Cryptocurrency Price Prediction: A Comparative Study using LSTM, GRU and Stacking Ensemble Algorithm for Time Series Forecasting

MD Ashikul Islam
Curtin University
Lot 9262, Curtin Water 2, Miri, Malaysia
700029975@studuent.curtin.edu.my

Abstract

Technology has significantly reshaped how humans interact with their tangible and intangible surroundings. Cryptocurrency is considered to be one of the most recent technological inventions which revolutionized how we perceive currencies and their functionality. It has become popular because of its safety, security and anonymity. However, volatility remains one of the major issues with cryptocurrencies to this day. Therefore, the primary aim of this paper is to develop LSTM (Long Short-Term Memory), GRU (Gated Recurrent Units) and a Stacking Ensemble Learning algorithm that efficiently predicts the price of a cryptocurrency for a given period of time. The predictions are then observed and analysed to determine the comparative performance of the said algorithms.

Keywords
Cryptocurrency, LSTM, GRU, Stacking Ensemble, Neural Network, Machine Learning, Artificial Intelligence

1 Introduction

The rise of cryptocurrency in recent years has influenced us to re-imagine how society thinks about money. It has attracted a lot of attention because of some unique characteristics as opposed to traditional currency. Being fairly new and completely unregulated, Bitcoin and other Cryptocurrencies has their fair share of drawbacks such as scalability, volatility, security issues etc. It is forecasted that cryptocurrencies will be widely adopted and accepted throughout the modern world within next 20 years. For it to reach that point, risks associated with cryptocurrency shall be assessed and addressed accordingly. One of the major issues with cryptocurrencies, partly because of which it has drawn so much attention is its volatility. Public perception, low adoption rate is some of the major reasons behind its volatile behaviour. Therefore, this paper attempts to predict the price of a cryptocurrency as accurately as possible using three different time series forecasting algorithm. The problem will be approached using LSTM (Long Short-Term Memory), GRU (Gated Recurrent Units) and finally a combination of the two which is called Stacking Ensemble, a special type of meta learning model which combines two well performing algorithm and determines which algorithm is to use at a certain point of time. Stacking Ensemble is known for bringing out the best from the algorithms it uses to learn the context. LSTM is known to perform well on longer sequence of data whereas GRU is known to be computationally efficient. Hence, these algorithms were chosen to solve this linear regression problem.

1.1 Objective

Due to the recent crypto boom, the combined value of cryptocurrencies is increasing on a daily basis. The boom itself was a result of the volatile behaviour of cryptocurrencies. However, it is evident that this boom is a double-edged sword. The volatility that caused the price to soar has also shrunk the market value after a short while. Very recently however, a steady growth has been observed in Bitcoin's price. The total value of all cryptocurrencies has already crossed 2 trillion mark (A. Kahrapal, 2021). Digital currencies are already accepted in some big companies and are expected to be accepted by more companies as we move forward. As a result, they are considered to be a very lucrative investment

ISSN 2961-5011 | https://doi.org/10.54389/NTPV9785

opportunity. Compared to fiat currencies such as Malaysian Ringgit or US Dollar, cryptocurrencies are fairly new and still remains a subject of extensive study. Even though these currencies fluctuate similarly as fiat currencies, what causes these fluctuations is a highly debatable topic. Due to its volatile behaviour, unpredictable factors and wide range of popularity, forecasting its price and behaviour has been a point of keen interest of Machine Learning algorithm developers. Over the past few years, different types of Deep Learning algorithm were applied to accurately predict the price and behaviour of cryptocurrencies. However, it was soon concluded that cryptocurrency time-series forecasting closely relates to the random walk process and it is very complex to accurately predict the price (I. E. Livieris et al. 287, 2021). Moreover, the price of cryptocurrencies over time lacks stationarity. Stationarity refers to a general similarity of pattern which takes place over time. Non-stationary data are always highly volatile which means all features associated with that data unpredictably changes over time. Therefore, a reasonably accurate forecasting algorithm is vital for making a safer investment and understanding cryptocurrencies better in general as these currencies are here to stay. Therefore, this study aims to forecast cryptocurrency price based on previous data using these three different algorithms and comparatively analyse their performance.

2 Neural Networks

Artificial Neural Network (ANN) or simply neural network is one of the most significant and promising technologies of 21st century. However, neural network cannot be considered as a recent invention. The very first concept of an artificial neural network was proposed by Warren McCullough and Walter Pitts in 1944 (L. Hardesty, 2021). The idea, concept and even some terminologies are largely inspired by the functionality of human brain, nature's finest and most complex creation. Artificial neural network's biological counterpart consists of almost a staggering 100 billion neurons. These neurons function as a whole using their massive 100 trillion synapse connections which translates to 1000 synapse per neuron. At its core, artificial neural network attempts to mimic the characteristics of a human brain. The basic element is a Neuron which takes input and produces output based on given constraints. It is very trivial at a single neuron level but when thousands or even millions of neurons are densely interconnected, they can perform complex tasks easily.

