Crypto Currency Price Prediction Using Machine Learning Python

Omkar Singh[1], Leanne Fernandes[2,] Atharva Malusare[3], Santanu Mandal[4]
[1] HOD & Research Group Head Dept. of DS, Mumbai University, India
[2]Dept. of IT, Thakur College of Science & Commerce, India
[3]Dept. of IT, Thakur College of Science & Commerce, India
[4]Dept. of IT, Thakur College of Science & Commerce, India

Abstract– Starting in the 1980s, cryptocurrencies, then referred to as cyber currencies began to become popular. In this respect, the history of cryptocurrencies can be traced back to the 1980s when they were called cyber currencies. Bitcoin, the first digital currency based on blockchain technology was introduced in 2008 which led to the rise of alternative cryptocurrencies and attracted interest for its potential effect on even financial systems during this time frame blockchain technology developed while other cryptocurrencies emerged forcing attention towards it and other sectors due to its potential consequences in financial systems and beyond since then regulatory implications and wider interest have been spawned by both that growth of cryptocurrency market and advancements in blockchain technology.

Key Words: Crypto Currency, LSTM, RNN, Price Prediction, Machine Learning, Bitcoin, Ethereum, Dogecoin, Binance, Cardano.

1. INTRODUCTION

As we all know that in the past few years Crypto Currency has taken an immense Growth. The price of Bitcoin has gone from 600$ in 2016 to 63558$ in April 2021 currently there are not sufficient tools for analysis and methodologies for accurately predicting crypto prices of the currencies.

This scarcity presents a challenge for investors and analyst who seek reliable means to forecast the price movements of digital assets within the crypto currency market.

One of the key challenges for investors and traders in the crypto currency market is predicting the price movements of these digital assets. Machine learning algorithms can analyse historical data, identify patterns, and make predictions based on these patterns. When applied to crypto currency price prediction, machine learning models can assist investors in making informed decisions about buying, selling, or holding crypto currencies.

Crypto currency is a peer-to-peer system that can enable any person anywhere to receive or send payments. It is a digital payment system that don't rely on banks to verify transactions. It is stored in a digital wallet. It received its name because it is using encryption to verify all the transactions.

Seeing in the increasing economics and geopolitics issues from last 2 years global currency value has been decreased, all the investors had a bad fall in stock market and have lost their wealth. This has started people's interest in digital currencies.

Therefore, our system helps in crypto currency price prediction using machine learning python. Some traders and analysts may use other methods like fundamental analysis, technical analysis to predict prices but there is always some risk and uncertainty. With this system the traders and investors have various new opportunities to explore new approaches and incorporate advancements in AI, data analytics, other relevant fields and tailored prediction solutions to suit different trading or investment strategies.

2. ORIGIN

Starting in the 1980s, cryptocurrencies, then referred to as cyber currencies began to become popular. In this respect, the history of cryptocurrencies can be traced back to the 1980s when they were called cyber currencies. Bitcoin, the first digital currency based on blockchain technology was introduced in 2008 which led to the rise of alternative cryptocurrencies and attracted interest for its potential effect on even financial systems during this time frame blockchain

technology developed while other cryptocurrencies emerged forcing attention towards it and other sectors due to its potential consequences in financial systems and beyond since then regulatory implications and wider interest have been spawned by both that growth of cryptocurrency market and advancements in blockchain technology

3. MACHINE LEARNING MODELS AND ANALYSIS

For this project we have used the LSTM model (Long Short-Term Memory) which is an extensive part of RNN (Recurrent Neural Network). The LSTM is mainly designed to address the vanishing gradient problem. First of all, let's understand what RNN is and how it is works. RNN is a type of neural network specially works in the sequential data such as time series prediction or more commonly in cryptocurrency prediction. In RNN the output of the previous step is fed as an input to get the output of the next step. In traditional Neural Network, all the inputs and outputs are independent from each other but in some cases to predict the next word or next output, the previous output is required and as a result it is required to remember the previous output. Thus, RNN solved this with the help of Hidden Layer or Hidden State. The RNN works with the sequential data, this hidden state helps the RNN to remember the information about the sequence of the data. At each time step, the network takes an input vector and combines it with the hidden state from the previous time step to produce an output and update the hidden state. This recurrent connection allows RNNs to incorporate information from previous time steps into the current prediction or output. Furthermore, understand how a hidden state in RNN works with some examples. Let us consider the following two input and output of the sequences

