Cryptocurrencies Price Prediction Using Deep Learning Techniques

Pooya Danandeh

Pooya144@gmail.com

Department of Computer Science, University of Tabriz, Tabriz, Iran

Abstract

The cryptocurrency market is known for being volatile and dynamic so predicting how it will perform is one of the most difficult things to do. There are so many macro and micro factors involved in the prediction, such as politics, news, global economic conditions, unexpected events, traders' psychological factors, a famous company's financial performance, and so on. All these aspects combine to make cryptocurrencies prices volatile and very difficult to predict with a high degree of accuracy. In this paper we try to use deep learning techniques as a game-changer in this field.

Keywords: Cryptocurrency market, Price Prediction, Deep Learning

1 Introduction

Basically, cryptocurrency market and all other stock market analysis divided into 2 parts: technical analysis and fundamental analysis. Fundamental analysis involves news around a stock or coin and technical analysis is prediction based on historical market data. In most cases we cannot predict on prices with fundamental analysis because it is complex and usually occur suddenly. So, we have to focus on technical analysis. In technical analysis we face time series patterns. So, we have to use LSTM layers. In continuation this paper we focus to predict Bitcoin price data in Binance exchange, with technical analysis and price action approach.

2 Implementation

2.1 Needed Tools

To implement a network to prediction on prices, we need some tools and here will import them. We import pandas and numpy for managing data, matplotlib for visualizing, ssl verify

data source ssl layer to load update data, MinMaxScaler from scikit-learn for normalization and Sequential, Dense, Dropout and LSTM from Keras to create our model.

```
import pandas as pd
import numpy as np
import ssl
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
```

2.2 Load and Preprocessing Data

In this section, we load Bitcoin up to date price dataset in daily timeframe contains highest price, lowest price, open price, close price and market trading volume in a particular day from August 7, 2017. We load this dataset at every run from here.

```
symbol = 'BTCUSDT'
filepath = f"https://www.cryptodatadownload.com/cdd/Binance_{symbol}_d.csv"
ssl._create_default_https_context = ssl._create_unverified_context

df = pd.read_csv(filepath, skiprows=1)
df.head()
```

	unix	date	symbol	open	high	low	close	Volume BTC	Volume USDT	tradecount
0	1.640995e+12	2022-01-01 00:00:00	BTC/USDT	46216.93	46527.26	46208.37	46332.51	386.43504	1.791172e+07	11081.0
1	1.640909e+12	2021-12-31 00:00:00	BTC/USDT	47120.88	48548.26	45678.00	46216.93	34937.99796	1.650716e+09	1059783.0
2	1.640822e+12	2021-12-30 00:00:00	BTC/USDT	46464.66	47900.00	45900.00	47120.87	30352.29569	1.428756e+09	910157.0
3	1.640736e+12	2021-12-29 00:00:00	BTC/USDT	47543.74	48139.08	46096.99	46464.66	39498.87000	1.873786e+09	1114074.0
4	1.640650e+12	2021-12-28 00:00:00	BTC/USDT	50701.44	50704.05	47313.01	47543.74	45853.33924	2.242102e+09	1345774.0

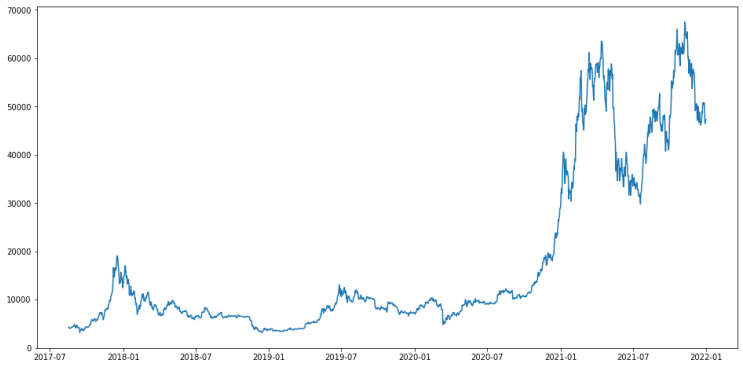
Due to using price action approach in our model we need just open and close prices. Before that, we need to set date column as index and sort them in order time and also plotting the data.

```
#setting index as date
df['date'] = pd.to_datetime(df.date,format='%Y-%m-%d')
df.index = df['date']

#plot
plt.figure(figsize=(16,8))
plt.plot(df['close'], label='Close Price history')
```

```
#creating dataframe
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df))
                        ,columns=['date', 'close'])
for i in range(0,len(data)):
    new_data['date'][i] = data['date'][i]
    new_data['close'][i] = data['close'][i]

#setting index
new_data.index = new_data.date
new_data.drop('date', axis=1, inplace=True)
```