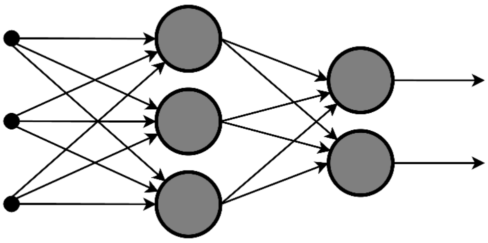


Figure 1. Single Layer Neural Network (Wikipedia)

Modern neural networks consist of several layers of neurons. Neural networks consisting of multiple layers is called Deep Neural Network. A single neuron is usually connected to several other neurons (Figure 1). It receives input from some neurons, processes it and sends it to the rest of the connected neurons. During the training of a learning algorithm, a neuron assigns weight to its incoming neuron connections. While active, the input data is multiplied by the weight and generates a single number result. The result is then passed on to the next neuron until it reaches to the output layer generating a result which is similar to the training data. During this period, the weight and constraints are adjusted to generate an output that complies with training data. This is how a Neural Network functions at a very basic level. However, modern Neural Networks are comprised of more complex procedures to perform advanced tasks.

2.1 LSTM

Long Short-Term Memory (LSTM) was introduced in 1997 to overcome the Vanishing Gradient problem. During that time period, there were other suggested methods to overcome this problem such as Back-Propagation Through Time (BPTT) and Real Time Recurrent Learning (RTRL). However, these methods could not efficiently solve the problem. The LSTM algorithm, however, could learn to bridge the time intervals without losing short time lag capabilities. LSTM makes use of a gradient-based algorithm which implements a constant error value throughout the entire LSTM Neural Network. The constant error flow was implemented using self-connected linear network. A multiplicative input and output gate protects and controls the content of the memory. This results in a complex and efficient memory cell or neuron. The memory cell can be summarized as below (S. Hochreiter and J. Schmidhuber, 1997, p.1735):

$$y^{out_j}(t) = f_{out_j}\left(net_{out_j}(t)\right) \tag{1}$$

$$y^{in_j}(t) = f_{in_j}\left(net_{in_j}(t)\right) \tag{2}$$

Where;’

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u\,(t-1) \tag{3}$$

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u\,(t-1) \tag{4}$$

And

$$net_{c_j}(t) = \sum_u w_{in_j u} y^u\,(t-1) \tag{5}$$

The summation limit u refers to different variables such as input units, output units, gate units or memory cells depending on the state of the network. These units contain information about the present state of the network. For example, an input gate may use information from other gates to determine whether to store or delete its content for the next feed. At time t, c_j’s output $y^{ej}(t)$ is (S. Hochreiter and J. Schmidhuber, 1997, p.1735) (Figure 2):

$$y^{cj}(t) = y^{out_j}(t)h(S_{cj}(t)) \tag{6}$$

Where the internal state of the network is:

$$S_{c_j}(0) = 0 \tag{7}$$

$$S_{c_j}(t) = S_{c_j}(t-1) + y^{in_j}(t)g(net_{cj}(t)) \tag{8}$$

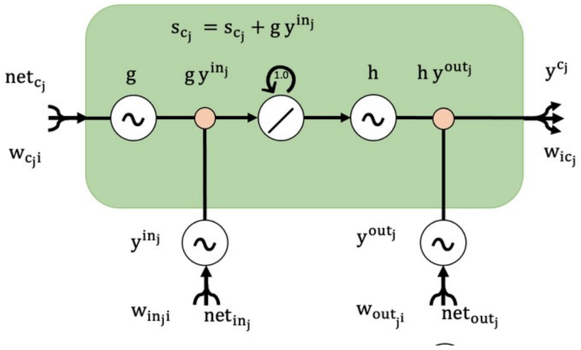


Figure 2. LSTM Memory Cell Architecture (R. Kizito et al. 1)

Gate Units: Gate units are introduced to avoid weight conflicts. The memory cell error flow of c_j is controlled by in_j. out_j controls the error flow from output connections. The network decides which information to keep using in_j and when to access memory cell using out_j.
Learning: A RTRL variation which takes dynamic results caused by input and output gates. Once and error signal triggers a memory cell, it is scaled by the output gate activation. Within the memory cell, the error signal can flow back without any manipulation. However, it is scaled once more by the input gate activation function when it passes through the input gate after leaving the memory cell. LSTM's computational complexity performance is considered to be very efficient which is $O(W)$ where w is the number of input weights.

2.2 Bidirectional LSTM

The concept of a Bidirectional Neural Network was introduced by M Schuster and K Pallial in 1997. The primary goal of this algorithm was to overcome the limitation of a regular RNN. In its very core, Bidirectional Neural Networks are same as a RNN except for one thing. That is another hidden layer which is a identical but reversed input layer. Of these two layers, one works in a positive time direction and the reversed layer works in negative time direction. Simply put, Bidirectional LSTM is two layers of LSTM where one layer works forward and the other working in a backward time direction (Figure 3). The algorithm essentially provides data from both past and future to the Neural Network. Therefore, the network has more context of the data which lets it find hidden connections and correlations between past and the future (M. Schuster and K. K. Paliwal, 1997, p.2673).

