MFE 230T2
Deep Learning for Cryptocurrency Prediction

Sravya Divi
Prakarsh Duhoon
Alexandre Goddard
Xiaoyan Peng
Yiyun (Yvonne) Zhu

Abstract

Cryptocurrency has emerged as one of the key digital currencies due to its global reach by design and the lower costs of distributed ledgers. Digital Currencies have seen dramatic price gains in 2019 and accurate predictions of these price movements can assist towards right investing decisions and potential increased profits. Prediction of Cryptocurrency price movement is very complex because of non-stationary behaviour of prices and non-linear patterns.

In this project we evaluated the widely used deep learning models to predict the next hour Bitcoin price movement, up or down by using historical price data as the base model. The model is enhanced by adding additional features to the base model like VIX, Market Features(Mining Difficulty, Hash Rate). Deep learning algorithms are considered to be the most powerful and the most effective methods in approximating extremely complex and non-linear classification and regression problems, therefore it was expected that a noticeable performance increase will be achieved by the incorporation of these models compared to classic machine learning algorithms. Surprisingly, our results demonstrated that the utilized DL algorithms (LSTM, CNN, LSTM+CNN), slightly outperformed the other ML algorithms utilized in our experiments, whereas instead a noticeable performance increase was anticipated.

In order to predict the future movement of predictions in a more reliable way thresholds are embedded into the model to evaluate for the data points only when the model is confident. This resulted in significant increase in accuracy among all levels of model improvements and has a significant impact to reduce transaction costs.

Introduction

Differ from traditional asset classes, Cryptocurrency is a digital asset and decentralized network which utilizes blockchain and strong cryptography to secure financial transactions. Bitcoin, which is the first, the most popular and the most valuable cryptocurrency, was launched in 2009. Before 2017, the market capitalization of cryptocurrencies was $17.7B, then it soured to $566.26B at the end of 2017 and decreased sharply to $128.78B in 2018. After 2019, the market capitalization continuously increases to $331.76B during COVID-19 period in 2020 because collapsing interest rates by central banks increased the appeal of digital assets. Until 2020 September, Bitcoin (BTC), Ethereum (ETH) and Tether (USDT) are three most liquid cryptocurrencies in the market and represent 56.7%, 11.8%, 4.26% of global cryptocurrency market capitalization, respectively.

The huge interests and trade transactions in Cryptocurrency resulted in the significant price fluctuation, especially in 2017 and 2018. The volatility of the price of Bitcoin was around 8% from October of 2017 to January of 2018. With difficulty in determining the main factors of volatility, the high volatility in Cryptocurrency creates the uncertainty for investors and government regulators. Due to the lack of Indexes, it is harder to correctly predict the price pattern for Cryptocurrency than that of traditional asset classes. Thus, the motivation of our project is to build a model to predict the price movement of Cryptocurrency for better investment.

Differ than other classification problems, Cryptocurrency price movement prediction can be considered as a time series problem and no consistent patterns in the data which allows us to model prices over time. Compared with traditional time series model ARIMA, Deep Learning algorithm, which aims to capture non-linear patterns by learning the history of the sequence of data, could be the better choice for time series data forecast with lags of unknown size and duration.

In this project, we attempt to build a model to predict the next hour Bitcoin price movement, either up or down, by using historical price data, technical factors, economic factors and Deep Learning algorithms such as CNN and LSTM models.

The reminder of this report is organized as follows: Sect.2 Literature Review for recent work in price prediction of Cryptocurrency using Deep Learning models. Section 3 presents the data collection and data processing. Section 4 discusses architecture and training results of both baseline model (SVM) and advanced Model (CNN, LSTM), including hyperparameter tuning, Rolling Horizon, and PnL Analysis. Finally, Section 5 shows our conclusion and further recommendations.

