IJITCE

ISSN 2347- 3657

International Journal of Information Technology & Computer Engineering

Email : ijitce.editor@gmail.com or editor@ijitce.com

International Journal of Information Technology & Computer Engineering

SNNO: 2347–3657
Volume 8,Issue2June ,2020

Predicting the Price of Bitcoin Using Machine Learning

Mrs. Lakshmi Lavanya Tumu , Mrs. Mukka Shirisha, Mrs. Gangula Pavani

Abstract—

The purpose of this article is to assess the feasibility of predicting the future U.S. dollar price of Bitcoin.
The pricing data is obtained from the Bitcoin Price Index. It's a matter of accomplished with varied levels of success by using a Bayesian optimised recurrent neural network (RNN) and a Long Short Term Memory (LSTM) network. The RMSE for the LSTM is 8%, and its classification accuracy is 52%. The famous ARIMA model for time series forecasting \sis constructed as a contrast to the deep learning algorithms. As predicted, the non-linear deep learning algorithms outperform the ARIMA prediction which performs badly. Finally, both deep learning models are benchmarked on both a GPU and a CPU \switch the training time on the GPU surpassing the CPU implementation by 67.7%.

I. INTRODUCTION

The most valued crypto currency, Bit coin [1], is traded on more than 40 exchanges in more than 30 different countries and regions. The current market value of the company is according to https://www.blockchain.info/, has a daily transaction volume of over 250,000 and a market cap of $9 billion USD. Due to Bit coin's relative youth and the associated volatility, which is significantly higher than that of fiat currencies [2], it presents a unique opportunity for price prediction. In addition, its decentralized structure sets it apart from other fiat currencies; for example, there is no comprehensive database of cash transactions or money in circulation for conventional fiat currencies. Prediction of developed financial markets like the stock market has been the subject of much study [3, 4]. As a time series prediction issue in a market that is still in its infancy, Bitcoin provides an intriguing analogy. Time series prediction techniques, such as the Holt-Winters exponential smoothing model, need data that can be decomposed into trend, seasonality, and noise in order to be useful [5]. When there are seasonal influences, such as in sales forecasting, this strategy performs better. These techniques are ineffective because the Bitcoin market is very volatile and does not follow a predictable seasonal pattern. Deep learning is necessary since the job is so difficult. Offers a promising technical answer, supported by its success in analogous settings. Considering the time-

1,2,3 Assistant Professor
1,2,3 Department of CSE
1,2,3 Global Institute of Engineering and Technology Moinabad, Ranga Reddy District, Telangana State.

SNNO: 2347–3657
Volume 8,Issue2June ,2020

Dependent nature of Bitcoin data, recurrent neural networks (RNNs) and Long short-term memories (LSTMs) are preferred over MLPs. The purpose of this article is to examine how well machine learning can forecast Bit coin's price, and to evaluate parallelization approaches run on multi-core and GPU systems. These are the contributions this paper makes: Seven articles out of around 653 published on Bitcoin [6] deal on machine learning for prediction. An ARIMA time series model is also built to compare the neural network models' performance to that of more conventional methods used in financial forecasting. A day's worth of Bitcoin's closing price as measured by the Coin desk Bitcoin Price Index serves as the independent variable here. We don't zero on on a single conversation, but rather a weighted average of the values listed on the five largest Bitcoin markets (Bitstamp, Bitfinex, Coin base, Ocean, and itBit). It would be more efficient to concentrate on a single exchange if we were to execute transactions based on the indications. We measure model quality by calculating the root-mean-squared error (RMSE) of the closing price and then encoding the forecast price into a categorical variable showing an increase, a decrease, or no change in the price. After taking this extra step, traders may get additional performance measures that might aid in the development of a trading strategy, including classification accuracy, specificity, sensitivity, and precision. This paper's data were culled from the websites Coin desk and Blockchain.info. Block chain information, such as the mining difficulty and hash rate, are presented in addition to the closing price, starting price, daily high, and daily low. Two simple moving averages (SMAs) and a smoothed-out closing price are among the elements that have been built (and are used as technical analysis indicators [7]).