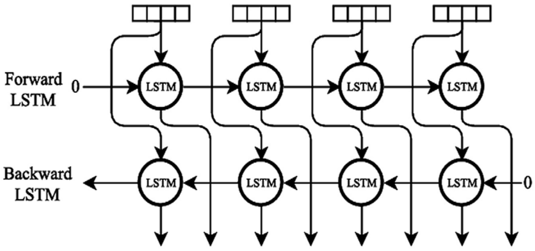


Figure 3. Bidirectional LSTM Architecture (S. Cornegruta et al. 2016, p.17)

Bidirectional LSTMs has proven to be very efficient over time because of its better understanding of the context and time.

2.3 GRU

Gated Recurrent Unit or GRU was introduced by Kyunghyun Cho (K. Cho et al. 2014) in 2014. It was developed based on the principle of LSTM and can be considered a simpler version of LSTM However, unlike LSTM it does not have an output gate and has a reset gate instead. This reset gate modulates the flow of information within the neural network (J. Chung et al. 2014). The gates used in GRU are update gate, reset gate and a current memory gate. GRU is more resource efficient and consumes less time for training than LSTM because of fewer gates as it does not maintain internal cell state like LSTM. The gates of GRU are:
Update Gate: The Update Gate is responsible for determining how much and which information passes through it to future. This essentially prevents the vanishing gradient from happening. The Update Gate can be calculated by:

$$z_j = \sigma[w_z x]_j + [U_z h_{t-1}]_j \tag{9}$$

In the equation above σ represents logistic sigmoid function. x and h_{t-1} represent input and the previous hidden state. W_r and U_r are weight matrices.
Reset Gate: The Reset Gate is responsible for determining how much and which information is to forget. The Update Gate can be calculated by (K. Cho et al. 2014):

$$r_j = \sigma[w_r x]_j + [U_r h_{t-1}]_j \tag{10}$$

The GRU unit is activated by the following equation:

$$h_j^{(t)} = z_j h_j^{(h-1)} + (1 - z_j)\tilde{h}_j^{(t)} \tag{11}$$

Where;

$$\tilde{h}_j^{(t)} = \Phi([w_x]_j + [U(r \odot h_{(t-1)})]_j) \tag{12}$$

where *r* represents Reset Gate and ⨀ is an element-wise multiplication. In this above formulation, when the value passed from reset gate is close to zero, the hidden state must ignore the previous hidden state and only accept the current state. This is how Reset Gate effectively filters irrelevant information. The update gate controls the amount of information to be passed over to the hidden state. This is where GRU is similar to LSTM and it helps remember long term information to be utilised during output. In simple terms, GRU lacks memory unit unlike LSTM. Hence, the entire hidden state is exposed and can be taken advantage of. Since it has less gates than LSTM, it can train faster than a LSTM.

2.4 Ensemble Learning Algorithm

Ensemble Learning refers to a learning model where two or more Machine Learning models are combined or generalized to achieve better performance. It is known to improve the result accuracy of different problems such as classification, regression etc. When two or more learning algorithm displays similar performance for a problem, Ensemble Learning algorithm can be used to improve the overall performance and achieve better results. It can also be used to error reduction, choosing optimal features, incremental learning, non-stationary learning etc. Technically, the number of possible ensemble learning algorithm can be unlimited due to a lot of possible combination and approach. However, regardless of possible combinations, the principal concept is same for all applications. There are a different types of Ensemble methods such as Bagging, Boosting, AdaBoost (Adaptive Boosting), Stacking etc. Stacking or Stacking Ensemble was used in this paper and discussed further below.

2.5 Stacking Ensemble Learning Algorithm

Stacking Ensemble is also known as Stacked Generalization. It was first introduced by David H. Wolpert back in 1991(D. H. Wolpert, 2009, p.2776). Stacking Ensemble is capable of combining different types of models to produce a more accurate result. The architecture of Stacking Ensemble consists of two types of models. These are Base model and Meta model or Meta-learning algorithm (Figure 4). Base models are models which will be used by the Meta-learning algorithm to make better generalization. The approach is such that the Ensemble learning algorithm can determine how to best combine the base models. In its core, Stacking Ensemble use trainable combiners. The combiners are trained on the base models to learn which model performs well on which period and produces a generalized and usually more accurate results than its base models (T. Schaul and J. Schmidhuber, 2010, p. 4650). Stacking is justified when the base models have similar performance on the same problem. Stacking can be used to improve overall solving accuracy. However, better performance than the base models is not always guaranteed. The performance depends on a number of factors. These factors include but are not limited to base model complexity, proper representation and sampling of training dataset, learning approach of the base models etc.