Literature Review

Recently, there are lots of efforts put into deep learning techniques for predicting Cryptocurrency prices. As mentioned in Emmanuel and Loannis's paper, Shintate and Pichl, who studies a trend prediction classification model for Cryptocurrency price, showed that

LSTM was superior to the general regression architecture but the PnL analysis suggested that simply buy-and-hold strategy beats their model.

Emmanuel and Loannis evaluated the performance of advanced Deep Learning algorithm, including LSTM, CNN, Bi-LSTM and their combinations, in predicting Cryptocurrency price movement of the BTC, ETH and XRP, using hourly price data from Jan-2018 to Aug-2019 from Kraken.com website. After compared results with Machine Learning benchmark models, which are SVR, 3NN and DTR, they got the following conclusions:

1. Using F1-score and Accuracy as performance evaluation metric, although the performance variations are minimal, CNN-LSTM and CNN-BiLSTM model exhibited the slightly better overall performance among all prediction models
2. DL models did not significantly outperform as expected because the price process is very close to Random Walk, which is proved by Stavroyiannis, one of the co-authors of this paper.
3. Alternative validation metric needs to be considered, because real-life trading prefers accurate predictions on random times rather than unreliable predictions on every moment.

Data Preprocessing

Raw Data

For this project, we explored lots of potential factors and divided them into 4 categories, including Bitcoin price related, Market related, Currency and Index, and Other. Currencies time series have been filled with the close value of the last opened hour. Same for commodities and indices.

Category	Frequency	Contents	Source
Bitcoin price	hourly	Open, Low, High, Close, Number of Trades	Bitstamp, Binance
Market related	daily	Mining difficulty, hash rate, etc.	Bitcoinity
Currency & Index	hourly	8 major currency pairs and 9 major indexes	histdata.com
Others	hourly/daily	VIX, GOLD, Brent, WTI etc	histdata.com

Because the time zones for download time series are different, we reconcile them into EST.

The detailed information is presented in the following table. All data ranges from Jan-2017 to Jan-2019. We utilized the rolling horizon to decide the proper cutting-edge training and testing set. The detailed illustration is in Section 4.

Feature Engineering

We utilized the open price of Bitcoin, rather than Close price for two reasons. Firstly, we use open price to be consistent with the time frame of other features whose time frame is the beginning of the hour. Also, using closing price may result in prediction with future information or look-ahead bias. So, using hourly open price series, we stand at the beginning of the hour and predict the price movement until the end of this hour or the very beginning of the next hour.

Then, we test for unit roots using ADF test and ACF plots.

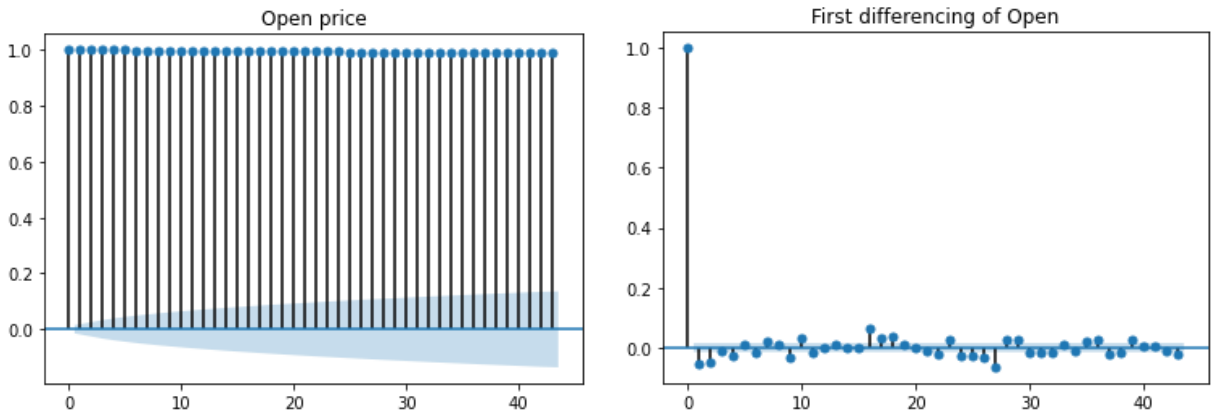


Figure 1: ACF plot for open price series (Left), and ACF plot for First difference of open price (Right)

