**Cryptocurrency price prediction using LSTMs**

**Predict Bitcoin price using LSTM Deep Neural Network in TensorFlow 2**

Project

by

Bhavesh patil

Shreeraj patil

Mrunali ghate

Poonam shinare

Janhavi Gosavi

YEAR OF STUDY

PROGRAM [Computer Science]

SINHGAD INSTITUTE OF TECHNOLOGY AND SCIENCE, NARHE

Submitted to the CSRBOX team

as the final project deliverable for

IBM SkillsBuild Data Analytics Innovation Camp

**Project Summary**

Cryptocurrencies are a digital way of money in which all transactions are held electronically. It is a soft currency which doesn’t exist in the form of hard notes physically. Here, we are emphasizing the difference of fiat currency which is decentralized that without any third-party intervention all virtual currency users can get the services. However, getting services of these cryptocurrencies impacts on international relations and trade, due to its high price volatility. There are several virtual currencies such as bitcoin, ripple, ethereum, ethereum classic, lite coin, etc. In our study,we especially focused on popular cryptocurrency,i.e.,bitcoin.Frommany types of virtual currencies, bitcoin has a great acceptance by different bodies such as investors, researchers, traders, and policy-makers. To the best of our knowledge, our target is to implement the efficient deep learning-based prediction models, specifically long short-term memory (LSTM) and recurrent neural network (RNN) to handle the price volatility of bitcoin and to obtain high accuracy. Our study involves comparing these two time series deep learning techniques and proved the efficacy in forecasting the price of bitcoin.

Bitcoin is a kind of Cryptocurrency and now is one of type of investment on the stock market.Stock markets are influenced by many risks of factor. And bitcoin is one kind of cryptocurrency that keep rising in recent few years, and sometimes sudden fall without knowing influence behind it on the stock market. Because it’s fluctuations, there’s a need and automation tool to predict bitcoin on the stock market. This research study learns how to create model prediction bitcoin stock market prediction using LSTM, LSTM (Long Short Term Memory) is another type of module provided for RNN later developed and popularized by many researchers, like RNN, the LSTM also consists of modules with recurrent consistency.

# TABLE OF CONTENTS

CHAPTER

1. INTRODUCTION 1

Topic Brief 2

Purpose of the Study 3

Hypotheses #

Significance of the Study #

Method of Procedure #

Collection of Data #

Treatment of the Data #

Data Source #

Definitions of Terms #

2. Conclusion and future work #

REFERENCES #

APPENDICES #

Chapter 1

1. **INTRODUCTION**

Bitcoin is a cryptocurrency that was created in January 2009. It is the world’s most valuable cryptocurrency and is traded on over 40 exchanges around the world, accepting over 30 different currencies. As a currency, Bitcoin offers a new opportunity for price forecasting as it has high volatility, which is much higher compared to traditional currencies.

The bitcoin system is a set of decentralized nodes that run the bitcoin code and store its blockchain. Metaphorically, a blockchain can be considered a collection of blocks. In each block, there is a collection of transactions. Because all the computers running the blockchain has the same list of blocks and transactions, and can transparently see these new blocks being filled with new bitcoin transactions, no one can cheat the system.

Recurrent neural networks (RNN) are the state-of-the-art algorithm for sequential data and are used by Apple’s Siri and Google’s voice search. It is an algorithm that remembers its input due to its internal memory, which makes the algorithm perfectly suited for solving machine learning problems involving sequential data. It is one of the algorithms that have great results in deep learning. In this article, it is discussed how to predict the price of Bitcoin by analyzing the information of the last 6 years. We implemented a simple model that helps us better understand how time series works using Python and RNNs.

**Topic Brief**

RNNs and LSTM are excellent technologies and have great architectures that can be used to analyze and predict time-series information.

### Recurrent Neural Networks

RNNs are a robust and powerful type of neural network and are considered one of the most professional algorithms because they are the only ones with internal memory.The algorithm performs very well for sequential data such as time series, speech, text, financial data, audio, video, weather, and more. RNNs are able to form a much deeper understanding of a sequence and its context compared to other algorithms.