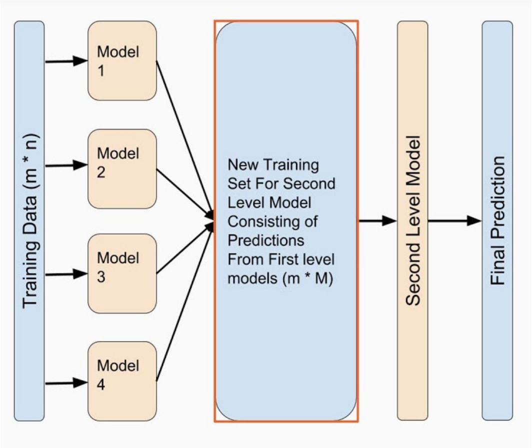

Training Data (m * n)
Model 1
Model 2
Model 3
Model 4
New Training Set For Second Level Model Consisting of Predictions From First level models (m * M)
Second Level Model
Final Prediction

Figure 4. Stacking Ensemble Architecture (S. Cornegruta et al. 6103)

3 Data Acquisition & Analysis

Acquiring clean and relevant data is arguably the most important steps when it comes to training a Machine Learning model. Big data and its potential application are the reason Machine Learning has become mainstream today. It does not matter how big the dataset is, if it is flawed or inaccurate, the ML algorithm, however advanced it is, is bound to yield a garbage output.

Dataset for this project was collected from *www.binance.com* using an Application Programming Interface (API). The collected dataset was of the cryptocurrency called Binance Coin (BNB).

The time line of the collected data is 1 year. It means whenever the API in run, it will collect past 1 years BNB data from Binance website. However, only the last three month's data was used to train and test the ML algorithm. This is because of the limitation of Graphics Processing Unit (GPU). Even though modern GPU's have a very high computational power in comparison to old ones, processing a one year long time sequence would take extremely long time and might even crash the compiler.

Upon collecting the data, it is stored in a text file in the local drive. A total of five data points were collected for BNB which are: *time, price high, price low, price, volume*. *time* refers to the time when the observation was recorded, *price high* and *price low* refers to the highest and lowest price for a given timeline observation. *price* refers to the closing price of the day and *volume* is the amount of trade that took place to and for BNB. The collected data sample can be summarized as below (Table 1):

Table 1: BNB Dataset Structure

Time	Price High	Price Low	Price	Volume
1622381699999	329.11000000	328.04000000	328.58000000	2529.41150000
1622381759999	328.62000000	327.68000000	327.98000000	4581.87200000
1622381819999	328.75000000	327.45000000	327.87000000	1799.25960000
1622381879999	328.00000000	327.63000000	327.63000000	3000.92770000
1622381939999	328.35000000	327.79000000	327.79000000	1873.67880000

3. 1 Data Pre-processing

Data Pre-processing refers to the modifying and structuring the dataset before feeding it to the ML algorithm. Without structuring of some extent, the raw data may cause anomaly in the result. Therefore, it is important to modify the raw data to follow a structure. For this project, only past three month's data will be used to train and test the algorithms. The interval of the collected data was 1 minute. This means that all five datapoints collected were within 1 minute's interval. The data pre-processing includes randomly removing a portion of the data and scaling/normalizing it. Data Scaling is vital to maintain a certain range of the output. Cryptocurrencies are known to be extremely volatile. Therefore, the difference between the high and low price for a certain time period might be extremely fluctuating. This fluctuation might result in a very strange output. Moreover, these fluctuating variables may not contribute equally in the training and end up creating a bias. To avoid this, input dataset in scaled within a certain limit, that is limiting the high and low point of the input to a certain range. In this project, *MinMaxScaler*() was used to transform the data with a limit of [0,1]. This means the highest value is going to be 1 and lowest will be 0. The *MinMaxScaler*() uses the following equation to scale the data:

$$StandardX = (x - x.min(axis = 0))(x.max(axis = 0) - x.min(axis = 0)) \tag{13}$$

$$ScaledX = StandardX * (max - min) + min[24] \tag{14}$$

where *min* and *max* are the range of the features.
The results are calculated and documented in terms of accuracy, mean absolute error, max absolute error and root mean squared error. Accuracy can be mathematically defined as follows:

$$Accuracy = 100\% - ErrorRate \tag{15}$$

Where;

$$ErrorRate = |ObservedV\ alue - ActualValue|\ ActualV\ alue * 100 \tag{16}$$

Mean absolute error can be mathematically defined as:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{17}$$

where y_i is prediction, x_i is actual value and n is total number of samples.

4 Bidirectional LSTM and GRU Architecture

The LSTM and GRU model were given the same input parameters and structure. Essentially, the only difference is the algorithm itself and its underlying functionality. Both LSTM and GRU model are given a generalized summary below. The models are initiated by defining a sequential stack of layers. It creates a linear layer stack of sequential instances. Sequential model can only have a single input and output. Upon defining the model type, the Bidirectional LSTM and GRU layers are implemented. This layer has 100 memory units. Every unit takes up to 200 input weights. It takes a total of 94400 input parameters. This layer also takes the *lookback* value as a parameter. The *lookback* variable defines how many minutes of past data are being taken into consideration for prediction. The algorithm is predicting the highest possible price for next hour taking the *lookback* value, training data and other defined parameters. The next two layer is a dense layer with 100 and 50 memory units each. Dense layers implement the following method:

$$output = activation\ (dot\ (input,\ kernel) + bias) \tag{18}$$

Both these layer uses the *Relu* activation function. *Relu* applies the rectified linear unit activation function. It uses a non-zero multiple of the input and modifies the maximum values of the activation. Kernel refers to the input weights. The bias variable is not applicable for this model. The first dense layer takes 20100 input parameters and the second dense layer takes 5050 parameters. The last or the output layer is another dense layer with *Sigmoid* activation. For values that are less than 5, sigmoid function returns a value close to zero and for values more than 5, it returns 1. The architecture of the model can be summarized as below:

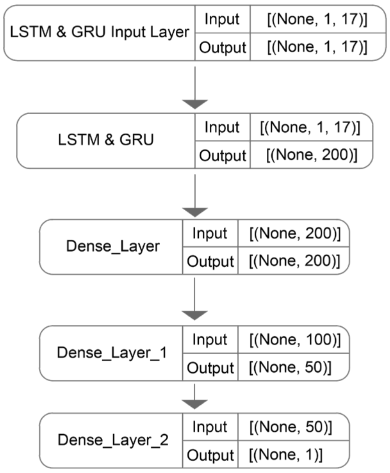


Figure 5. LSTM and GRU Model Architecture

The models are trained using the 'Adam' optimizer. It is a stochastic descent method which depends on adaptive estimation of first and second order moments. Adam is a very efficient algorithm with a optimized memory usage. It is well suited for large amount of data and parameters (D. P. Kingma and J. Ba, 2014). The model's layers, memory units, weights, input size and parameters can be summarized by below table (Table 2):

Table 2: Layer Structure and Parameter

Layer	Output Shape	Parameter
LSTM and GRU Layer	(None, 200)	94400
Dense	(None, 100)	20100
Dense 1	(None, 50)	5050
Dense 2	(None, 1)	51

Learning Rate: Learning Rate is arguably one of the most important hyperparameters in a ML model. This parameter decides the amount of change that is applied when the amount of error is calculated each time with new input weights. It is important because a smaller value may cause the model to take a significant amount of time to train or it may even get stuck while training. On the other hand, a larger value may make the model to consider inappropriate weights to include in the computation. Learning Rate essentially refers to the number of weights which are taken into account during the training process. It is a variable that often is a positive number ranging between 0 and 1. Learning Rate can be defined by the following equation:

$$n_{n+1} = \frac{n_n}{1+ d_n} \tag{19}$$

Where *n* is the learning rate, *d* is a decay parameter and *n* are the iteration step.

4.1 Stacking Ensemble Architecture

The Stacking Ensemble model implemented in this research is a Linear Regression model. Linear Regression visualizes the relationship between two variables by imposing a linear equation. It can be summarized by the following equation:

$$Y_i = f\left(X_{i,}\beta\right) + e_i \tag{20}$$

Where Y_i is the dependent variable, f is function, $X_{i\ is}$ the independent variable, β is unknown parameter and e_i is the error terms.

The meta-learner trains based on outputs from the LSTM and GRU model and the ideal values. Upon completion, the algorithm learns how to best combine the values based on the given training output and ideal values. In terms of implementation, the *LinerRegression*() model works based on least squares or non-negative least squares acting as a predictor. Passing X and Y into this as training data returns the coefficient of determination of the prediction. The coefficient of determination R^2 is defined as $1 - \frac{u}{v}$, where u is the residual sum of squares and v is the total sum of squares.

5 Results and Analysis

Since this is a comparative study, both LSTM and GRU algorithm will be trained and tested on the same dataset and setting. Hence, the parameters such as epoch, learning rate, lookback value will be same for both algorithms. Both LSTM and GRU algorithm was run with two variations of 50 and 100 epochs. The prediction timeline is 1 hour into the future. This means the algorithms will predict the highest price for next one hour. The reasoning behind changing parameter is that it provides a clearer picture of the algorithm's performance.

5. 1 LSTM

Prediction of highest price for next 1 hour

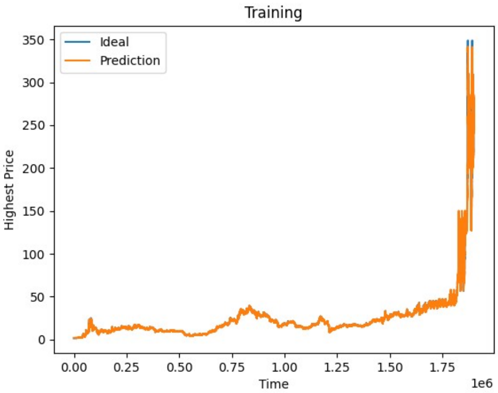


Figure 6. LSTM Training Accuracy

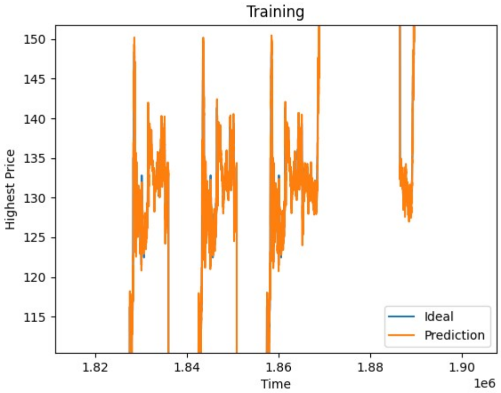


Figure 7. LSTM Training Accuracy (Zoomed in)