Name	**ADF Statistics**	**P value**
BTC Open price	-1.605	0.481 (fail reject)
BTC Open price first difference	-19.871	0.000 (reject)

The plots above show that price has high autocorrelation and are non-stationary, while the first differencing of close price has no sign of autocorrelation and seasonality. The ADF test shows a similar result. While the open price series failed to reject the null hypothesis that there is a unit-root, the first difference of open price rejects the null. Thus, we take first differencing of all time series data, including open price, as input.

Then, we derived several features, which is similar to common factors in high frequency strading, from raw open price series.

1. Hourly Realized Volatility: use minute BTC data to calculate hourly realized volatility
2. Flag: determine whether current open price is a threshold, the local max/min. To avoid the look-ahead problem, we shifted the true flag one hour.
3. Max drawdown: calculate the difference between current open price and previous threshold
4. Two Momentum indicators: Relative Strength Index and Stochastic Indicator.
5. Bollinger Bands: We have computed the upper and lower band and then feed the model with the differences between the current price of the BTC and these two bands.

Finally, we created the labels for price up (+1) and down (-1) movement using open price series. The target label is quite balanced, with 51.29% for up label and 48.71% for down label.

Data analysis

We applied normal EDA procedures such as missing values, outlier detection, distribution and correlations. There are 111 missing values in total in our prior dataset using Binance API, which is an extremely small portion. Most of the missing values are not consecutive, except for 2018-02-08 to 2018-02-10. We tried Linear interpolation and simple average prediction using past 5 hours open prices. However, both of them have bad performance for consecutive missing values. Fortunately, we search Binance API and other source Bitstamp to backfill these missing values.

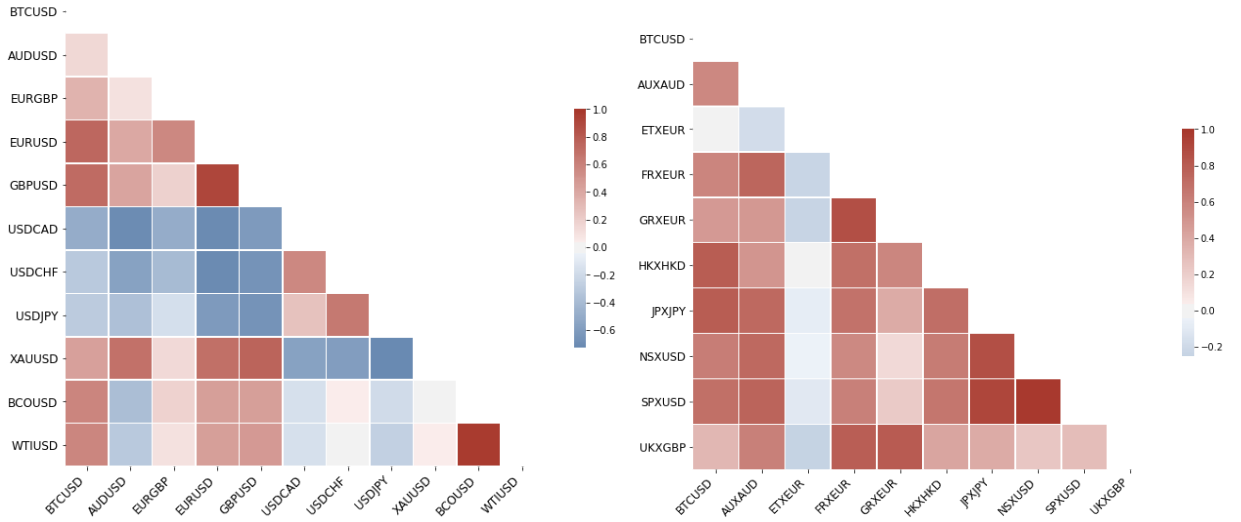


Figure 2: Correlation plot for Commodities and BTC (Left), and for Indices and BTC (Right)