XY=[a,b,c,d,…,y,z]=[b,c,d,e,…,z,a]

We will first try to train a MLP (Multi-Layer Perceptron) with one input and output from X and Y. We can write this relationship in maths as f(x) →y where x is an element of X and y is an element of Y and f (·) is our MLP. After training, if given the input a=x, our neural network will give an output b=y because f (·) learned the mapping between the sequence X and Y. Now, let's try to teach other sequences to the same MLP.

XY=[a,a,b,b,c,c,⋯,y,z,z]=[a,b,c,⋯,z,a,b,c,⋯,y,z]

More likely, this MLP will not be able to recognise or learn the relationship between X and Y. This is because a normal MLP can't learn and understand the relationship between the previous and current outputs. Now, we will use the same sequences to train an RNN. In general, in an RNN we take two inputs one for our input and the previous hidden values and two outputs one for the output and the next hidden values. F(x, ht) →(y, ht+1)

Important: here ht+1 represents the next hidden value. Below we will execute some sequences of this RNN model.

x = a and h = 0 (a, next_hidden) <- f(x, h) prev_hidden = next_hidden x = a and h = prev_hidden (b, next_hidden) <- f(x, h) prev_hidden = next_hidden x = b and h = prev_hidden (c, next_hidden) <- f(x, h) prev_hidden = next_hidden If we look at the above process we can see that we are taking the previous hidden state values to compute the next hidden state. What happens is while we iterate through this process prev_hidden = next_hidden it also encodes some information about our sequence which will help in predicting our next character.

About LSTM:

The LSTM is an extension of recurrent neural networks which is mainly designed to overcome the limitations of RNN. The vanishing gradient is particularly problematic for traditional RNN because they are unable to retain information over long sequences. Thus, LSTM is designed to overcome the issue with the help of their gate architecture which helps to regulate the flow of information and gradients throughout the network. LSTM contains a memory cell, which is a container that can hold information for a long period of time. These networks are capable of learning long term dependencies in a sequential data, which makes them suitable for task such as time series forecasting. In LSTM the memory cell is controlled by the three gates that are input gate, output gate and forget gate. These gates are responsible on managing the information on what should be add, remove to and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell and the output gate controls what information is output from the memory cell.

Architecture and working of an LSTM:

The LSTM architecture has a chain structure that contains four neural networks and different memory blocks called cells. Information is retained by the cells and the memory manipulations are done by the gates. In mathematically this gate is computed with the help of a sigmoid activation function. The sigmoid activation functions. The sigmoid activation function is a mathematical function which is often used in neural networks. It is called "sigmoid" because it's in S-shaped curve.

- The Input Gate: The input gate decides which new information to add in the cell state. It determines whether the current input and the previous hidden state is relevant for updating the cell state. The output of an input gate (between 0 and 1) is multiplied with the output of tanh block that produces the new values that must be added to previous state. This vector is then added to the previous state to generate the current state.

Mathematically, the input gate it is computed using a sigmoid activation function and is typically represented as follows: $it = \sigma (Wi \cdot [ht-1, xt] + bi)$

Where:

Wi is the weight matrix associated with the input gate, [ht−1, xt] is the concatenation of the previous hidden state ht−1 and the current input xt, bi is the bias vector for the input gate, and σ is the sigmoid activation function, which squashes the input to a value between 0 and 1.