After these processes, we will need to split training and test data. Because of this we consider data for 1000 days as training data and remains (almost 600 days) for test data. Then we should normalize them using MinMaxScaler function.

```
#creating train and test sets
dataset = new_data.values

train = dataset[0:1000,:]
valid = dataset[1000:,:]

#converting dataset into x_train and y_train
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

Due to using LSTM layers, our data should be in a timespan format. So, we create our training dataset in 60 days timespan for every entry. The shape of an entry will be 60×1 (close price of last 60 days)

```
x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])

x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

2.3 Design the Model

Due to working with timespans our best choice is LSTM layers, we simply define a sequential architecture and put 2 LSTM layer and also a Dense layer to predict the 61th day close price. As mentioned above, input shape will be 60×1.

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,
               input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
_________________________________________________________________
```

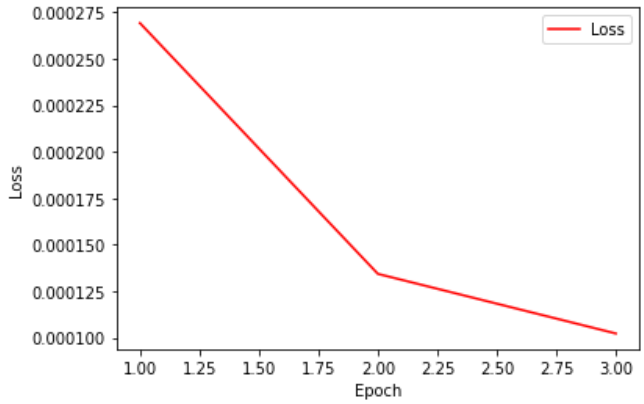
2.4 Compile and Training the Model

We will compile the model with Mean Square Error (mse) loss function and Adam optimizer. Also, we consider 3 epochs for training process with batch size of 1.

```
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(x_train, y_train, epochs=3, batch_size=1, verbose=1)
```

```
Epoch 1/3
940/940 [==============================] - 36s 36ms/step - loss: 3.2114e-04
Epoch 2/3
940/940 [==============================] - 27s 29ms/step - loss: 1.4976e-04
Epoch 3/3
940/940 [==============================] - 27s 29ms/step - loss: 1.1681e-04
```

And the training result:



2.5 Evaluate the Network

Our test dataset is almost last 600 days in BTC/USDT chart. We will make prediction on it. But first we should reshape test data similar to training data i.e. 60 days timespan for every input. Also, our input shape is 60×1.

```
inputs = new_data[len(new_data) - len(valid) - 60:].values
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)

X_test = []
for i in range(60,inputs.shape[0]):
    X_test.append(inputs[i-60:i,0])
X_test = np.array(X_test)

X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))

closing_price = model.predict(X_test)
closing_price = scaler.inverse_transform(closing_price)
```

And result of predictions on chart:

Fantastic predictions!

3 Conclusion

Deep learning showed us can be game-changer in field of prediction on cryptocurrency market and stock market prices. Before this paper we tried to using machine learning algorithm such as linear regression and KNN regressor, but both of them unable to predict with low error. With a simple LSTM model, we can predict on BTC/USDT price. To learn and predict on other cryptocurrencies prices you can change the "Symbol" variable.

Note: Please don't take this as financial advice or use it to make any trades of your own.

4 References

[1] Predicting Stock Price Machine Learning and Deep Learning Techniques

https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques/

[2] Machine Learning to Predict Stock Prices

https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233

[3] Predicting Stock Prices Using Machine Learning

https://neptune.ai/blog/predicting-stock-prices-using-machine-learning