We explored the correlation between Bitcoin price and other external features. The above correlation plots suggest that Bitcoin price is inversely correlated USD and positively correlated with GOLD and OIL. And for Indices, Bitcoin price is positively correlated with most of the indices, ranging from 0 to 0.4. Some indices are highly correlated with each other, we take good care of them to avoid the collinearity in the prediction model.

Architecture

Baseline - SVM

We begin by starting off with the author suggested SVM architecture, later on improve it by adding features like VIX, Commodities, hash rate, mining difficulty.

The kernel used for SVM Model is rbf(radial basis function) and the hyperparameter used to tune the SVM is C=10 and gamma=0.001

- `{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}`
- `SVC(C=10, gamma=0.0001)`

The impact on accuracy score on the base model (using only the prices) and the enhanced model after adding features increased from 33.24% to 49.61%

DL Models

We begin by starting off with the author suggested CNN-LSTM model architecture, later on improve it and also investigate CNN and LSTM models separately.

Our initial author based CNN-LSTM model consists of the following Sequential model:

- 2 Convolution layers with 32 and 64 units with ‘RELU’ activation & ‘Valid’ padding
- Flatten layer to transition the information flow from CNN to LSTM layer
- LSTM layer with 50 units and RELU activation function
- Final Dense layer
- Loss function: Mean Squared Error (MSE) was changed to Mean Absolute Error (MAE)
- Optimizer: Adam

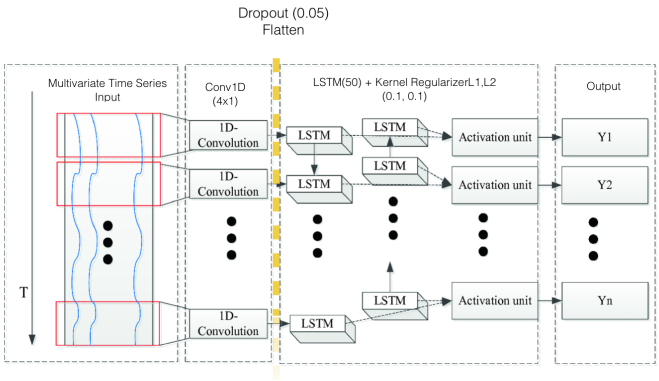


Figure 4: CNN-LSTM Architecture

This architecture was used by the author to predict the price change (Regression); the authors also used the output for detecting price movement (Classification).

Changing the loss function from MSE to MAE helped in significant improvement in the training process and overall performance of the model. This was possible because MAE results in larger absolute values in vicinity of (-1,1); therefore gives the model a better feedback during the training.

Our first step improved CNN_LSTM model has the following architecture:

- Convolution layer with 32 units with 'RELU' activation & 'Valid' padding
- Dropout layer
- Convolution layer with 64units with 'RELU' activation & 'Valid' padding
- Dropout layer
- Flatten layer to transition the information flow from CNN to LSTM layer
- LSTM layer with 50 units and RELU activation function and elastic net regularization
- Final Dense layer with tanh activation function
- Loss function: Mean Absolute Error
- Optimizer: Adam

We added the dropout layers after CNN layers and added the elastic net to the LSTM layer to prevent overfitting to the extent possible.
Besides the CNN-LSTM model, we also attempt using only one type of layers to test the best performance contributor. For the CNN models, most of the architectural details remain the same except for the removal of the LSTM layer. Similarly, for the LSTM model, we feed the data directly to the LSTM layer and subsequent segment of the LSTM model.