- The Output Gate: The output gate controls the flow of the information from the cell state to the hidden state. It determines how much of the information stored in the cell state should be output at the current time step. In output gate, the input state and the previous state are gated as before so that to generate another scaling fraction that is combined with the output of tanh block that brings the current state. Mathematically, the output gate ot is computed using a sigmoid activation function and is typically represented as follows:

$ot = \sigma (Wo \cdot [ht-1, xt] + bo)$ Where:

Wo is the weight matrix associated with the output gate,

[ht−1, xt] is the concatenation of the previous hidden state ht−1 and the current input xt, bo is the bias vector for the output gate, and σ is the sigmoid activation function, which squashes the input to a value between 0 and 1.

- The Forget Gate: Just like the output gate, the forget gate controls the flow of the information from the previous cell state to the current cell state. It determines which information from the previous state should be retained and which should be removed. In forget gate the input is combined with the previous output to generate a fraction between 0 and 1 that determines how much of the previous state need to be retained or in other words, how much of the state should be forgotten.
- Mathematically, the forget gate ft is computed using a sigmoid activation function and is typically represented as follows:

$$ft = \sigma (Wf \cdot [ht-1, xt] + bf)$$

Where:

Wf is the weight matrix associated with the forget gate, [ht−1, xt] is the concatenation of the previous hidden state ht−1 and the current input xt, bf is the bias vector for the forget gate, and σ is the sigmoid activation function, which squashes the input to a value between 0 and 1.

4. METHODS

While implementing our machine learning model we have first came up with the idea of random forest classifier which is a supervised machine learning algorithm used for classification, regression and other tasks. This classifier is specially used for handling complex datasets which contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. As a result, we have first created the model using the random forest classifier has it features to handle large number of datasets using it decision trees. This classifier works by creating number of decision trees during the training phase. Each tree is constructed by using a random subset of the data set which introduces the randomness and diversity into the trees preventing them from being too similar to each other and as a result it also reduces the risk of overfitting the data in the model. This classifier is also useful in working with the missing data within the dataset as it creates decision trees. Mathematically, the prediction of a Random Forest classifier can be represented as follows:

Let T be the set of decision trees in the forest, and fi (x) be the prediction of the i-th decision tree for input sample x.

For a classification task, the predicted class ŷ for input sample x is: ŷ = mode{fi(x)|∀i∈T}

For a regression task, the predicted value ŷ for input sample x is:

ŷ = |T|1∑i=1|T|fi(x)

Where |T| denotes the number of decision trees in the forest.

Next model we have implemented by using the Extreme Gradient Boosting Classifier or in simple XGBoost Classifier. It is a powerful algorithm especially used for gradient boosting. Gradient boosting is a powerful ensemble learning technique used for both regression and classification tasks in a machine learning. This XGBoost is known for its efficiency, speed, and effectiveness in producing high-quality predictive models. Additionally, XGBoost is efficient in handling of missing values, which allows it to handle real-world data with missing values without requiring significant preprocessing. It has a built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time. As a result, we have implemented the second model using this algorithm. Mathematically, the prediction of an XGBoost classifier can be represented as follows:

The XGBoost classifier makes predictions by aggregating the outputs of all the weak learners (trees) weighted by a shrinkage parameter η ŷ = ∑k=1Kη·fk(x)

Where fk(x) is the prediction of the k-th tree for input sample x.

In Closing we have implemented the third model by using the Long Short Term Memory algorithm or in short the LSTM which is the extension of Recurrent Neural Network (RNN). IT is specially designed to overcome the issue of vanishing gradient problem inherent in traditional RNNs. These networks are capable of learning long term dependencies in a sequential data, which makes them suitable for task such as time series forecasting. IT contains a memory cell, which is a container that can hold information for a long period of time. This memory cell is controlled by the three gates that are input gate, output gate and forget gate. These gates are responsible on managing the information on what should be add, remove to and output from the memory cell. The input gate controls what information is to be added to the memory cell. The forget gate controls what information is removed from the memory cell and the output gate controls what information is output from the memory cell. The LSTM architecture has a chain structure that contains four neural networks and different memory blocks called cells. The information is been retained by the cells and the memory manipulations are done by these gates. In mathematically this gate is computed with the help of a sigmoid activation function