II. RELATED WORK

There is a dearth of studies that focus on utilizing machine learning algorithms to forecast Bitcoin prices. As proposed by [9], latent source modeling was put into action in [8] to enable cost forecasting. Bitcoin's 89 percent return in 50 days and 4.1 Sharpe ratios are highlighted. Predictions of Bitcoin values have also been attempted using text data gleaned from social media and other sources. In [10] researchers looked at employing support vector machines, the number of times Wikipedia was seen, and the network hash rate to do sentiment analysis. [11] Looked at the correlation between the value of Bitcoin, the number of tweets about Bitcoin, and the number of page views for Bitcoin on Google Trends.

In a similar vein, [12] used Google Trends views to forecast trade volume rather than Bitcoin price. Small sample sizes and the ease with which false information may spread via (social) media platforms like Twitter or message boards like Reddit might be drawbacks of this kind of research. [13]. There is a severe lack of liquidity in Bitcoin exchanges. There is a higher potential for market manipulation as a consequence. This is why the general mood on social media is disregarded. [14] Used a combination of support vector machines (SVM) and artificial neural networks (ANN) to analyze the Bitcoin Block chain and provide price predictions for Bitcoin, claiming a price direction accuracy of 55% with a standard ANN. Block chain data alone, they reasoned, could not provide enough information for reliable forecasting. Additionally, in [15], Block chain data was used to implement SVM, Random Forests, and Binomial GLM (generalized linear model), with the authors noting a prediction accuracy of over 97%; however, because the authors did not cross-validate their models, their findings were not applicable to a broader audience. The use of wavelets to forecast Bitcoin values has also been documented [16, 17], where it is shown that there are significant positive connections between search engine impressions, network hash rate, and mining difficulty.
These results are expanded upon by include information from the Block chain itself, in the form of the hash rate and the difficulty, as well as information from the main exchanges as given by Coin Desk.

III. METHODOLOGY

The CRISP approach to data mining was used for this work. 1 CRISP-justification DM's over KDD [26], which is more often used for data mining, is based on the commercial context of the prediction problem. Task. The time period covered by the Bitcoin dataset utilized is from August 19, 2013, to July 19, 2016. Figure 1 shows a time series visualization of these data. Since it no longer correctly depicts the network, data collected before to August 2013 has been omitted. The Block chain is used to get the difficulty and hash rate in addition to the Open, High, Low, Close (OHLC) statistics from Coin Desk. The information was also normalized such that its mean was zero and its standard deviation was one. Standardization \saws selected over normalization since it better matches the activation \functions utilized by the deep learning models.

International Journal of Information Technology & Computer Engineering
SNNO: 2347–3657
Volume 8,Issue2June ,2020

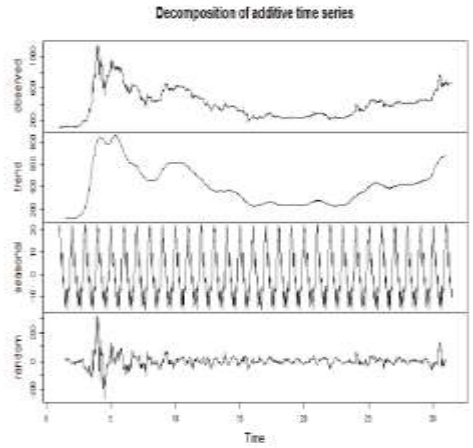

Figure 1: Decomposition of the Bitcoin Time Series Data

A. Feature Engineering and Feature Evaluation

The goal of feature engineering is to simplify prediction by machine learning models by isolating relevant patterns in data. One of the essential components of data mining to successfully complete prediction tasks [27], [28]. Indicators like the Simple Moving Average (SMA) have been used in many recent articles [29, 30] to aid in machine learning categorization tasks. A suitable technical indicator, such as a simple moving average (SMA) of the price over the last x days, is provided as an example.

Borate (a wrapper over the random forest classification technique) was utilized to decide which characteristics to include in the evaluation. Multiple classifiers are used to make a final determination, as in an ensemble technique. Similar to the random forest classifier, this method uses a similar approach to get its results.

It introduces uncertainty into the model by amassing data from an ensemble of random samples used to assess qualities, and it reveals which attributes are more significant [31].