More Layers & Hyperparameter Tuning

As discussed in Sutskever et al (2014) and Irsoy and Cardie (2014), adding more LSTM layers in the RNN architecture helps empirical performance. Therefore, we attempted the following modification to the structure (blue is the new layer):
LSTM: LSTM(64) + LSTM(32) + Dense
CNN_LSTM: Conv1D + Dropout + Conv1D +LSTM(64) + LSTM(32) + Dense
There is minor improvement on accuracy from 0.539 to 0.552. Next, we performed hyperparameter tuning.
Bayesian hyperparameter tuning is used in the experiment to tune with the objective of accuracy. The procedure focused on the CNN-LSTM architecture, particularly on the first convolutional layer, final layer, and learning rate.
Below is a summary of our hyperparameter search space:

Layer	Hyperparameter Name	Search Value	Optimal Value
Conv1D	num_filters	32, 64	64
Conv1D	activation function	relu, tanh, sigmoid	relu
Dense	activation function	tanh, softmax	tanh
Compile	learning_rate	1e-2, 1e-3, 1e-4	1e-03

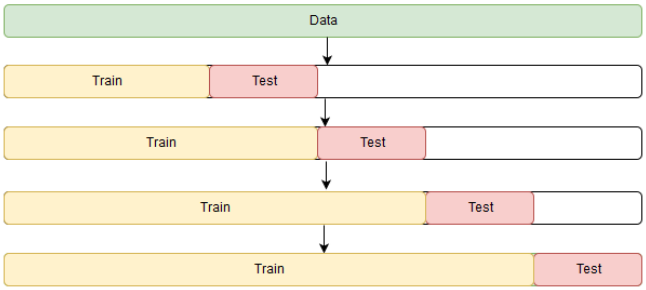
The rationale behind the Bayesian method is that it uses a surrogate probability model of the objective function to find the true set of optimal values with the least amount of calling on the actual objective function. Indeed, the tuning process takes about 87 seconds on average.

Rolling Horizon

In order to test the robustness of the architecture and validate the data quality, we used a forward training approach (fig to expand the horizon from taking just the first half of the dates (i.e. 2017-2018) and gradually increasing by 10% on the training data with more 2018 days to reach to a 90/10% train/test split in CNN-LSTM framework. An universal 10% validation set is reserved from the training split, so for example, an 80/20% split will include 70% training, 10% validation, and 30% test data in a chronological order.
We can observe that the model learned quickly and did not need all the data to reach a plateau level of performance. Particularly, the issue of over-training rises up given the fast learning speed. As discussed in the next section, setting a threshold to trigger sooner prediction will alleviate overtraining and save computational resources.

	Accuracy	AUC	F1
50/50%	0.5386	0.5402	0.6439
60/40%	0.5546	0.5564	0.6374
70/30%	0.5431	0.5437	0.6496
80/20%	0.5452	0.5453	0.6448
90/10%	0.5597	0.5625	0.6177

Figure 5a: Train/Test Split Comparison



5b: Forward Chain Mechanism

Predictions Using a threshold

Another interesting technique to improve the accuracy and practical feasibility of the model is to employ a prediction when the model is confident enough. We bounced this idea off from the paper.

Given our tanh activation function layer in the final layer gives an output between -1 and 1, after tuning we employ a threshold of 0.99 in absolute terms on the model output. Therefore, whenever the model gives values Abs(output) >=0.99, we use the prediction. We employ this technique across all model variants and found it improved the performance significantly!

When using the CNN_LSTM model with price data only, the accuracy jumped from 54.5% to 57.6%. Also, we notice that upon backtesting the model without and with transaction costs (covered in PnL and transaction cost analysis section), the model after cost performance improves significantly. Hence, the practical feasibility is significantly enhanced.

Adding External Data

In order to improve the predictability power of our model we decide to investigate if feeding our model with other data than the BTC open prices could lead to better results. After some data analysis, we found out that the BTC is negatively correlated with the USD and positively with the Gold. It could play a role of a safe-haven asset. In that way we have also incorporated the VIX in the external data. We also added the past 4 hours volume of BTC traded.