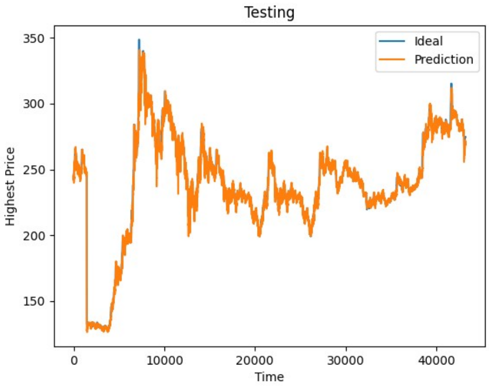


Figure 8. LSTM Testing Accuracy

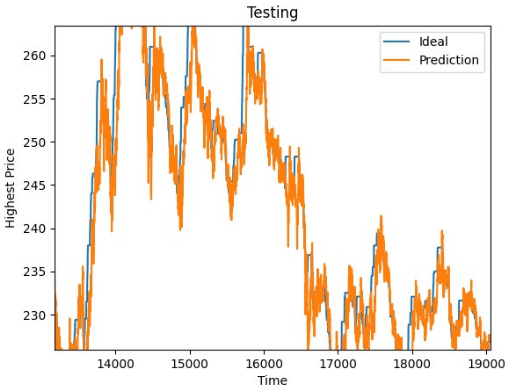


Figure 9. LSTM Testing Accuracy (Zoomed in)

5.2 GRU

Prediction of highest price for next 1 hour

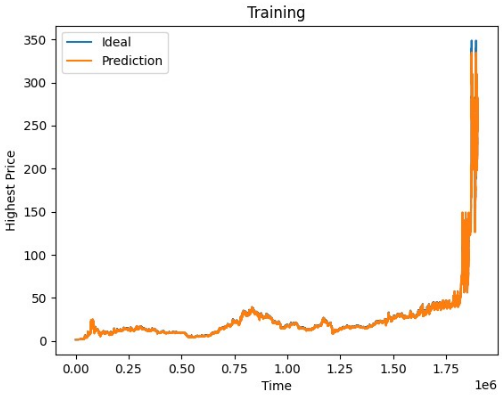


Figure 10. GRU Training Accuracy

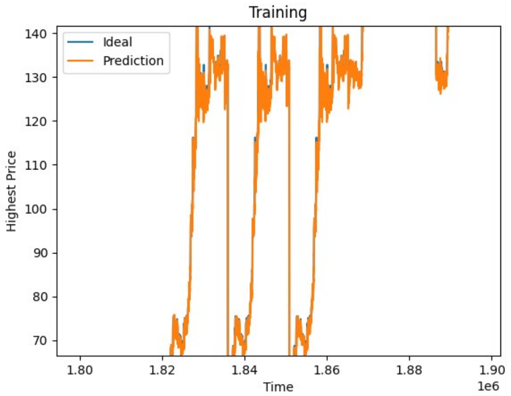


Figure 11. GRU Training Accuracy (Zoomed in)

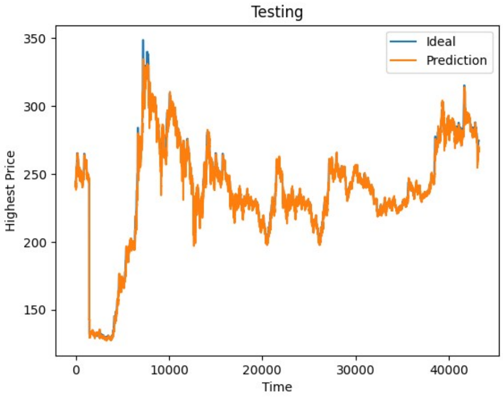


Figure 12. GRU Testing Accuracy

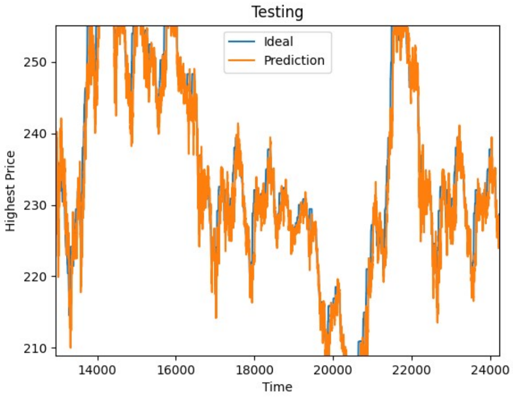
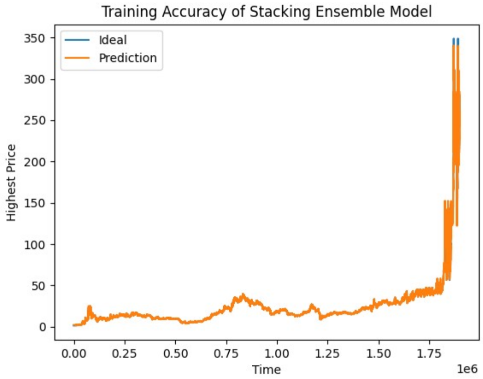


Figure 13. GRU Testing Accuracy (Zoomed in)

