# SHORT-TERM FORECASTING IN BITCOIN TIME SERIES USING LSTM AND GRU RNNs

M. Caux[1], F. Bernardini[2], J. Viterbo[2]

[1] Universidade Federal Fluminense, Brazil
marcelocaux@id.uff.br
[2] Universidade Federal Fluminense, Brazil
{fcbernardini, viterbo}@ic.uff.br

**Abstract.** In recent years, bitcoin has become a very attractive investment in financial industry, which is not controlled by governments, but is based on trust between transfers under the technology of block chain. Hence, forecasting future bitcoin cryptocurrency values is a problem that has attracted the attention of many researchers in the field, while proving to be a very challenging problem. This work presents an experimental analysis using LSTM and GRUs for forecasting bitcoin values in a minute-granulated time for the entire next day. To this end we also present our methodology for conducting the experiments. The final goal is to create the core of a financial prediction tool around the RNNs. In our experiments, we achieved interesting results such as a SMAPE of 0.0002, a RMSE of US$ 3.844 and a rRMSE of 0.0028 in a day where bitcoin rates vary from US$ 13.2K and US$ 14.6K, surpassing the results of SMAPE found in the literature and proposed limit of SMAPE smaller than 0.007 for forecasts.

CCS Concepts: • **Applied computing**;

Keywords: kdmile, Bitcoin, Financial Forecasting, Recurrent Neural Network, LSTM, GRU

## 1. INTRODUCTION

Forecasting financial assets has long played a very important role in financial industry, and diverse statistical and computational techniques have been used to tackle this problem and supporting decision-making in different financial market segments. In recent years, bitcoin has become a very attractive investment, capable of generating giant profits or losses for investors, and which has a peculiar behavior, as it is not controlled by governments, but is based on trust between transfers under the technology of block chain. Hence, forecasting future bitcoin cryptocurrency values from its historical value series is a problem that has attracted the attention of many researchers in the field. Several machine learning algorithms and statistical methods for constructing models have been used. We could find the following constructed models to this end: Decision Tree [Rathan et al. 2019], Linear Regression [Rathan et al. 2019] [Chauhan et al. 2019], Support Vector Regression [Chauhan et al. 2019], ARIMA [Yamak et al. 2019] [Mangala et al. 2019], Neural Network Regression [Chauhan et al. 2019] and Recurrent Neural Networks (RNN) [Chauhan et al. 2019; Yamak et al. 2019; Mangala et al. 2019]. These works dedicated to forecast bitcoin quotes in different ways: predict the direction (trend) of quotes for next day, predict the closure quote for next day and predict quotes for next five days. However, none of these works tried to predict quotes minute by minute, we have not found similar approach in our research. This forecast, if precise, could give a complete view of bitcoin behavior during one day in the future, creating a powerful tool any investor could use in a short term trade of bitcoin. In this way, predicting bitcoin cryptocurrency values minute by minute is challenging.

There are many frameworks and tools for supporting models construction using machine learning

algorithms, being Weka and Scikit-Learn the most used among members in academy. For constructing RNNs and other types of deep neural networks, Keras and Tensorflow offers implementation of many techniques for helping the difficult task of creating models. Dealing with time series requires some attention regarding to how to process the data due to the sequence of the data in time, which can present seasonality, trend and stationary components [Cowpertwait and Metcalfe 2009] that are not present in non-time series data.

The objective of this work is to create a work method that allows the daily construction of RNN models for predict all bitcoin values minute by minute for one day, starting at 00:00 until 23:59, to be used as the core of a bitcoin forecast financial tool. In order to make it possible, we used a bitcoin time series with minutes granularity so RNNs can learn the bitcoin behavior at the same pace we want to forecast and evaluate two different architectures of RNN, LSTM and GRU, this choice comes from the bibliographic review and the statement that RNN-based models dominate the literature review for financial time series forecast found at [Sezer et al. 2020]. Our purpose imposes the challenge in finding the best configuration for RNNs, best optimizer and hyper-parameters, the more adequate size for training and test data sets, and avoidance of training issues like over-fitting, endless training or lack of convergence, the measure of success was defined as a SMAPE measure under 0.007, which was the best value found in literature review and can guaranty exceptional forecasts to be the core of a bitcoin forecast financial tool. Our results show that GRU outperform LSTM and it was capable to overcome the proposed SMAPE measure under 0.007 for all forecast dates.