We have also computed some technical features which are giving indication on the BTC price dynamics. We have computed the realized volatility (using minutes data), the Relative Strength Index which captures the speed and the change of the last BTC open price and the Stochastic Oscillator. We have also computed the Bollinger which is a measure of the volatility and we have computed and normalized the distance between the current price of the BTC and the upper and lower Bollinger Band. In addition of the information on the price dynamics, given that the BTC was by this time mainly traded by retail investors very sensitive to this kind of indicators, we believe that these features can improve the predictability of our model.

When using these features on the CNN_LSTM model along the threshold technique previously mentioned, we increased the accuracy of our model from 57.6% to 66.8%. We notice that the PnL simulation of our model without and with transaction cost improves substantially as shown in the section below.

Backtesting & Transaction Cost Analysis

One of the important considerations before using any model for predictions is how consistent it has been and whether it is profitable after transaction costs. We address both these questions in this section.

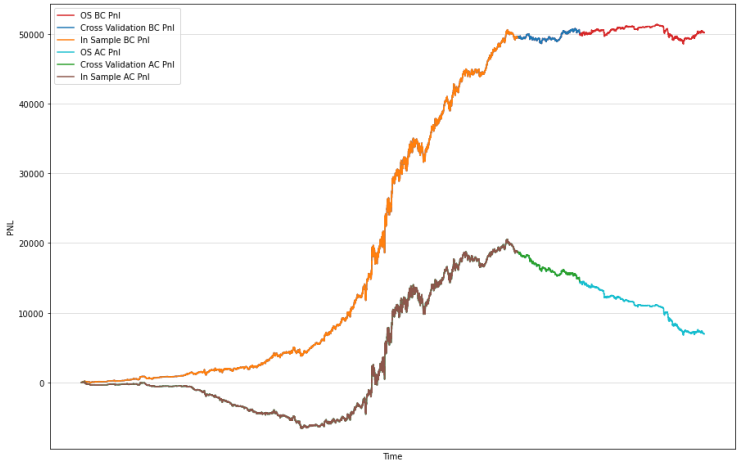
In Backtesting, we check how the predictions of our model would have performed if we had used the model in the past. We leverage all of the data for backtesting across In-Sample (70%), Cross-validation (10%) and Out-of-sample data (20%). Their performance across the different segments of the data has been noted in different colors in the plots below.

For transaction cost (TC), we consider the linear cost model. Using the fact if we open or close 1 BTC worth position on Binance Futures, the cost turns out to be $5. We scale this value (TC coefficient) to the changes in our prediction based positions and adjust TC from PnL generated.

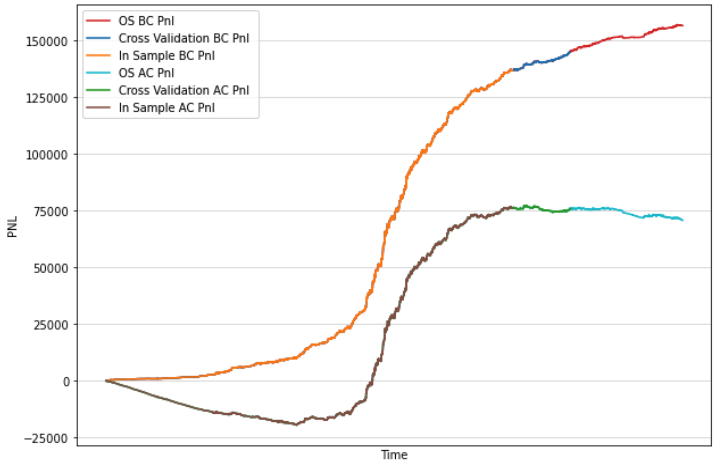
PnL generation: For a given time stamp, we capture the subsequent price movement, scale it by our position size and the predicted direction to compute the profit or loss earned in that time frame. Then, we compute the cumulative sum of the profit/loss outstanding to get the cumulative before-cost PnL. To adjust for transaction cost, we subtract from the profit/loss of any timestamp the absolute change in the position times the transaction cost coefficient. This helps to compute cumulative after-cost PnL