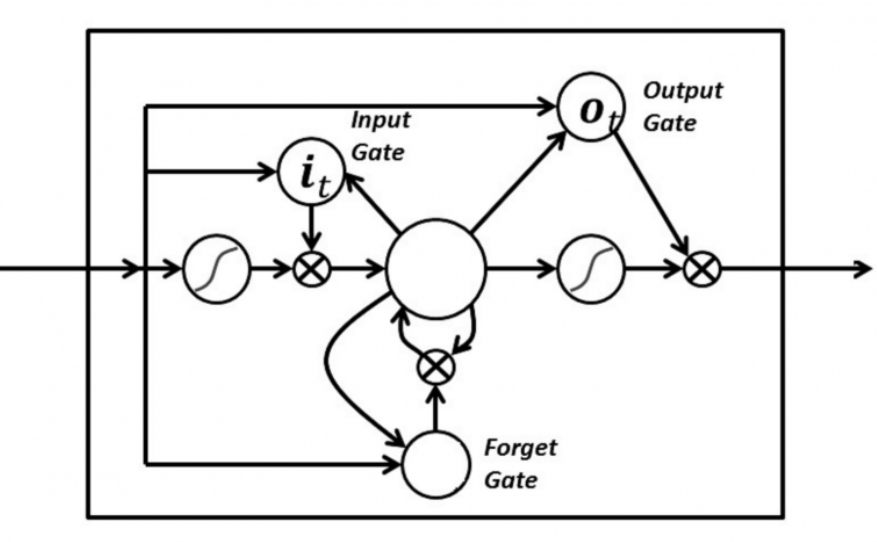
In an RNN, the information goes through a cycle. When making a decision, it considers the current input and also what it has learned from the inputs it has received previously.

### Long Short-Term Memory (LSTM)

### Long short-term memory networks are an extension of recurrent neural networks, which basically extend the memory. Therefore it is well suited to learn from important experiences that have very long time lags in between.

LSTMs enable RNNs to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory.

In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn’t important (forget gate), or let it impact the output at the current timestep (output gate).



The gates in an LSTM are analog in the form of sigmoids, meaning they range from zero to one. The fact that they are analog enables them to do backpropagation.

**Purpose of the Study**

The proposed study considers two different deep learning-based prediction models to predict price of bitcoin by identifying and evaluating relevant features by the model itself. After applying both the models for bitcoin prediction, we can determine which model is much more accurate for the future fulfillment of our target and select appropriate parameters to obtain a better performance. In this work, we have proposed deep learning mechanisms such as LSTM and RNN which are the latest and efficient techniques for the forecasting of bitcoin price. As bitcoin is the most popular cryptocurrency, the price volatility issue should be handled within a short period of time.

**Hypotheses**

There is a lot of uncertainty in the crypto markets right now, thus there are many conflicting predictions about the price of bitcoin. However, it is worth distinguishing between them depending on the time frame. In the short term, uncertainty reigns supreme. Right now there is no convergence at all between the various predictions regarding bitcoin price movements in the coming days. To tell the truth, many are bearish, so there is a lot of speculation about a new descent below $30,000, or even lower.On individual exchanges, it is possible that the price on the 22nd was even lower, but taking CoinMarketCap’s historical data, which averages across different exchanges, the picture is more general. Among those who argue that bitcoin’s price could fall back below the psychological $30,000 mark in the coming days, many say it could even return to the June 22 low, or make new annual lows at $28,000.

**Significance of the Study**

1. **Analyze a Dataset:-** Dataset includes daily level information on the number of Price of Bitcoin Price.It has 8 columns which are as follows:-

* DATE
* OPEN
* HIGH
* LOW
* CLOSE
* VOLUME

**2.** **Prediction and Estimation:-** For ML operations on any dataset we need to clean and process data.

* · Clean Data:No missing value is there in BTC\_USD.csv Dataset.
* · Select Feature:Last three features are selected for prediction.

**Method of Procedure**

**Collection of Data**

Initially started with knowledge of the Bitcoin market which was obtainable publicly on Kaggle two. The Dataset contains historical knowledge of Bitcoin from Dec 1st, 2014 to Jan 8th, 2018 separated into intervals of one minute. The information includes the details of the difference value, the closing value, the best value, the low always value, and the degree and weighted value of each time stamp. The Dataset was labeled as "True" if the value went up at the top of the one-minute timestamp and was labeled as "False" if it remained or attenuated at a constant position.

We have taken time series data ,Temporal datasets are quite common in practice. Your energy consumption and expenditure (calories in, calories out), weather changes, stock market, analytics gathered from the users for your product/app and even your heart produce *Time Series*.

Our dataset comes from Yahoo! Finance and covers all available data on Bitcoin-USD price.