This work is organized as follows: Section 2 describes theoretical reference for allowing to understand our proposed methodology, Section 3 presents a literature review, Section 4 presents our proposed methodology, Section 5 presents our results and finally Section 6 presents our conclusions.

## 2. THEORETICAL REFERENCE

### 2.1 LSTM — Long-Short Term Memory

LSTM networks [Gulli and S. 2017] are a distinct type of Recurrent Neural Networks. They are able to learn long-term dependencies and are very popular for working with sequential data such as time series data. LSTM cell uses an input gate $i_t$; a forget gate $f_t$; and an output gate $o_t$. These gates define whether the data can pass through or not depending on the data's priority. Gates also enable the network to learn what to save, what to forget, what to remember, what to pay attention, and what to output. The cell state and hidden state are used in gathering data for processing in the next state. The structure of the LSTM unit is shown at Fig. 1.

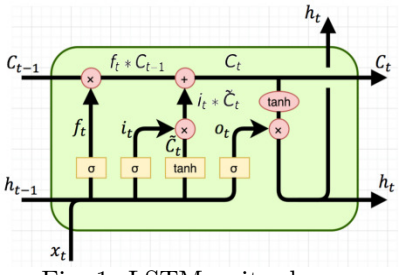


Fig. 1: LSTM unit scheme.

**The gates have the following equations**

—Input Gate: $i_t = \sigma(W_i h_{t-1} + W_i h_t)$

—Forget Gate: $f_t = \sigma(W_f h_{t-1} + W_f h_t)$

—Output Gate: $o_t = \sigma(W_o h_{t-1} + W_o h_t)$

—Intermediate Cell State: $\tilde{C} = tanh(W_c h_{t-1} + W_c h_t)$

—Cell State (next memory input): $c_t = (i_t * \tilde{C}) + (f_t * c_{t-1})$

—New State: $h_t = o_t * tanh(C_t)$

—$X_t$ : Input Vector

—$h_t$ : Output Vector

—***W, U and f:*** Parameter matrices and vector

$\otimes$ Element-wise multiplication

$\oplus$ Element-wise summation/*concatenation*

### 2.2 GRU — Gate Recurrent Unit

The Gated Recurrent Unit (GRU) [Gulli and S. 2017] is also a Recurrent Neural Network, likewise the LSTM. It, however, has a less complicated structure compared to LSTM. It lacks an output gate, though it has an update gate $z_t$ and a reset gate $r_t$. These gates are vectors which decide what information should be passed to the output. Fig. 2 shows the GRU unit. The Reset gate defines how to combine the new input with the previous memory. The definition of how much of the last memory to keep is done by the Update gate.

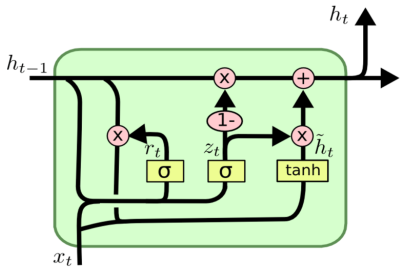

Fig. 2: GRU unit scheme.