5. EVALUATION

Finalizing the result of Random Forest Classifier we have taken the parameters of closing price of bitcoin as a feature to predict the future market price of bitcoin. We have also taken the sentiment analysis of bitcoin as a negative and positive sentiment to perform a better prediction if any unplanned event occurs. This performed well but not as good as it should be performed. The accuracy over the past prices

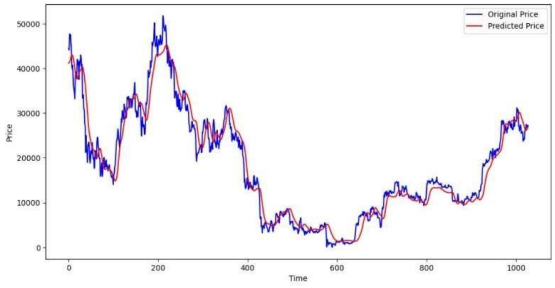
Performed well. The accuracy was 51% which was under performed by the algorithm.

The next model that we trained was the XGBoost model. We have taken the same parameters of closing price of bitcoin as a feature to predict. Comparing with the first model i.e. Random Forest Classifier it over the past prices was 60% as compared to the random forest. To provide more accuracy over the data we have done the back testing of the data within this model. We have taken 15 days of interval of the data to perform the back testing, the difference came of 5% after performing the back testing of the data. As compared to Random Forest Classifier with performed well in predicting the future prices.

The last model that we trained was the LSTM model. This LSTM performed very well compare to the rest two models. Here also we have taken the closing price of bitcoin as a parameter to perform the training of the model. To provide a better analysis we have taken past 100 and 200 days of mean averages. After splitting in testing and training part of 7:3 ratio we have added the parameters of past 100 days of mean averages to the training part. We have then trained the model with the epoch of 50 i.e. it will train the model 50 times to provide an accurate result. If we would train the model above 50 times it would than overfitted in the data. The model would not provide accurate result on the other data and if less than 50 than more overly the model

would be underfitted. After training of the model, we tested it with the other 30% data, and it performed highly accurate than the other two random forest and xgboost which provided the accuracy of 95%. In regards to long term predictions it has been observed that the LSTM outperformed both the other algorithms.

Random Forest Classifier	51%
XGBOOST Classifier	60%
LSTM	95%



6. CONCLUSION

In conclusion the use of machine learning algorithms in python for prediction of cryptocurrency prices is a significant promising area of research; while employing machine learning models to this end can provide valuable insights into the volatile nature of cryptocurrency markets and help in forecasting price trends as well as making more informed investment decisions, it is important to recognize that predicting cryptocurrency prices has some built-in difficulties and uncertainties which includes market sentiment regulatory developments and technology changes however there are several limitations associated with the study including market sentiment regulatory developments and technological advancements thus more studies should be carried out in order to get better understanding about how these models behave

REFERENCE

1."Bitcoin Price Index - Real-time Bitcoin Price Charts", Coin Desk Available: https://www.coindesk. com/price “

2. “A. Ng, "Linear Regression with Multi Variable", Stanford, CA”

3. “D. Nelson, A. Pereira and R. de Oliveira, "Stock Market’s Price Movement Prediction with LSTM Neural Networks", in Neural Networks (IJCNN).

4.“M. Dixon, D. Klabjan and J. Bang, “Classification-based Financial Markets Prediction using Deep Neural Networks", Illinois Institute of Technology.” https://doi.org/10.1162/neco.1997.9.8.1735

5. S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780.”

6. D. Shah and K. Zhang, "Bayesian regression and Bitcoin", Massachusetts Institute of Technology.”

7. I.Georgoula, D. Pournarakis, C. Bilanakos, D. Sotiropoulos and G. Giaglis, "Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices".

8. Babitha, D., Ismail, M., Chowdhury, S., Govindaraj, R., & Prakash, K.B. (2020). Automated road safety surveillance system using hybrid cnn-lstm approach. International Journal of Advanced Trends in Computer Science and Engineering, 9(2), 1767-1773. doi:10.30534/ijatcse/2020/132922020