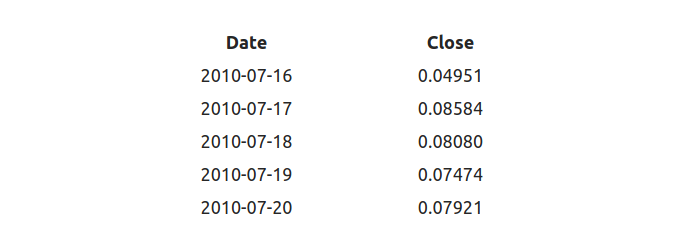
and we have loaded it into a Pandas dataframe.

csv\_path="https://raw.githubusercontent.com/Bp970/DataSet-Of-BTC\_USD/main/BTC-USD.csv"

df = pd.read\_csv(csv\_path, parse\_dates=['Date'])

df = df.sort\_values('Date')

we sort the data by Date just in case. Here is a sample of the data we’re interested in:



We have a total of *3201* data points representing the Bitcoin-USD price for *3201* days (~9 years). We’re interested in predicting the closing price for future dates.

Modeling

**Modeling**

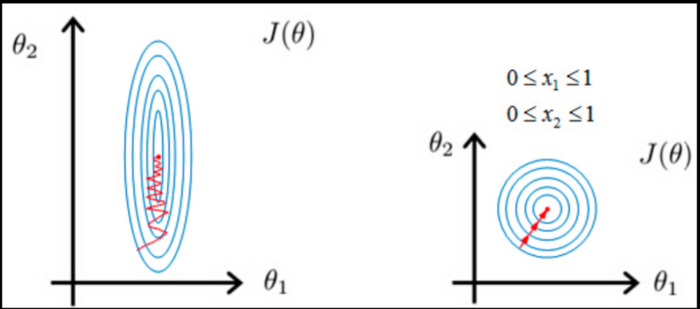
All models do not allow for operating on sequence data. Fortunately, we can use a special class of Neural Network models known as Recurrent Neural Networks (RNNs) just for this purpose. *RNNs* allow using the output from the model as a new input for the same model. The process can be repeated indefinitely.

One serious limitation of *RNNs* is the inability of capturing long-term dependencies in a sequence. One way to handle the situation is by using an **Long short-term memory (LSTM)** variant of *RNN*.

The default LSTM behavior is remembering information for prolonged periods of time. Let’s see how you can use LSTM in Keras.

## Data preprocessing

First, we’re going to squish our price data in the range [0, 1]. Recall that this will help our optimization algorithm converge faster.



We have used the MinMaxScaler from scikit learn.

scaler = MinMaxScaler()

close\_price = df.Close.values.reshape(-1, 1)

scaled\_close = scaler.fit\_transform(close\_price)

The scaler expects the data to be shaped as (x, y), so we add a dummy dimension using reshape before applying it.

We have removed NaNs since our model won’t be able to handle them well.

scaled\_close = scaled\_close[~np.isnan(scaled\_close)]

scaled\_close = scaled\_close.reshape(-1, 1)

We use isnan as a mask to filter out NaN values. Again we reshape the data after removing the NaNs.

**Making sequences**

LSTMs expect the data to be in 3 dimensions. We need to split the data into sequences of some preset length. The shape we want to obtain is:

[batch\_size, sequence\_length, n\_features]

SEQ\_LEN = 100

def to\_sequences(data, seq\_len):d = []

for index in range(len(data) - seq\_len):d.append(data[index: index + seq\_len])

return np.array(d)

def preprocess(data\_raw, seq\_len, train\_split):data = to\_sequences(data\_raw, seq\_len)

num\_train = int(train\_split \* data.shape[0])

X\_train = data[:num\_train, :-1, :]

y\_train = data[:num\_train, -1, :]

X\_test = data[num\_train:, :-1, :]

y\_test = data[num\_train:, -1, :]

return X\_train, y\_train, X\_test, y\_test

X\_train, y\_train, X\_test, y\_test =\preprocess(scaled\_close, SEQ\_LEN, train\_split = 0.95)

The process of building sequences works by creating a sequence of a specified length at position 0. Then we shift one position to the right (e.g. 1) and create another sequence. The process is repeated until all possible positions are used.