5.3 Stacking Ensemble

Prediction of highest price for next 1 hour



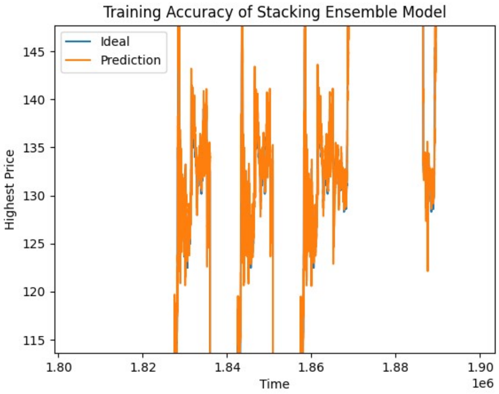
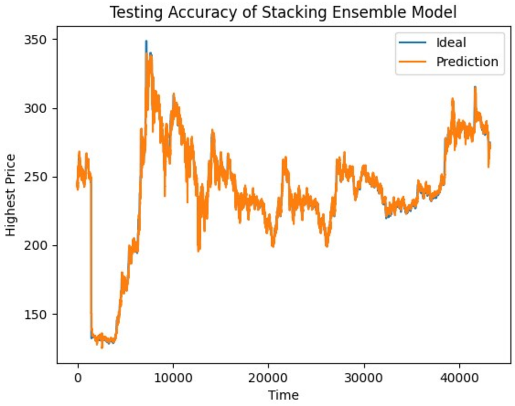


Figure 14. Stacking Ensemble Training Accuracy Figure 15. Stacking Ensemble Accuracy (Zoomed)

5.3 Stacking Ensemble

Prediction of highest price for next 1 hour



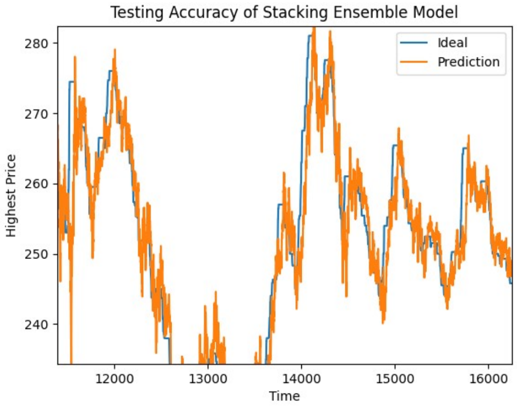


Figure 16: Stacking Ensemble Testing Accuracy Figure 17: Stacking Ensemble Accuracy (Zoomed)

5.4 Comparative Analysis

In the above sections, all results from the constituent algorithms are documented individually (Figure 6-17). However, it is very difficult to interpret individual graphs in a comparative study. In the following graph, the testing result from all algorithms are merged together to portray a clear picture. All numerical results such as training accuracy, testing accuracy, mean squared error etc has been documented in a tabulated format (Table 3).

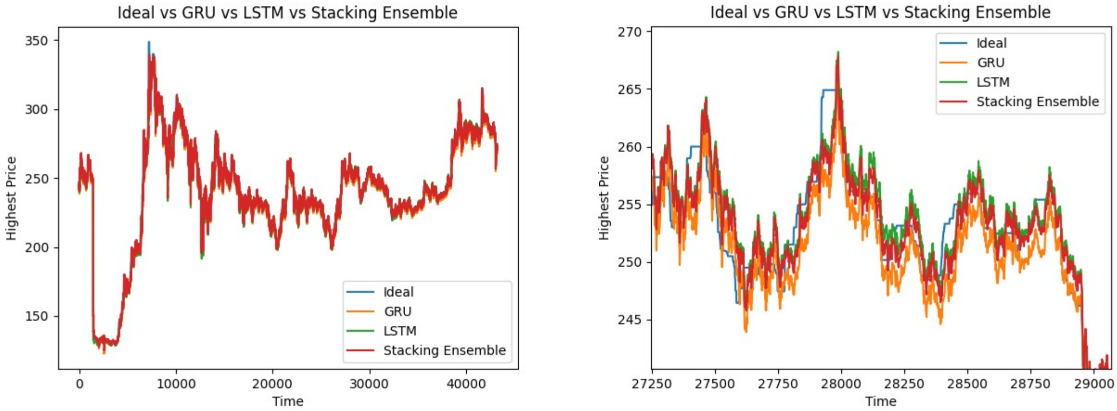


Figure 18: Ideal vs GRU vs LSTM vs Stacking Ensemble Training and Testing Accuracy respectively

Table 3: Numerical comparison between result instances

LSTM		GRU		Stacking Ensemble	
Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy	Training Accuracy	Testing Accuracy
97.53%	98.43%	97.42%	97.75%	99.14%	99.20%
MeanAE Train	MeanAE Test	MeanAE Train	MeanAE Test	RMSE Train	RMSE Test
.35	2.65	.39	3.05	.63	3.31

Table 3: Numerical comparison between result instances

From figure 13, we can see that green and red lines tend to overlap each other which means that LSTM and Stacking Ensemble are producing impeccably similar results. The LSTM model for Stacking Ensemble algorithm was trained with 100 epochs and 0.0001 learning rate and the same goes for GRU. With same parameters and hyper-parameters. It can be concluded that LSTM tends to perform better than GRU in this scenario. The accuracy of GRU tends to fluctuate more than its counterparts which is consistent over time. The similarity between the results of LSTM and Stacking Ensemble denotes that the meta learning algorithm tends to be somewhat biased towards LSTM which is justifiably correct since LSTM is producing better results than GRU and displaying more consistency over time. The observed trends, non-stationarity and seasonal cycles of the test data over time makes it difficult for the algorithms to keep up. However, despite all these hindrances, Stacking Ensemble displayed promising results which has significant real-world implications.

