NAML Report - BTC price prediction

Camilloni Andrea, Cercola Matteo, Giacometti Giovanni

July 10, 2022

Abstract

Since the launch of Bitcoin in 2009, cryptocurrencies have gained a lot of popularity, due to a rapid growth of their market capitalization and the vast range of opportunities they offer. Cryptocurrencies prices are however difficult to forecast, due to their volatility and dynamism. This study tries to construct a robust predictive tool, focusing on Bitcoin closing price. Three models are proposed: two of them exploit Long Short Term Memory (LSTM) neural network, one trained on raw prices and the other on stazionarized prices, whereas the third one is based on Support Vector Regression algorithm.

1 Introduction

Bitcoin (BTC) is the most known and traded cryptocurrency. It consists of a decentralized and peer-to-peer network and a public ledger, the blockchain, where all transactions are recorded. Complex protocols, such as the proof-of-work, are employed to reach consensus over the network and to ensure the security of Bitcoin's possessions. [1].
After Bitcoin, many others virtual currencies have been created, infiltrating traditional financial market and making crypto money one of the most popular and promising source of profit.
Last years have shown that cryptocurrencies prices, as well as the stock market's ones, are affected by many factors, such as pandemic bursts and political issues. These sources of uncertainties determine high fluctuations and volatility of digital currencies prices, outlining non stationary and unpredictable behaviors. In this scenario, a solid and reliable forecasting tool could help investors when making decisions about their cryptocurrencies portfolio.
This work implements 2 methodologies, long short-term memory (LSTM) [2] and support vector regression (SVR) [3], in order to make one-step ahead prediction of BTC closing price given previous days prices. Two of the proposed models are applied on Bitcoin raw data, while one of the LSTM architecture has been trained and tested on stationarized data.
The effectiveness of the proposed models are evaluated considering the Root Mean Square Error (RMSE), the Mean Absolute Error (MAE) and the accuracy in predicting price movements.

2 Related work

Many works regarding cryptocurrency price prediction are available in the literature. Some of them exploit only currencies price and some, more complex, consider both price and directional movement in order to enhance the accuracy. Ferdiansyah et al.[4] proposed a RNN built over several LSTM modules. They performed experiments over a 4 years period, starting from 27-06-2014 until 27-06-2019, with a split of 3 years for training and validation, and 1 year for testing. They have achieved, in predicting the BTC closing price, 288.59 error using RMSE as metric.
Livieris et al[5] proposed a different approach: a multiple-input cryptocurrency model was developed, taking different cryptocurrencies prices as input. The goal is to extract and exploit knowledge from mixed data in order to make the final prediction more accurate. Each input is processed independently through convolutional and LSTM layers. Then, they are merged together into a concatenate layer, followed by a series of Dense layers. They considered prices of Bitcoin, Ethereum and Ripple from 01-01-2017 to 31-10-2020 , using the following split: training set from 01-01-2017 to 18-02-2020, validation from 01-01-2020 to 31-05-2020 and test set from 01-06-2020 to 31-10-2020. The performance shown by their architecture on BTC price prediction is around 257.728 RMSE error and it shows an accuracy of 53.04% in predicting the movement.

3 Methods

3.1 Time series

A time series is a sequence of time ordered data points. Time series analysis aims at extracting meaningful patterns in order to forecast future values based on previous observations.
Given observations up to time t:

$$(y_0, y_1, \ldots, y_t)$$

time series forecasting consists in predicting k future values:

$$(\hat{y_{t+1}}, \ldots, \hat{y_{t+k}}) = \hat{f}(y_0, y_1, \ldots, y_t)$$

where f is the k-step ahead predictor.

3.1.1 Stationarity

A **stationary** time series is one whose properties are time-invariant. Time series characterized by trends or seasonality are not stationary, since their statistical properties, such as mean and covariance, change over time.
Stationarity can be assessed by plotting the time series and visually seeking for trends or seasonal components, but there are more formal and precise methods

to detect stationarity.
One of the most popular is the **Augmented Dickey Fuller** (ADF) test [6], which consists in testing the null hypothesis that the time series has a unit root, hence it is not stationary. The null hypothesis is rejected if the test statistic is less than a given critical value.
If the ADF test shows that a time series is not stationary, it is possible to perform some transformation in order to make it stationary.

First Difference method is one of the techniques that can be exploited. It consists in replacing each data point with the difference with the following one. Given the values at time t and $t-1$, the new value at time t would be:

$$Y_t' = Y_t - Y_{t-1} \tag{1}$$

This methodology is however characterized by long term memory loss, since differencing limits past information to propagate through the values.
A possible solution is to employ **Fractional difference** [7], which makes a time series stationary while preserving some long term memory. Fractional difference is defined by the equation:

$$\tilde{Y_t'} = (1-L)^d Y_t \tag{2}$$

where L is the lag operator and d the order of the fractional differencing.