We save 5% of the data for testing. The datasets look like this:

(2945, 99, 1)

(156, 99, 1)

We’re creating a 3 layer [LSTM](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/LSTM) Recurrent Neural Network. We have used Dropout with a rate of 20% to combat overfitting during training:

|  | DROPOUT = 0.2  WINDOW\_SIZE = SEQ\_LEN - 1 |
| --- | --- |
|  | model = keras.Sequential() |
|  | model.add(Bidirectional( |
|  | CuDNNLSTM(WINDOW\_SIZE, return\_sequences=True), |
|  | input\_shape=(WINDOW\_SIZE, X\_train.shape[-1]) |
|  | ))  model.add(Dropout(rate=DROPOUT)) |
|  | model.add(Bidirectional( |
|  | CuDNNLSTM((WINDOW\_SIZE \* 2), return\_sequences=True) |
|  | )) |
|  | model.add(Dropout(rate=DROPOUT)) |
|  | CuDNNLSTM(WINDOW\_SIZE, return\_sequences=False) |
|  | )) |
|  | model.add(Dense(units=1)) Training We’ll use Mean Squared Error as a loss function and Adam optimizer.  BATCH\_SIZE = 64  model.compile(  loss='mean\_squared\_error',  optimizer='adam'  )  history = model.fit(  X\_train,  y\_train,  epochs=50,  batch\_size=BATCH\_SIZE,  shuffle=False,  validation\_split=0.1  ) **Predicting Bitcoin price** y\_hat = model.predict(X\_test)  We can use our scaler to invert the transformation we did so the prices are no longer scaled in the [0, 1] range.  y\_test\_inverse = scaler.inverse\_transform(y\_test)  y\_hat\_inverse = scaler.inverse\_transform(y\_hat) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Data Source**

* https://raw.githubusercontent.com/Bp970/DataSet-Of-BTC\_USD/main/BTC-USD.csv
* Yahoo! finance.

**Definitions of Terms**

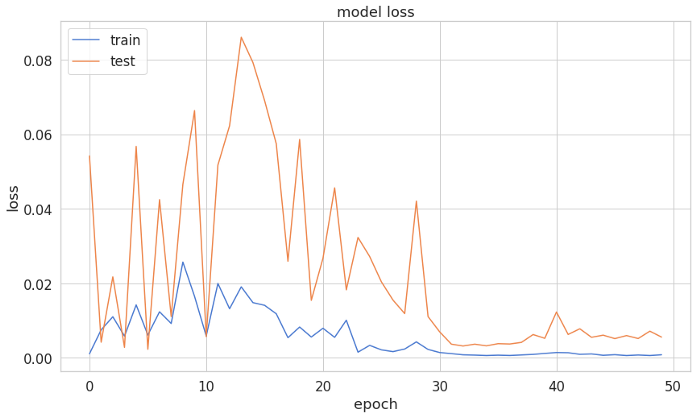
**Bidirectional RNNs** allows you to train on the sequence data in forward and backward (reversed) direction. In practice, this approach works well with LSTMs.

**CuDNNLSTM** is a “Fast LSTM implementation backed by cuDNN”. Personally, I think it is a good example of leaky abstraction, but it is crazy fast!

Our output layer has a single neuron (predicted Bitcoin price). We use Linear activation function which activation is proportional to the input

we do not want to shuffle the training data since we have used Time Series..

After a lightning-fast training (thanks Google for the free T4 GPUs), we have the following training loss:



Chapter 2

**CONCLUSION**

Since Bitcoin and the Blockchain technology were introduced in 2008, it has taken a pre-

dominant place in the cryptocurrency field. There are millions of users around the world,

especially in the United States. Predicting the price of cryptocurrencies has been a popular

topic, from which we can take advantage of the arbitrage for an investment

Cryptocurrency price prediction using LSTMs by utilizing the recurrent neural network. The model is developed using LSTM and achieves higher precision and recall than the traditional auto regressive approach. This Cryptocurrency price prediction using LSTMs can be used to predict the price fluctuation of the cryptocurrency and integrated to an autonomous trading system to assist the buying or selling of digital assets.

Our proposed model has succeeded to provide the result prediction of bitcoin from yahoo finance stock market. Our model with time series techniques can produce the results and the results can predict the price for the next days with split the data to train and test that we mention in the article above.

Future research will focus on modified LSTM layers, adding dropout and modified number epochs, and using different instability dataset to test how good the prediction results or try to use sentiment analysis combined with LSTM method to see the impact of the uncertainty in value bitcoin.