The table 2 illustrates a comparative numerical metrics of LSTM, GRU and Stacking Ensemble. The LSTM and Stacking Ensemble resembles the graph interpretation showing high level of similarity.

The difference between LSTM and Stacking Ensemble results is quite small in terms of numeric. However, this small gap can make a huge difference in real-world applications. For problems where both performance and accuracy both are important, Stacking Ensemble algorithm can bring out the best from both of its constituent algorithms. Their error metrics, although different, are mathematically similar.

From the analysis above, we can conclude that, Stacking Ensemble performs better than its individual constituents. Even though the difference is very small in number, it can make a huge difference in practical applications.

6 Conclusion

Through this paper, LSTM, GRU and Stacking Ensemble algorithms were implemented to predict the price of cryptocurrency BNB. The overall analysis of the result shows that the predictions are reasonably accurate in different parameters. With almost 99% testing accuracy, Stacking Ensemble Neural Network showed a promising capability to predict even the most volatile of time-series data. The results may not yet be suitable enough to make a real-world decision based on it but it definitely shows a very promising start among other things. LSTM is a very widely used algorithm which is known to produce very good results. From the analysis of this paper, it can be seen that with correct implementation and some trial and error, Stacking Ensemble can overcome the shortcomings of LSTM and GRU, in other words, its constituent algorithms. With proper training data, feature extraction and possible implementation of social signal integration, it can be used to make real world decisions. The neural network, however, has demonstrated some performance issues which can be overcome which will possibly result in a much better accuracy in predicting the future prices.

7 Acknowledgement

I would like to thank my family for their continuous support in my journey. It would not have been possible for me to come this far without their unconditional support. I would also like to extend my gratitude to Dr. Yi Chiew Han for his continuous support throughout the execution of this project. His guidance carried me through all the steps of completing this project.

References

A. Kharpal, *Cryptocurrency market value tops USD 2 trillion for the first time as Ethereum hits record high*, https://www.cnbc.com/2021/04/ 06/cryptocurrency-market-cap-tops-2-trillion-for-the-firsttime.html, [Online; accessed 21-May-2021], Apr. 2021.

I. E. Livieris, N. Kiriakidou, S. Stavroyiannis, and P. Pintelas, "An advanced cnn-lstm model for cryptocurrency forecasting," *Electronics*, vol. 10, no. 3, p. 287, 2021.

L. Hardesty, explained: Neural networks, https://news.mit.edu/2017/ explained- neural- networks-deep- learning- 0414, [Online; accessed 23-May-2021], Apr. 2017.

A neural network with multiple layers, https://en.wikipedia.org/wiki/Artificial_neural_network/media/File:Multi-Layer_Neural_Network-Vector-Blank.svg, [Online; accessed 23-May-2021], Apr. 2015.

M. Minsky and S. A. Papert, Perceptrons: An introduction to computational geometry. MIT press, 2017 S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

M. Schuster and K. K. Paliwal, \Bidirectional recurrent neural networks, "IEEE transactions on Signal Processing, vol. 45, no. 11, pp. 2673{2681, 1997.

R. Kizito, P. Scruggs, X. Li, M. Devinney, J. Jansen, and R. Kress, \Long short-term memory networks for facility infrastructure failure and remaining useful life prediction," IEEE Access, vol. PP, pp. 1{1, May 2021. doi: 10.1109/ACCESS.2021.3077192.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

S. Cornegruta, R. Bakewell, S. Withey, and G. Montana, \Modelling ra- diological language with bidirectional long short-term memory networks,"pp. 17{27, Jan. 2016. doi: 10.18653/v1/W16-6103.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using rnn encoder-decoder for statistical machine translation*, 2014. arXiv: 1406.1078.

S. Cornegruta, R. Bakewell, S. Withey, and G. Montana, "Modelling radiological language with bidirectional long short-term memory networks," pp. 17–27, Jan. 2016. doi: 10.18653/v1/W16-6103.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014. arXiv: 1412.3555[cs.NE].

D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992. R. Polikar, "Ensemble learning," *Scholarpedia*, vol. 4, no. 1, p. 2776, 2009, revision #186077. doi: 10.4249/scholarpedia.2776.

T. Schaul and J. Schmidhuber, "Metalearning," *Scholarpedia*, vol. 5, no. 6, p. 4650, 2010, revision #91489. doi: 10.4249/scholarpedia.4650. K. B. Korb, "Introduction: Machine learning as philosophy of science," *Minds and Machines*, vol. 14, no. 4, pp. 433–440, 2004.

D. P. Kingma and J. Ba, \Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.