3.2 LSTM

Classical Recurrent Neural Network (RNN) suffer from the vanishing gradient problem [2], which does not make them suitable to learn long-term dependencies. Long Short Term Memory (LSTM) network is a specific type of RNN designed to solve these errors and thus able to deal with time series characterized by long-term memory.
LSTM architecture resembles the way in which a RNN is built. It is composed by a chain of cells and each of these modules contains a state, which represents the memory of the cell, and some gates. Figure 1 shows an LSTM cell.
Gates are the way in which each cell decide how much information can go through or be discarded. They are three:

- Input gate: estimates the importance of the information conveyed by the input. It is characterized by the following equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

  where t is the time step, i_t the input gate, W the weight matrix, b the bias vector, h_{t-1} the previous hidden state, and $\tilde{C_t}$ the value generated by tanh. The sigmoid function is used to filter the information, by discarding it if the result of (3) is 0. Information that goes through the tanh (4) function is kept only if sigmoid result is 1. The output values generated form the activation functions will be a value between -1 and 1.

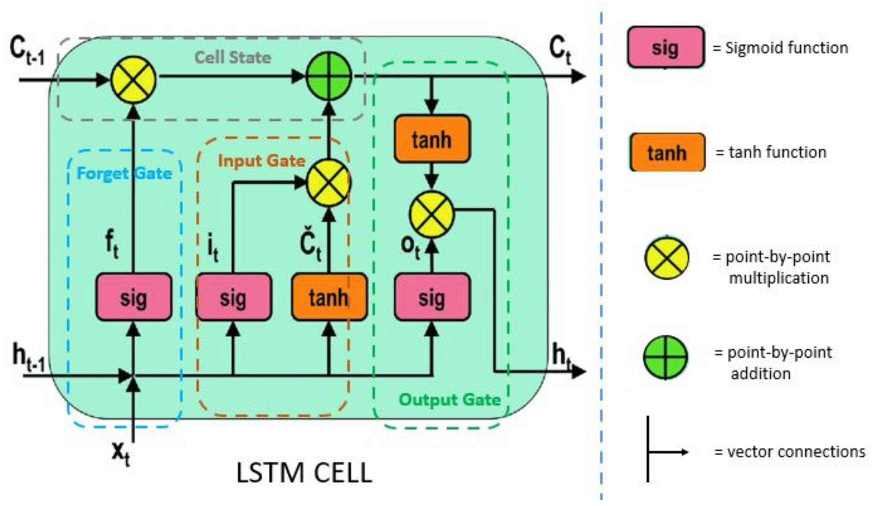

Figure 1: LSTM module. (Source: Introduction to Long Short Term Memory (LSTM) [8])

- Forget gate, denoted by the following equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

  It establishes which information can be discarded and ignored, hence dealing with the vanishing gradient problem.

- Output gate, determines the value for the next hidden state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t * tanh(C_t) \tag{7}$$

  The values of the current state x_t and previous hidden state h_{t-1} are passed into a sigmoid function (6), the new cell state C_t in passed into a tanh function, and those results are eventually multiplied point-by-point(7).

As for the state, it is updated according to the following equation based on the outcome of the input and forget gate.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

C_{t-1} is the state at the previous timestamp. If f_t is 0, the previous cell state C_{t-1} will be dropped and the new cell state C_t will be updated with the input vector i_t, after performing point-by-point multiplication. [9]

3.3 SVR

Support Vector Machines (SVMs) are a machine learning class of models and algorithms suited to solve classification tasks. In 1996, Harris Drucker et al. [3] introduced a new version called Support Vector Regression (SVR), that can be used to address regression problems.
Unlike classical regression techniques, which try to minimize the error between true values and predictions, SVR tries to fit within an hyperplane as many points as possible. The hyperplane is found by mapping the original input space into a space of higher dimension, which is the projection of the input features by means of a kernel function.
SVR is formulated as follows:

$$f(X) = w^T \Phi(X) + b \tag{9}$$

where $f(\cdot)$ is a non-linear function, X is the matrix containing the data points, w is the weight vector, b the bias and $\Phi(X)$ is the new feature space, obtained by applying the kernel to the input matrix. Eq. 9 can be transformed into a convex optimization problem:

$$\begin{array}{ll} minimize & \frac{1}{2} w^T w \\ subject\,to & \begin{cases} y_i - (w^T \Phi(X_i) + b) \leq \epsilon \\ y_i - (w^T \Phi(X_i) + b) \geq \epsilon \end{cases} \end{array} \tag{10}$$

where $\epsilon (\geq 0)$ is fixed by the user and represents the maximum acceptable deviation.
To solve eq. 9 we assume $f(x)$ existing, which also implies that the optimization problem is feasible. However, this is not always the case and it may be necessary to take errors into consideration.
The formulation of this new problem is the following:

$$\begin{array}{ll} minimize & \frac{1}{2} w^T w + C \sum_{i=1}^{m} (\xi_i^+ + \xi_i^-) \\ subject\,to & \begin{cases} y_i - w^T \Phi(X_i) - b \leq \epsilon + \xi_i^+ \\ w^T \Phi(X_i) + b - y_i \leq \epsilon + \xi_i^- \\ \xi_i^+, \xi_i^+ \geq 0 \end{cases} \end{array} \tag{11}$$

$C(> 0)$ is a fixed regularization constant representing the weight of the sum of the slack variables. $(w^T w)$ represents the original objective function, while the new term is called empirical term and measures the ϵ-intensive loss function.
Eq. (11) penalizes all data points whose y-values differ from $f(x)$ by more than ϵ. The ξ_i^+ and ξ_i^- constants correspond to the upper and lower deviations, as shown in fig. 2. All data points inside the ϵ-tube do not contribute to the regression model. Data points outside the tube are called support vectors, and they are used in determining the decision function.
To solve eq.(11) Lagrangian multipliers must be introduced, allowing to extend the previous formulation to non-linear functions. From the derivative of the loss function, one can substitute into eq.(9) and obtain the following formulation:

$$f(x) = \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) K(x_i, x_j) + b \tag{12}$$

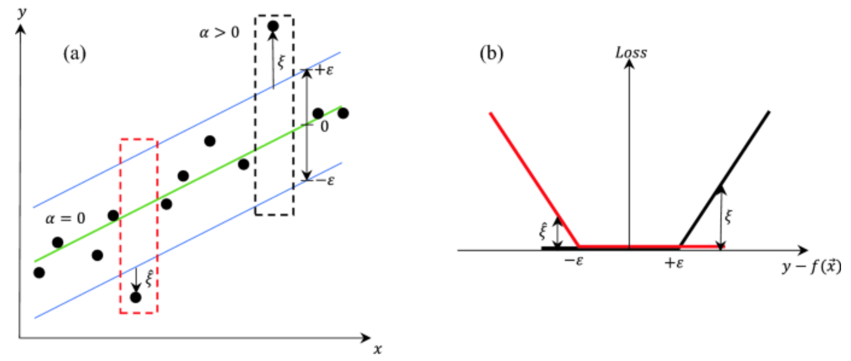

Figure 2: Left: ϵ Tube. Right: ϵ intensive loss function

Kernels Function

Linear	$K(x_i, x_j) = x_i^T x_j$
Poly	$K(x_i, x_j) = (x_i^T x_j + 1)^d \, d = 1, 2 \ldots$
RBF	$K(x_i, x_j) = exp(\gamma \left\| x_i - x_j \right\|^2)$

Table 1: Kernels function

where α_i^+ and α_i^- are Lagrangian multipliers, and $K(x_i, x_j)$ is the kernel function.
Table 1 shows possible kernel functions to use within SVR.

3.4 Adam

Adam, which stands for Adaptive Moment Estimation, is an efficient stochastic gradient descent algorithm. It is based on the estimation of the first and second order moments, which are computed according to the following formulas:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{14}$$

where β_1 and β_2 are hyperparameters, usually set to 0.9 and 0.999 respectively, and g_t is the gradient of the loss function. As Kingma et Al. [10] pointed out, since the moments are both initialized as vectors of zero's, they are biased towards zero, especially during the first timesteps. This bias is corrected by computing the following estimations:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{15}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{16}$$

The weights update rule is the same used in RMSprop:

$$\theta_{t+1} = \theta_t - \frac{\alpha * \hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon} \tag{17}$$

where α is the fixed learning rate, usually set to 0.0001 and epsilon is a constant for numerical stability, usually set to 10^{-8}

4 Dataset

Experiments have been carried out on the dataset provided by Yahoo finance stock market, based on the USD Exchange rate and publicly released by CCC - CryptoCompare Currency in USD.
The structure of the dataset is shown in table 2. It contains Opening, High, Low and Closing prices in USD dollars and the daily trading Volume.