REFERENCES

* Dataset="https://raw.githubusercontent.com/curiousily/Deep-Learning-For-Hackers/master/data/3.stock-prediction/BTC-USD.csv"
* Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. Journal of Computational Science 2, 1–8 (2011)
* Box, G., Jenkins, G., Reinsel, G., Ljung, G.: Time series analysis : forecasting and control. In: ISBN 1-118-67502-9. OCLC 915507780 (2016)
* Chen, R., Lazer, M.: Sentiment analysis of twitter feeds for the prediction of stock market movement. Stanford Computer Science 229
* Hochreiter, S., Schmidhuber, J.: Lstm can solve hard long time lag problems. In: Proceedings of the 9th International Conference on Neural Information Processing Systems. pp. 473–479
* Hutto, C.J., Gilbert, E.: Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International AAAI Conference on Weblogs and Social Media. pp. 216–255
* Joshi, M., Das, D., Gimpel, K., Smith, ., a. N.: Movie reviews and revenues: An experiment in text regression. human language technologies. In: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 293–296
* Pimprikar, R., Ramachadran, S., Senthilkumar, K.: Use of machine learning algorithms and twitter sentiment analysis for stock market prediction. International Journal of Pure and Applied Mathematics pp. 521–526 (2017)
* Wang, H., Can, D., Kazemzadeh, A., Bar, F., Narayanan, S.: Hand gesture recognition using fourier descriptors. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. pp. 115–1

**APPENDICES**

pip install tensorflow

import os

import numpy as np

import tensorflow as tf

from tensorflow import keras

import pandas as pd

import seaborn as sns

from pylab import rcParams

import matplotlib.pyplot as plt

from matplotlib import rc

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.layers import Bidirectional, Dropout, Activation, Dense, LSTM

from tensorflow.python.keras.layers import CuDNNLSTM

from tensorflow.keras.models import Sequential

%matplotlib inline

sns.set(style='whitegrid', palette='muted', font\_scale=1.5)

rcParams['figure.figsize'] = 14, 8

RANDOM\_SEED = 42

np.random.seed(RANDOM\_SEED)

csv\_path = "<https://raw.githubusercontent.com/curiousily/Deep-Learning-For-Hackers/master/data/3.stock-prediction/BTC-USD.csv>"

df = pd.read\_csv(csv\_path, parse\_dates=['Date'])

df = df.sort\_values('Date')

df.head()



df.shape

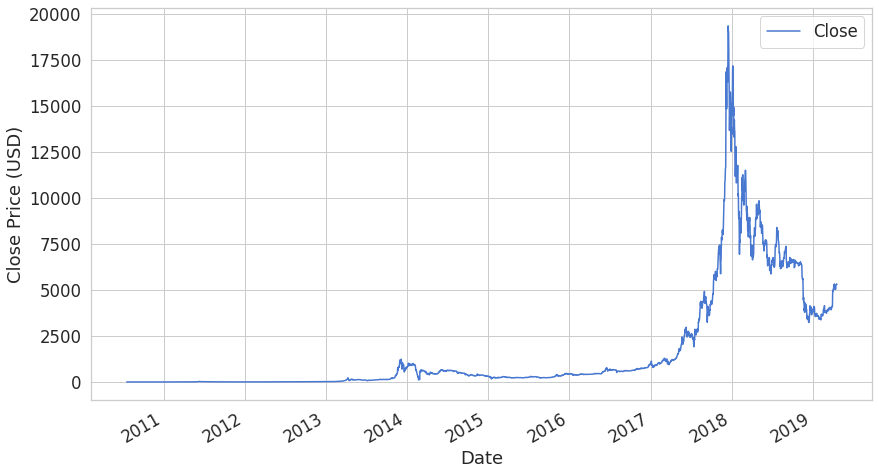
**(3201, 7)**

ax = df.plot(x='Date', y='Close');

ax.set\_xlabel("Date")

ax.set\_ylabel("Close Price (USD)")

Text(0, 0.5, 'Close Price (USD)')



# Normalization

scaler = MinMaxScaler()

close\_price = df.Close.values.reshape(-1, 1)

scaled\_close = scaler.fit\_transform(close\_price)

scaled\_close.shape

**(3201, 1)**

np.isnan(scaled\_close).any()

**False**

scaled\_close = scaled\_close[~np.isnan(scaled\_close)]

scaled\_close = scaled\_close.reshape(-1, 1)

np.isnan(scaled\_close).any()