The random forest analysis indicated that all characteristics were crucial to the model, but the simple moving average (SMA) across 5 and 10 days had the most relevance. Also crucial was the final pricing once noise was removed.

B. Deep Learning Models

Models built using deep learning must have their network settings well thought out for optimal performance. These are the three most common methods of parameter selection:

Search strategies like genetic algorithms and grid search are examples of heuristic search strategies that may be used for deep learning models. Specifically, this research used both manual grid search and Bayesian optimization. Elman RNN uses grid search, which entails picking two hyperparameters with a minimum and maximum value. The best-performing parameters may then be found by searching that feature space. This method was used instead of Bayesian optimization since it was more applicable to the parameters in question. Keras, a Python library, was used to create this model [32]. Whenever feasible, Bayesian optimization was opted for when picking LTSM parameters in a manner similar to the RNN. Based on the assumption that the function was sampled from a Gaussian process, this heuristic search strategy keeps track of the posterior distribution of the function as the effects of changing the hyperparameters are noticed. After that, one may choose hyperparameters for the next trial by optimizing the predicted improvement over the best result [33]. On validation data, both the RNN and the LSTM network's performance are measured with safeguards in place to avoid over fitting. We employ dropout in both layers and terminate model training if validation loss does not decrease after 5 iterations.

IV. IMPLEMENTATION

A. RNN

The time interval window was the first variable examined. Literature suggests that such networks have difficulty learning long-term dependencies [34]. Optimization based on gradients. To analyze the connection between the present closing price and past or future closing prices, an autocorrelation function (ACF) was performed on the closing price time series. This is not a foolproof method of prediction at this length, but it beats out the others. In most instances, there is a lag of up to 20 days between the opening price and the closing price, with rarer instances occurring at 34, 45, and 47 days. Consequently, the time period tested over the grid ranged from 2 days to 20 days, 34 days to 45 days, and 47 days to a month. Larger time intervals, up to one hundred days in increments of five, were also tried to guarantee a thorough search. We found that duration of 24 for the window provided the best results. There are more hyperparameters to adjust besides the time window: The network's learning process, known as stochastic gradient descent (SGD), is controlled by the learning rate parameter. In a similar vein, momentum adjusts the learning rate to prevent the model from reaching a local minimum (in terms of error) and instead push it towards the global minimum [35]. The RMSprop optimizer was employed to enhance SGD because it maintains a moving average of recent gradients and is therefore more resistant to data loss [36]. Heaton [37] claims

that the bulk of non-linear functions may be approximated adequately by a single hidden layer. Two more hidden layers were considered, but the results were not as good, therefore the two selected layers were the ones with the lowest validation error. To choose between the input and output nodes, Heaton suggests choosing the number of hidden nodes. If there were less than 20 nodes in each stratum, performance would suffer. Good performance was seen with both 50 and 100 nodes in the tests.

However, this may lead to over fitting and a considerable increase in training time if there are too many nodes. It was decided that the final model should consist of 20 nodes since it was the minimum number required to achieve satisfactory performance. A nonlinear step-by-step equation called an activation function is also required to transmit signals across layers. Tanh, ReLu, and Sigmoid were among the possibilities considered. There was no statistically significant difference between Tanh and the others.

B. LSTM

The LSTM excels at learning long-term dependencies over shorter time scales. Therefore, selecting a long window had less of an effect on the LSTM. This procedure mirrored the RNN's in that it followed a method based on autocorrelation lag. When using a narrower window, the LSTM struggled. The optimal duration was 100 days, and two concealed LSTM layers were selected. With just two layers, you may uncover nonlinear associations in a time series. The RNN model also dictated the selection of 20 hidden nodes across both layers. Bayesian optimization of the network parameters was implemented using the Hyper as library2. The optimum model was sought by the optimizer, which took into account dropout levels in each layer and the best optimizer to use. When compared to other methods, RMSprop once again proved to be the most effective. Due of the LSTM's fixed sequence of tanh and sigmoid activation functions for its various gates inside the cell; these functions were not modified in the LSTM model. Between 50 and 100 epochs, LSTM models converged with early halting. It was observed that, as with the RNN, the batch size had a more significant impact on running time than accuracy. Possible explanation: the dataset is too tiny.