**Each element of Fig. 2 has the following equations:**

—Update Gate:
$z_t = \sigma(W_z h_{t-1} + U_z x_t)$
—Reset Gate:
$r_t = \sigma(W_r h_{t-1} + U_r x_t)$
—Cell State:
$\tilde{h}_t = tanh(W_C(h_{t-1} * r)U_c x_t)$
—New State:
$h_t = (z * c) + ((1 - z) * h_{t-1})$
$\otimes$ Element-wise multiplication
$\oplus$ Element-wise summation/*concatenation*

### 2.3 Metrics for Evaluating the RNNs

In order to identify which quality measures to apply, it is needed to understand the characteristics of the data in the time series to be used for training. Shcherbakov *et al* [Shcherbakov et al. 2013] presents an interesting comparison of different forecast error measures with their drawbacks and recommendations of usage based on the time series data characteristics. He also states that SMAPE (Symetric Mean Absolute Percentage Error) is a good measure for forecasting time series. So, we selected this measure as the one to be used to evaluate our models for predicting one day of bitcoin quotes. $SMAPE$, is an accuracy measure based on relative errors. Aside $SMAPE$, we also used $MAPE$(Mean Absolute Percentage Error), $RMSE$, which is a frequently used measure of the differences between actual values and forecast values and $rRMSE$, the normalized $RMSE$, which facilitates the comparison of different regression models. All these measures are commonly used in applications of machine learning for regression problems.

## 3. LITERATURE REVIEW

Chauhan, Kumar and Saini [Chauhan et al. 2019] used five machine learning algorithms (Linear Regression, Support Vector Regression, Neural Network Regression, LSTM RNN and Convolutional NN) for predicting bitcoin values in a random time. This work does not reveal much details about the data series used or models configuration. We could understand that they used a LSTM RNN with only one layer with four neurons and one dense output. Analysing their available graphs, we identified prediction granularity as daily. For evaluating the models, they used RMSE for regression models and precision measure for RNN and CNN models. They concluded that LSTM presented better results over the other models.

Mangata et al [Mangala et al. 2019] tested four algorithms for constructing models to predict the direction (up or down) of bitcoin price for the next day. It is worth to observe that they modeled the problem as a classification learning problem. They used ARIMA statistical analysis, Logistic Regression, Suport Vector Machine and LSTM, being the last three ones machine learning algorithms

to process a bitcoin minute-granulated time series. Their premise is that investor have a chance to avoid loss in bitcoin trade. Their conclusion was that ARIMA performs better than LSTM for next days but is poor for 5-7 days prediction; LSTM perform consistently up to 6 days; and the accuracy were 53% and 50% respectively.

Rathan, Sai et Manikanta [Rathan et al. 2019] tested two machine learning algorithms, Decision Tree and Linear Regression models, to process a bitcoin daily-granulated time series. The authors aim to forecast 5 days of bitcoin quotes and claims an accuracy in price prediction of 95.8% and 97.5% for Decision Tree and Regression respectively.

Uras et al [Uras et al. 2020] aims to predict the daily closure value of bitcoin using the following learning algorithms to process a bitcoin daily granulated time series: Multi Layer Perceptron, LSTM, Simple Linear Regression and Multiple Linear Regression. The authors have used Keras to implement the LSTM RNN, with one LSTM layer of 64 units connected to one dense output with shuffle set to false, using Adam optimizer and MSE as loss measure, 600 epochs with batch size of 72. The authors conclude their better results comes when shortened the regime times. Those regime times are floating time windows that divides the full time series in smaller pieces. Their initial training size was 200 days, obtaining a MAPE of 0.0084. When the regime time was reduced to 120 days, MAPE decreased to 0.0070. So, they concluded this approach has allowed to avoid the "random walk problem". The best accuracy measures found for bitcoin forecast were using LSTM, where MAPE = 0,007 and rRMSE = 0,012.

## 4. OUR METHODOLOGY

In this section we define the steps of our methodology, each one having a goal and all steps together generates the desired results. Fig. 3 shows the sequence of steps, more details are shown at section 5.