**False**

Preprocessing

SEQ\_LEN = 100

def to\_sequences(data, seq\_len):

d = []

for index in range(len(data) - seq\_len):

d.append(data[index: index + seq\_len])

return np.array(d)

def preprocess(data\_raw, seq\_len, train\_split)

data = to\_sequences(data\_raw, seq\_len)

num\_train = int(train\_split \* data.shape[0])

X\_train = data[:num\_train, :-1, :]

y\_train = data[:num\_train, -1, :]

X\_test = data[num\_train:, :-1, :]

y\_test = data[num\_train:, -1, :]

return X\_train, y\_train, X\_test, y\_test

X\_train, y\_train, X\_test, y\_test = preprocess(scaled\_close, SEQ\_LEN, train\_split = 0.95)

X\_train.shape

**(2945, 99, 1)**

X\_test.shape

**(156, 99, 1)**

**Model**

DROPOUT = 0.2

WINDOW\_SIZE = SEQ\_LEN - 1

model = keras.Sequential()

model.add(Bidirectional(CuDNNLSTM(WINDOW\_SIZE, return\_sequences=True),

input\_shape=(WINDOW\_SIZE, X\_train.shape[-1])))

model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(CuDNNLSTM((WINDOW\_SIZE \* 2), return\_sequences=True)))

model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(CuDNNLSTM(WINDOW\_SIZE, return\_sequences=False)))

model.add(Dense(units=1))

model.add(Activation('linear'))

**Training**

model.compile(

loss='mean\_squared\_error',

optimizer='adam'

)

BATCH\_SIZE = 64

history = model.fit(

X\_train,

y\_train,

epochs=50,

batch\_size=BATCH\_SIZE,

shuffle=False,

validation\_split=0.1

)

**Epoch 1/50**

**42/42 [==============================] - 20s 114ms/step - loss: 0.0052 - val\_loss: 0.0432**

**Epoch 2/50**

**42/42 [==============================] - 3s 62ms/step - loss: 0.0154 - val\_loss: 0.0036**

**………………………..**

**Epoch 49/50**

**42/42 [==============================] - 2s 46ms/step - loss: 0.0013 - val\_loss: 0.0030**

**Epoch 50/50**

**42/42 [==============================] - 2s 45ms/step - loss: 0.0012 - val\_loss: 0.008**

model.evaluate(X\_test, y\_test)

**5/5 [==============================] - 0s 33ms/step - loss: 0.0029**

**0.002934458199888468**

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

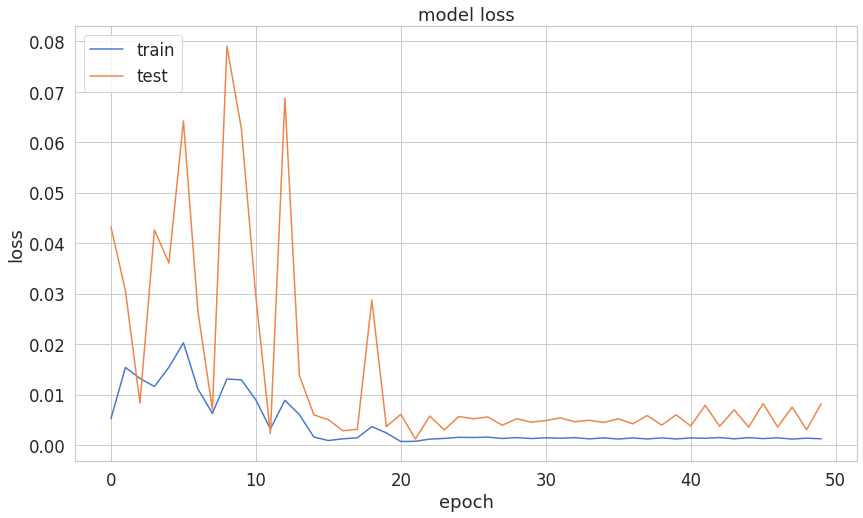
plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()



# Prediction

y\_hat = model.predict(X\_test)

y\_test\_inverse = scaler.inverse\_transform(y\_test)

y\_hat\_inverse = scaler.inverse\_transform(y\_hat)

plt.plot(y\_test\_inverse, label="Actual Price", color='green')

plt.plot(y\_hat\_inverse, label="Predicted Price", color='red')

plt.title('Bitcoin price prediction')

plt.xlabel('Time [days]')

plt.ylabel('Price')

plt.legend(loc='best')

plt.show();