Below are the results we get when we backtest two version of the CNN_LSTM model with price data only (with and without using threshold for predictions):

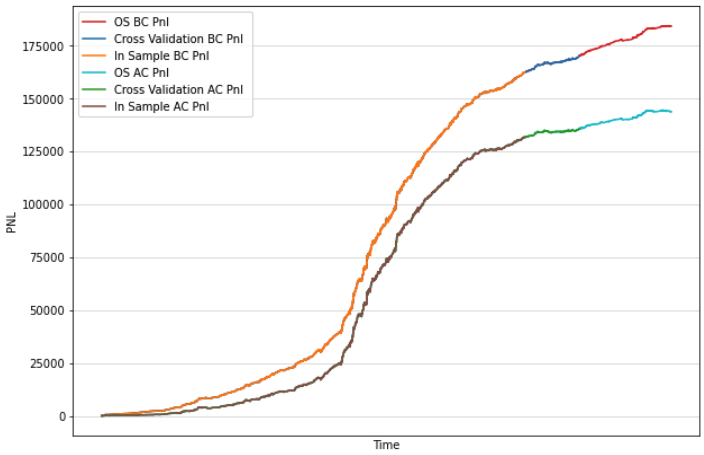
Figure 6a: BTC PnL Analysis (Without Threshold Prediction)



6b: BTC PnL Analysis (Threshold Prediction)



6c: BTC PnL Analysis (Threshold and External data)



As discussed in the ‘Prediction using a threshold’ section, we confirm the significant impact of the technique on both before and after cost performance of the model(s).

Future Works

In this project, we have shown that using neural networks to predict cryptocurrency price movements work well and can be substantially improved by introducing macroeconomic datasets and bitcoin mining information, increasing number of neural network layers, applying hyperparameter tuning, and adding thresholds in training, the last of which particularly enhances the metrics and saves computational time. After some optimization, the model accuracy increases from 0.53 to 0.66.

The model can be further improved in many aspects. We can test multi-step prediction and use a simplified GRU architecture to test the robustness of the framework. The original authors, Pintelas, Livieris, et al., suggested a holistic approach to examine the stationarity of the time series, while Prado (2018) hinted the need to apply fractional differentiation. In future, we can apply more differentiation procedures to refine the prediction.

References

Pintelas E., Livieris I.E., Stavroyiannis S., Kotsilieris T., Pintelas P. (2020) Investigating the Problem of Cryptocurrency Price Prediction: A Deep Learning Approach. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 584. Springer, Cham. https://doi.org/10.1007/978-3-030-49186-4_9

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

https://coinmarketcap.com/

https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Sainath, Tara, Vinyals, Oriol, Senior, Andrew, and Sak, Hasim. Convolutional, long short-term memory, fully connected deep neural networks. In ICASSP, 2015.

KryptoOracle: A Real-Time Cryptocurrency Price Prediction Platform Using Twitter Sentiments. Shubhankar Mohapatra, Nauman Ahmed, Paulo Alencar. Feb 21 2020.

“Bitcoin Exchange: Cryptocurrency Exchange.” Binance, www.binance.com/en/.

“Data.bitcoinity.org.” Bitcoinity.org, data.bitcoinity.org/.

HistData.com, www.histdata.com/.

Keras.com, https://keras.io/api/layers/recurrent_layers/lstm/, https://keras.io/guides/sequential_model/.

https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/

https://stats.stackexchange.com/questions/274478/understanding-input-shape-parameter-in-lstm-with-keras

https://missinglink.ai/guides/keras/keras-conv2d-working-cnn-2d-convolutions-keras/