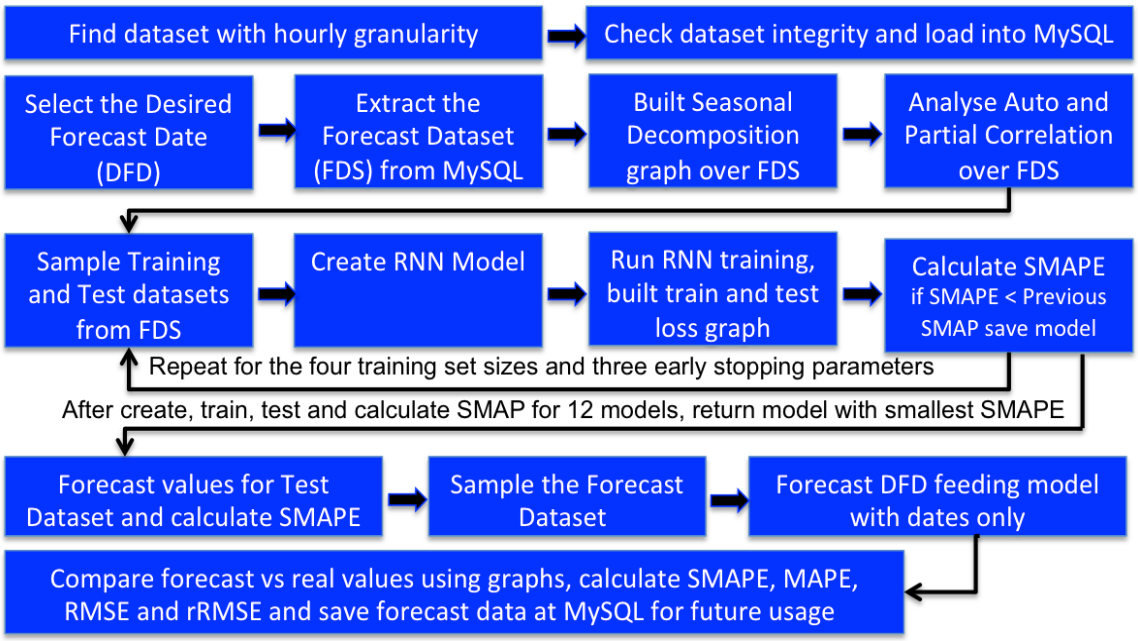


Fig. 3: The Methodology Workflow.

## 5. EXPERIMENTAL AND RESULTS ANALYSIS

### 5.1 Experimental Environment

—Hardware: Intel i7 8th gen., 24GB RAM, 1TB SSD, Nvidia GTX1060 6GB RAM

—Software: Ubuntu 18.04, Python 3.7.2, Anaconda 4.6.2, Jupyter notebook 4.4.0, Tensorflow 1.11.0, Keras 2.2.4, CUDA 9.2.148.1, numpy, pandas, scipy, sklearn, math, numba, matplotlib, for Graphs, used plotly and MS Excel

—RNN configuration:
  —LSTM: Two 512 neurons layers and one neuron dense output layer
  —GRU: One 512 neurons layer and one neuron dense output layer
  —Both: Optimizer=adam, loss=mean_squared_error, batch_size=64

### 5.2 Dataset Description and Analysis

Kaggle(`https://www.kaggle.com/mczielinski/bitcoin-historical-data`) sources our dataset in a set of different text files imported to MySQL in order to be organized and extracted according to desired periods of time, table I describes the dataset. This dataset covers bitcoin values from 12-31-2011 to 08-12-2019 in a minute base. However, the period time selected for our analysis is much smaller and represents the middle of the strongest rally of bitcoin ever, *i.e.*, 12-15-2017 to 01-15-2018, a total of 46,080 data points. This period of time shows a sequence of peaks and valleys of quotes, representing a very challenging behavior to be learned, giving a good chance to check the ability of RNN in learning it. In this way, the first question that comes up was how to train the RNNs in order they could learn this "chaotic" behavior.