C. Model Comparison

The evaluation metrics are calculated using a confusion matrix that shows the percentage of correct and incorrect classifications. One definition of accuracy is the sum of all the times an estimate was percentages (up, down and unchanged) of right forecasts for price movements. Analysis of sensitivity, specificity, and accuracy is also conducted to address the problem of asymmetrical distribution of benefits (the bitcoin price tends to rise). Detection sensitivity indicates how well a model can identify true positives. How well a model avoids false positives is measured by its specificity. Accuracy, lastly, is a measure of how many correctly categorized forecasts were really useful. Evaluation and comparison of regression accuracy is performed using Root Mean Square Error (RMSE). An 80/20 holdout validation technique is utilized to instrument model assessment.

Since ARIMA models have been widely utilized in price prediction applications, we constructed (and optimized) one in order to simplify a comparison of the deep learning approaches to more conventional ones (e.g. [38], [39]). The data was divided into 5 time intervals, and then projections were made for the next 30 days using an ARIMA model. Several ARIMA models were fitted to the data after first being differenced. The R forecast package's auto.arima function provided the best fit.

V. EVALUATION

As can be seen in Table I, LSTM obtained the highest \accuracy while the RNN achieved the lowest RMSE. When it comes to accuracy and reliability, the ARIMA forecast failed miserably. RMSE. The ARIMA model estimated that the price would grow slightly every day. The model did not produce any false positives. One cause for this may be due \stop the class imbalance in predictive section of the ARIMA \forecast (the price tends to constantly grow) (the price tends to always increase). Because of this, the specificity and accuracy were both very high (both 100%). This doesn't prove stellar efficiency overall, but it does show that it's rather excellent at picking up on shifts in price trend (s).

Table I: Model Results

Model	Temporal Length	Sensitivity	Specificity	Precision	Accuracy	RMSE
LSTM	100	37%	61.30%	35.50%	52.78%	6.87%
RNN	20	40.40%	56.65%	39.08%	50.25%	5.45%
ARIMA	170	14.7%	100%	100%	50.06%	53.74%

Inconclusive findings on the validation dataset show that all models had trouble properly learning from the data. The model achieved an error rate of less than 1% on the training data. On evidence of validity comparatively, the RNN had an error rate of 7.15% and the LSTM was at 8.07%. The neural network

SNNO: 2347–3657
Volume 8,Issue2June ,2020

models' 50.25 and 52.78 percent accuracy is only a hair better than the probabilities of a 50-50 guess in a binary classification challenge (price up vs. down). The RNN was basically of \sno benefit when utilizing a temporal duration above 50 days. Conversely, the LSTM excelled between 50 and 100 days, with optimal performance being achieved at 100 days.

Table II: Performance Comparison

Model	Epochs	CPU	GPU
RNN	50	56.71s	33.15s
LSTM	50	59.71s	38.85s
RNN	500	462.31s	258.1s
LSTM	500	1505s	888.34s
RNN	1000	918.03s	613.21s
LSTM	1000	3001.69s	1746.87s

Table II contrasts many methods to assess their training. The Intel Core i7 2.6 GHz processor was used. The graphics processing unit (GPU) was a 2GB NVIDIA George 940M. Each computer has Ubuntu 14.04 LTS loaded onto a solid-state drive. So that the RNN and LSTM could be compared, we gave them both a batch size of 50 and a temporal duration of 50. When compared to the central processing unit (CPU), the graphics processing unit excelled. When comparing the total amount of time spent training both networks, the GPU was 67.7 percent more efficient. When compared to CPU-only training, RNNs benefited from a 58.8% speedup while LSTMs saw a 70% boost in training speed when using a GPU. Based on the results of the Glances monitoring, the CPU split the algorithm into 7 separate threads. A higher level of parallelism is made possible by the GPU's 384 CUDA cores. In terms of data size, these models were quite modest, consisting of just two layers. Implementing on a GPU will be especially beneficial for more complex models with more layers or more datasets. It's possible that the LSTM's lengthier training time compared to the RNN with the same network parameters is related to the LSTM's need to solve more equations, since more activation functions means more equations. This makes one wonder whether or not an LSTM is worth it over an RNN given the additional computation required. In the world of stock market forecasting, the tiniest of deviations may have a huge impact. That's why it makes sense to use an LSTM in this scenario. In other contexts, the extra calculation isn't worth the marginal performance boost.