Date	**Open**	**High**	**Low**	**Close**	**Volume**
01-01-2017	963.658	1003.080	958.698	998.325	147775008
02-01-2017	998.617	1031.390	996.702	1021.750	222184992
...	...	...	...	...	...
30-10-2020	13437.874	13651.516	13136.198	13546.522	30581485201
31-10-2020	13546.532	14028.213	13457.530	13780.995	30306464719

Table 2: Yahoo finance stock market dataset

4.1 Train, test, validation split

For evaluation purposes data was collected and divided according to the period presented in [5], from 01-01-2017 up to 31-10-2020.
The dataset was splitted twice: the first set, for training and validation, contains dates up to 31-05-2020, whereas the remaining samples are used for testing. 80% of the first set constitutes the training set and 20% the validation set.

4.2 Non stationary behavior

ADF test	
ADF test statistic:	-2.263287
p-value:	0.184082
Critical values:	
1%:	-3.435
5%:	-2.864
10%:	-2.568

Table 3: ADF test on BTC closing price from 01-01-2017 up to 31-10-2020.

Fig. 3 shows the BTC closing price over the period considered, which is characterized by a clear non stationary behavior. Table 3 presents the results for the ADF test, showing a test statistic higher than any of the critical values. Since there is no reason to reject the null hypothesis, the time series is not stationary.

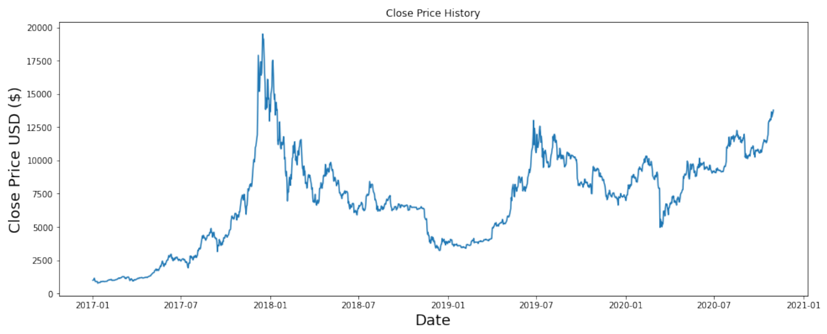

Figure 3: Closing price BTC-USD

5 Experiments

5.1 Models

LSTM with non stationary data($LSTM_{ns}$) consists of a LSTM layer of 256 units, a dense layer of 128 neurons with activation function ReLu, a Dropout layer with rate 0.1 and an output layer of one neuron. The method is compiled using as optimizer the Adam optimizer and as loss function the Mean Squared Error.
SVR with non stationary data(SVR_{ns}) is characterized by a radial basis function (RBF) as kernel function, $\gamma = 0.2$, $\epsilon = 0.002$ and the regularization parameter C=0.05
LSTM with stationary data ($LSTM_s$) consists of a LSTM layer of 256 units, a LSTM layer of 128 units, two dense layers of 32 neurons with activation function ReLu and an output layer of one neuron. The method is compiled using as optimizer the Adam optimizer and as loss function the Mean Squared Error.

5.2 Data preprocessing

This section contains the explanation of how the data is manipulated before being input to the different models. First, the dataset is split into training, validation and test sets. Since the input data are time series, the samples are not shuffled, in order to keep the time information. Then, the data are normalized between 0 and 1, so as to ease the training phase. To normalize the data has been used the MinMaxScaler, which scales and translates each feature individually such that it is in the given range, so between 0 and 1. The following formula is the one used by the scaler to normalize the data:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{18}$$

In the $LSTM_s$ method, before the normalization, the data are transformed into stationary data. So fractional difference is applied and the ADF test is performed to check that the transformation was successful.
The LSTM method has an input whose dimension is number of timesteps times number of features, where the number of timesteps represents how many input points are used to make the prediction. So, the data of the $LSTM_s$ and the $LSTM_{ns}$ are reorganized to match the input form requested by the LSTM method. The $LSTM_s$ is created using 7 days of timestep and only 1 feature, which is the close price. Whereas, the $LSTM_{ns}$'s input has 7 days of timestep and 5 features, the open price, the highest and the lowest price, the close price and the volume.