### 5.3 Creating the Forecast Dataset ($FDS$) for the Desired Forecast Day ($DFD$)

In order to train the RNN and forecast the quotes of an entire day, given by the $DFD$, we construct a Forecast Dataset $FDS$[1]. Tipically, this $FDS$ contains all quotes for the $DFD$ plus a predefined number of previous days. For the forecast, we used only the TimeStamp field converted to the format 'yyyy-mm-dd hh:mm:ss' and the Weighted_Price column. To answer the question on how big the $FDS$ should be, we tried different training set sizes that could represent the observed behavior in the near past. We tested several sizes on 125 simulations and selected window sizes of 10, 15, 24 and 30 days for training set size, as they have shown best results. Test set size was set for 2 days, twice the forecast size desired, also keeping the training set closer to the $DFD$. This proximity is intended to learn close changes in bitcoin behavior. In short, to forecast one day you need to have the data series for $DFD$ plus previous 32 days. In order to forecast quotes for the period between 12-15-2017 and 01-15-2018 (32 days), the $FDS$ contains all quotes from 11-13-2017 (12-15-2017 minus 32 days) to 1-15-2018 in a total of 64(30+2+32) days and 92160 data points. In order to sample the Training

[1]For technological matters, we select records from MySQL and process the result dataset through a Python script

| Column Name | Type | Brief Description |
|---|---|---|
| TimeStamp | Unix Timestamp | Timestamp of minute of a date |
| Open | Currency USD | First value of the minute |
| High | Currency USD | Highest value of the minute |
| Low | Currency USD | Lowest value of the minute |
| Close | Currency USD | Last value of the minute |
| Volume_(BTC) | Real | No. of bitcoins negotiated |
| Volume_(Currency) | Currency USD | Total value negotiated |
| Weighted_Price | Currency USD | Weighted Price for the minute |

Table I: Kaggle dataset layout.

| Training set size | Begin | Finish |
|---|---|---|
| 10 | $DFD$ minus 12 days | $DFD$ minus 3 days |
| 15 | $DFD$ minus 17 days | $DFD$ minus 3 days |
| 24 | $DFD$ minus 26 days | $DFD$ minus 3 days |
| 30 | $DFD$ minus 32 days | $DFD$ minus 3 days |

Table II: Sampling Training Datasets.

dataset for the $DFD$ follows table II, the Test dataset always contains $DFD$ minus 2 and $DFD$ minus 1 days.

### 5.4 Autocorrelation Analysis and Seasonal Decomposition

The autocorrelation and Partial Autocorrelation of our time series shows it is non stationary and Seasonal Decomposition shows a strong Trend and a very small seasonality with small residual, those results were found for all forecasts done.

### 5.5 The Proposed Training Technique

Section 5.3 shows the proposed methodology used for constructing our $FDS$. We used four different sizes of training set: 10, 15, 24 and 30 days. We used those sizes to achieve better training quality and consequently get more precise forecasts. At the first simulations we have perform the RNN training for each of the four training set sizes, select the model with best SMAPE and perform the forecast for desired date, for some forecasts the SMAPE was smaller than 0.007, but many forecasts got results over this barrier, so clearly, the usage of different sizes of training set alone was not sufficient.

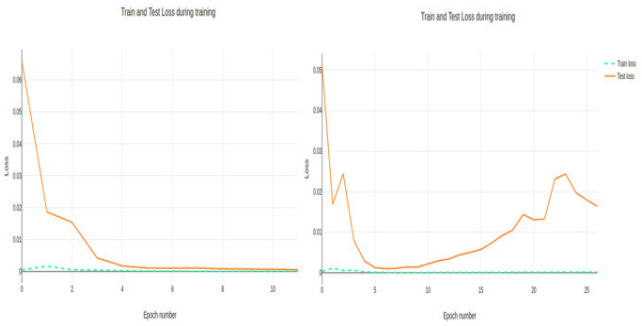


Fig. 4: Training without and with overfitting

After each training, we generate a graph showing train and test loss, analysing those graphs we saw that every time a forecast doesn't reaches the desired SMAPE, the training test loss curves shows an ascending curve at final training epochs, characterising training overfit. Fig. 4 shows the overfitting with ascending curve at the end in the right graph, a good quality training is shown at left, note the stable and descending curve at the end. The conclusion of this analysis was that training overfit needed to be eliminated. To correct training, the **EarlyStopping** training function was used. It is composed by the following parameters: *monitor*, which defines what to be monitored; *val_loss*, which is the loss in testing phase of each epoch; *min_delta*, which is the minimum change in the monitored quantity to qualify as an improvement; and *patience*, which defines number of epochs that produced the monitored quantity with no improvement after which training will be stopped, this parameter works together with min_delta as follows. If a sequential number of epochs (the patience) doesn't produce an improvement of min_delta in val_loss, the training stops and the model is considered trained. We call each combination of these parameters a different configuration of

| Set No. | *min_delta* | *patience* |
|---|---|---|
| 1 | 0.005 | 8 |
| 2 | 0.00005 | 8 |
| 3 | 0.001 | 4 |

Table III: *min_delta* and *patience* sets.