VI. CONCLUSION AND FUTURE WORK

Evidently useful for Bitcoin prediction are deep learning models like the RNN and LSTM, with the LSTM being better adept of recognizing longer-term relationships. Contrarily, an excessive variation it's challenging to turn this into spectacular validation findings because of the nature of this assignment. Therefore, it is still a challenging undertaking to do. There's a delicate balance between making sure a model is accurate and letting it learn too much. The dropout function is a great help in doing so. However, although Bayesian optimization helped to improve dropout selection, it was not enough to provide reliable validation findings.

Although the ARIMA prediction seemed to perform well according to the sensitivity, specificity, and accuracy measures, its actual performance was much poorer than the neural network models. The LSTM fared somewhat better than the RNN. The training time for the LSTM, however, is much longer.

An increase of 70.7% in training speed for the LSTM is indicative of the performance improvements derived from the parallelization of machine learning algorithms on a GPU. Model. Better outcomes could be attainable if the work were seen only via a categorization lens. The study is limited in that the model has not been deployed in a real-world situation where it may be used for predictive purposes as opposed to historical analysis. In addition, the model should perform better if predictions can be made using streaming data. In the future, it may be possible to provide sliding window validation, which is not done so in current work. The data itself is noisy, which is an issue.

By examining the model's weights, we may determine whether the dataset's complexity and hash rate variables should be eliminated. To train efficiently, deep learning models need a large quantity of information. There would be 512,640 data points in a year if the data granularity was set to per minute. This kind of information is unavailable for the past, but it is being collected daily from Coin Desk in preparation for the future. As we've seen, parallelization of algorithms is not restricted to graphics processing units (GPUs). There is some evidence that machine learning models perform better on Field Programmable Gate Arrays (FPGA) than on a GPU [40], making FPGAs an intriguing option to GPU devices for parallelization.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

SNNO: 2347–3657
Volume 8,Issue2June ,2020

[2] M. Bri`ere, K. Oosterlinck, and A. Szafarz, "Virtual currency, tangible return: Portfolio diversification with bitcoins," *Tangible Return: Portfolio Diversification with Bitcoins (September 12, 2013)*, 2013.
[3] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.
[4] H. White, "Economic prediction using neural networks: The case of IBM daily stock returns," in *Neural Networks, 1988. IEEE International Conference on*. IEEE, 1988, pp. 451–458.
[5] C. Chatfield and M. Yar, "Holt-winters forecasting: some practical issues," *The Statistician*, pp. 129–140, 1988.
[6] B. Scott, "Bitcoin academic paper database," *suitpossum blog*, 2016.
[7] M. D. Rechenthin, "Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction," 2014.
[8] D. Shah and K. Zhang, "Bayesian regression and bitcoin," in *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*. IEEE, 2014, pp. 409–414.
[9] G. H. Chen, S. Nikolov, and D. Shah, "A latent source model for nonparametric time series classification," in *Advances in Neural Information Processing Systems*, 2013, pp. 1088–1096.
[10] I. Georgoula, D. Pournarakis, C. Bilanakos, D. N. Sotiropoulos, and G. M. Giaglis, "Using time-series and sentiment analysis to detect the determinants of bitcoin prices," *Available at SSRN 2607167*, 2015.
[11] M. Matta, I. Lunesu, and M. Marchesi, "Bitcoin spread prediction using social and web search media," *Proceedings of DeCAT*, 2015.
[12] ——, "The predictor impact of web search media on bitcoin trading volumes."
[13] B. Gu, P. Konana, A. Liu, B. Rajagopalan, and J. Ghost, "Identifying information in stock message boards and its implications for stock market efficiency," in *Workshop on Information Systems and Economics, Los Angeles, CA*, 2006.
[14] A. Greaves and B. Au, "Using the bitcoin transaction graph to predict the price of bitcoin," 2015.
[15] I. Madan, S. Saluja, and A. Zhao, "Automated bitcoin trading via machine learning algorithms," 2015.