5.3 Results

In this section, several results of the evaluated models are presented. The models taken into account, present different architectures, training parameters and training data. As said in previous section, two of the models work with non stationary data, while one of them with stationary one. Furthermore, the examined networks were compared with the ones proposed in [5]. Performance metrics, shown in table 4, were computed according to the prediction of the same test set of [5].
The evaluation metrics considered were:

- Root mean squared error

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2} \tag{19}$$

- Mean absolute value

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \tag{20}$$

- Movement accuracy

$$\begin{gathered}\Delta y_t = y_t - y_{t-1} \qquad t = 1, \dots, N \\ \Delta \hat{y_t} = \hat{y_t} - \hat{y_{i-t}} \qquad t = 1, \dots, N \\ S = \{t : sign(\Delta y_t) == sign(\Delta \hat{y_t})\} \\ Mov.Acc. = \frac{|S|}{N}\end{gathered} \tag{21}$$

(way to compute the accuracy of price increasement or decreasement on the following day with respect to the today's price)

where N is the number of forecasts, y_t is the actual value, $\hat{y}_t$ is the predicted value, Δy_t and $\Delta \hat{y}_t$ are the difference with the previous day value, S is the set for which the sign of Δy_t is equal to the one of $\Delta \hat{y}_t$, and $| \cdot |$ is the cardinality operator.
All model exhibited similar performance, regarding the performance metrics MAE, RMSE. The models working with non stationary data outperformed the implementation with stationary data, due to the data transformation applied on $LSTM_s$; the stationarized time series loses memory properties during its processing and this affects the model training. More specifically, the evaluated models present the following results:

- $LSTM_{ns}$ achieves 244.99, 351.03 and 50.66% for MAE, RMSE and movement accuracy metrics, reporting the best results on MAE with respect to the other 2 implementations.
- SVR_{ns}, 249.88, 350.89 and 47.36% respectively, achieving the best RMSE error, but also the lowest movement accuracy.
- $LSTM_s$, 433.76, 556.62 and 53.95% respectively, resulting in the worst metric errors, but higher movement accuracy than $LSTM_{ns}$ and SVR_{ns}.

Stationary data showed a performance slightly worse than not stationary one, since they lose too much information that LSTM layers is not able to exploit. The results of the proposed models differ from the one achieved by Ioannis et al., which are presented in table 5. This is due to the different architectures implemented and different data employed. Indeed, Ioannis et al. models were trained also on other cryptocurrencies to better exploit different patterns.

Proposed models results			
Model	**MAE**	**RMSE**	**Accuracy**
$LSTM_{ns}$	**244.99**	351.03	50.66%
SVR_{ns}	249.88	**350.89**	47.36%
$LSTM_s$	433.76	556.62	**53.95%**

Table 4: Performance of the evaluated models

Ioannis et al. results			
Model	**MAE**	**RMSE**	**Accuracy**
$Model_1$	169.817	256.688	55.03%
$Model_2$	169.604	256.318	53.64%
$MICDL$	170.761	257.728	53.0%

Table 5: Performance of the Ioannis et al [5] models

6 Conclusion

The Bitcoin market is influenced by many uncertainties factor, such as political issues, economic events and pandemic situation. Trying to predict future values and movements in this context may not be enough to make the decision of investing in stocks market.
The models proposed in this work have shown good results in predicting Bitcoin closing price. As for MAE and RMSE errors, good outcomes were obtained. However, the achieved accuracy is unsatisfactory, since it is comparable with the one of a random choice. Hence, the proposed models should be considered for price prediction rather than movement prediction.
Future works will focus on implementing modified LSTM layers and adding convolutional layers to better exploit hidden patterns. Far research on how to deal with stationary data will be carried out.

References

[1] Satoshi Nakamoto(2008) - *Bitcoin: A peer-to-peer electronic cash system.*

[2] Sepp Hochreiter and Jürgen Schmidhuber(1997) - *Long Short-Term Memory*, MIT Press, December 1997

[3] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, Vladimir Vapoik - *Support Vector Regression Machines*, Advances in Neural Information Processing Systems 9 (NIPS 1996)

[4] Ferdiansyah Ferdiansyah, Siti Hajar Othman, Raja Zahilah Raja Md Radzi, Deris Stiawan and Yoppy Sazaki(2019) - *A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market*, IEEE Congress on Evolutionary Computation (CEC)

[5] Ioannis E. Livieris, Niki Kiriakidou, Stavros Stavroyiannis and Panagiotis Pintelas (2021) - *An Advanced CNN-LSTM Model for Cryptocurrency Forecasting*, Electronics 2021, 10, 287.

[6] J.Humberto Lopez - *The power of the ADF test*, Economics Letters(1997)

[7] J. R. M. Hosking - *Fractional Differencing*, Biometrika. Apr. 1981 Vol. 68. No. 1. P. 165-176

[8] Shipra Saxena - *Introduction to Long Short Term Memory (LSTM)*, Analytics Vidhya (May, 2021)

[9] *Understanding LSTM Networks*, (May, 2021)

[10] Diederik P. Kingma, Jimmy Lei Ba *Adam: a method for Stochastic Optimization*, (May, 2021)