**EarlyStopping**. By combining 4 sizes of training datasets with 3 configurations of **EarlyStopping** we create 12 RNN models and select the model with best SMAPE on training to forecast $DFD$, we call this technique **12 training technique**. After experiments, three sets of combinations of *min_ delta* and *patience*, see table III, shows to be effective in eliminate overfit.

### 5.6 Feeding Training, Test and $DFD$ datasets to RNN Models

Fig.3 shows the cyclical process where Test dataset is sampled once since it is fixed, but Training dataset is sampled for each size (10,15, 24 and 30 days) combined with the three configuration of early stopping in a double loop structure, on each pass of the internal loop a new model is created and feed with the training and test datasets combined with one configuration of early stopping and trained, training results are analysed graphically and SMAPE is calculated and compared with previous model SMAPE, if the last model trained has an smaller SMAPE, we save current Training dataset size, configuration of early stopping, SMAPE and the model itself as the $bestModel$, after 12 models created and trained we have one $bestModel$ and parameters selected. With this $bestModel$ we run a second training using the selected Training dataset size, the selected configuration of early stopping and Test dataset and perform a forecast on the Test dataset in order to check how good was the training comparing predicted to observed values. The next step is sample the $DFD$ dataset from the $FDS$, composed by all the 1440 time points of the $DFD$, and feed the $bestModel$ with only the date component of $DFD$ dataset, those dates were never seem by the $bestModel$, run the $bestModel$ forecast and finally calculate SMAPE, RMSE, rRMSE and MAPE and plot forecast vs real values for $DFD$.

Applying our work technique, we achieved an SMAPE for the $DFD$ smaller than 0.007. The output of our work technique is the model LSTM or GRU with the smallest SMAPE for the training.

### 5.7 Results

The SMAPE achieved by each RNN for predicted dates is plotted at Fig. 5. Using batch_size=64 LSTM has failed to achieve SMAPE smaller than 0.007 for one date, this issue was corrected by changing batch_size to 32, with a huge cost for training, this correction results in SMAPE smaller than 0.007 but do not exceeded GRU results for the date. The GRU also shows less variation for the best and worst forecasts. For the best forecasts LSTM and GRU perform equally, but the worst forecasts they show a large difference in SMAPE, 0.0025 vs. 0.0082 which is not negligible, see Fig.7, other measures confirm GRU best performance, worst RMSE US$46.3488 vs. US$157.4199, worst rRMSE 0.0208 vs. 0.0757. The rRMSE measure shows how GRU performs forecasts in a more linear way than LSTM, being less affected by discrepancies, better learning the behavior of the bitcoin time series, Fig.6 shows a plot of Forecast versus Observed values at 2017-12-28. Other findings in the this analysis are related to the convergence and speed of LSTM and GRU, the last one is faster and usually requires smaller Training Sets to reach best forecasts.

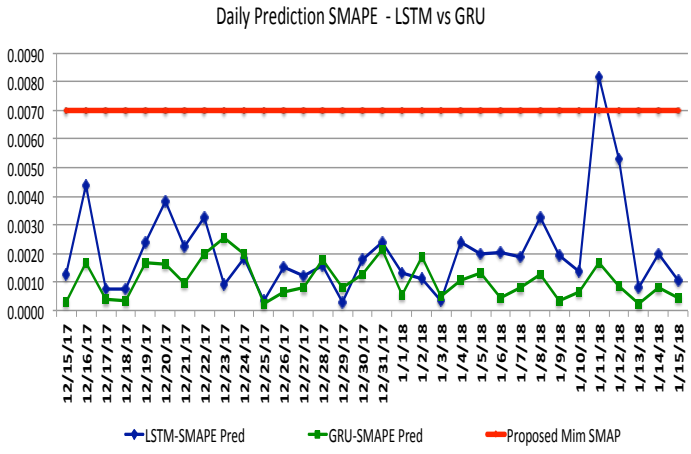


Fig. 5: Graphical Analysis of SMAP.


