parameters that could be taken into account when evaluating the value of specific cryptocurrency:

- Number of transactions in block
- Speed of transactions
- Size of the block
- Frequency of blocks
- Number of nodes (miners) in the network
- Rate of inflation (issue of new blocks – based on block award)
- Size of the community
- Security (used consensus mechanisms, hashing algorithms,…)
- Founder, creator
- Year of invention

But the problem is, that this parameters are only looked at, when investing for the mid or long term. Meaning, people will put money into the digital asset – Bitcoin only if they will hold it (not sell it) for longer period of time.

In my case, I used LSTMs and because of that it would be impossible to accurately predict the price in longer term. With every inaccurate prediction taken into account, the next prediction would be even worse and so on. Also, there are a lot of outside factors that can move the price drastically really fast (political news, Elon Musk tweets,…) and LSTM cannot take these factors into consideration.

Because of the reasons described above, I have decided to use only technical analysis. The *database* included *date*, *daily holc* (high, open, low, close) *prices* and *daily volume* for the last 5 years (1.1.2017 - 24.12.2021). We were looking at daily values, so using all of the holc prices would be pointless. Also, because of complexity reasons, I have decided not to use the daily volume parameter.

6. Tools

For my project I have used python programming language and it's libraries:

- numpy: for arrays, mathematical functions
- pandas: to import the dataset
- matplotlib.pyplot: for visualization
- sklearn.preprocessing: for data scaling
- keras: to implement RNN

*(The whole code of the project is added to e-classroom aswell as the description of it.)

7. Results

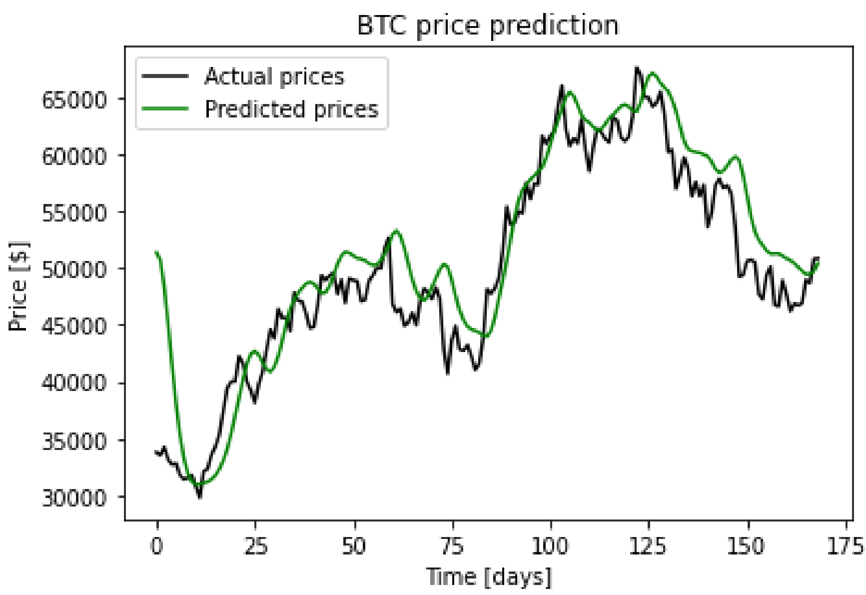
My LSTM model, is working in a way that it looks at a price of the last 60 days and then predicts the price of 61st day. And it does so until it comes to the end of the dataset.

I have built a model with *4 LSTM layers*. Each of them is followed by the *dropout of 20%* to try to prevent overfitting. At the end it goes through the *Dense() function* so it returns just one value – predicted price (we only need 1 neuron at the end to represent the output).

I compiled the model using *adam optimizer* and *mean squared error* for the *loss* function because it was the most recommended one for my type of prediction model and used in almost all of the similar tasks.

For training the model I experimented with the variety of different batch sizes and number of epochs. At the end I found the optimal *batch size* of *32* and *30 epochs*. Bigger number of epochs was slightly better but the training time was too long. So 30 was the best ratio between precision and training time. Much bigger number (around 100) did not work because the model was definitely overfitting.

I tested the model on the last 169 days and made the graph below:



*In the graph, day 0 is 9.7.2021 and day 168 is 24.12.2021.

At first, the graph looks very nice. It fits almost too nice. But we have to remember how the model works. It looks at the past 60 days, and then predicts the 61st day. Also, there is an error in the first 10 days, but the reason for me is unknown.

Based on the graph, I got the feeling that the model is overfitting which could be the case, because this is one of the drawbacks of LSTMs. It is also very difficult to apply the dropout regularization method to prevent this problem. I have tried with many different values, but the results stayed similar.

Because the upper graph does not help us much if we want to use the model to actually trade BTC with it (it only shows past prices – too late to make a trade, because this has already happened), I tried to predict the price for the next day (which did not happen yet). That way, we could see, based on the predicted price, in which direction the graph is moving and then open the trade. My last value in the dataset was from 24.12.2021 and I tried to predict the price on 25.12.2021. Then I added to the dataset the actual price on 25.12.2021 and compared it with our prediction and predicted the price for 26.12.2021. I did this for couple of days and got the following results:

Date	Predicted Price [$]	Actual Price [$]	Difference [%]
24.12.2021	49993,895	50822,2	1,629809414
25.12.2021	50366,055	50429,86	0,126522263
26.12.2021	50804,61	50809,52	0,009663543
27.12.2021	49963,277	50640,42	1,337159131
28.12.2021	49038,625	47588,86	-3,046437759
29.12.2021	50884,14	46444,71	-9,558526687

As we can see, the difference between predicted and actual price is pretty small (except on 29.12.2021), but that does not help us much if we want to use the model for trading.

The way we have to use the model for trading is that we look at the actual and predicted price for a specific day (for example on 25.12.2021) and also the predicted price for the next day (in our example 26.12.2021). We can see that the predicted price on 26.12.2021 is higher than on 25.12.2021, that means that the model is predicting the price will go up. So we can make a buy order (in which case we make a profit if the price goes up), and as we can see, if we look at the actual price on 26.12.2021, we would make a profit, because it actually went up.

In our 5 cases it made the right decision 3 times and wrong decision 2 times. That would be a great result, accuracy of 60% which is amazing for a trading bot. But of course number of cases is way too small (we would have to try it on at least 100 examples if not way more). At the end, accuracy would come very near 50%, meaning it would predict the movement of the price upwards or downwards in a totally random way – it would be a coin flip.

8. Comment

In my project I found out that using LSTM is not the best way to predict the future price. It is true that they are really good in a lot of areas, they are not able to predict the price of Bitcoin. Based on the graph, predicted prices are just slightly moved to the right from actual prices. That probably means that LSTM just uses a value close to the previous day as a prediction for the next day.

9. References

1. "Understanding LSTM Networks" Olah, Christopher. Colah's Blog, 27 Aug 2015.
2. "Time Series - LSTM Model" tutorialspoint. Time Series Tutorial
3. "Using LSTM Networks To Predict Crypto Prices" Boyed, Ankur. Ankur Boyed, 15 Nov 2019.