Fig. 6: Comparison of Predicted vs. Observed (GRU Worst Forecast)

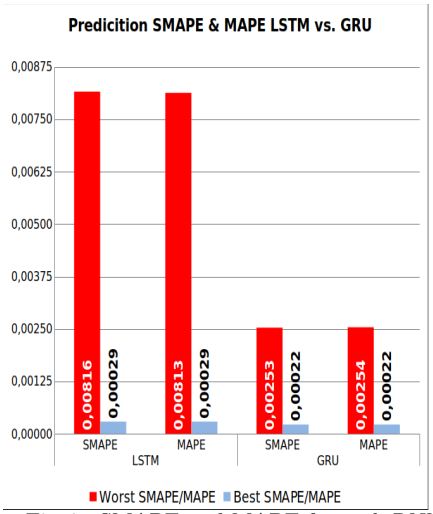


Fig. 7: SMAPE and MAPE for each RNN.

## 6. CONCLUSIONS

Our work proposes a methodology for constructing recurrent neural networks for predicting minute by minute bitcoin time series for an entire Desired Forecast Date ($DFD$), our methodology constructs 12 Recurrent Neural Networks (RNNs), varying the training and the test sets and selecting the model with smaller SMAPE to forecast $DFD$. Our results show that our work techniques turned possible to get forecasts SMAP smaller than 0.007 as proposed, reaching 0.0002 in best case, using GRU RNN. GRU has outperformed LSTM and shows to be more sensible in learn complex behavior expressed by the bitcoin Time Series. It looks important that the bitcoin Time Series is rich in details, our results suggest that this richness can be captured and used for better forecasts, even in complex bitcoin trading moves. This work shows it is possible guaranty exceptional forecasts using RNNs with our work methodology and be the core of a bitcoin forecast financial tool, also if compared to works found in literature review the results are very solid, since every day forecast has surpassed the desired SMAP with consistence.

## REFERENCES


Chauhan, A., Kumar, M., and Saini, V. K. Bitcoin financial forecasting. *Int. J. Advanced Research, Ideas and Innovations in Technology* 5 (2), 2019.

Cowpertwait, P. and Metcalfe, A. *Introductory Time Series with R*. Springer, 2009.

Gulli, A. and S., P. *Deep Learning with Keras*. Packt Publishing Ltd., 2017.

Mangala, N., Bhat, A., Avabratha, G., and N.Bhat. BITCOIN price prediction using machine learning. *Int. J. Information and Computing Science*, 2019.

Rathan, K., Sai, S. V., and Manikanta, T. S. Crypto-currency price prediction using decision tree and regression techniques. In *Proc. 3rd Int. Conf. on Trends in Electronics and Informatics (ICOEI 2019)*, 2019.

Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019, 2020. Available at `https://doi.org/10.1016/j.asoc.2020.106181`.

Shcherbakov, M., Brebels, A., Shcherbakova, N., Tyukov, A., Janovsky, T., and Kamaev, V. A survey of forecast error measures. *World Applied Sciences Journal* vol. 24, pp. 171–176, 2013.

Uras, N., Marchesi, L., Marchesi, M., and Tonelli, R. Forecasting bitcoin closing price series using linear regression and neural networks models, 2020. Preprint submitted to Elsevier, in evaluation process. Available at `https://arxiv.org/abs/2001.01127v1`.

Yamak, P. T., Yujian, L., and Gadosey, P. K. A comparison between arima, lstm, and gru for time series forecasting. In *Proc. 2nd International Conference on Algorithms*, 2019.