BITCOIN CRYPTO CURRENCY PREDICTION USING MACHINE LEARNING

Mrs. Joshi Padma Narasimhachari [1],B Sharon Angel[2] ,B.Bindu[3], CH.Supriya[4],
[1] Asst. Professor in Sreyas institute of engineering and Technology, JNTUH, India,padmajoshi@sreyas.ac.in [2] B.Tech in Sreyas Engineering college, [3] B.Tech in Sreyas Engineering college ,[4] B.Tech in Sreyas Engineering college.

ABSTRACT

The goal of this paper is to determine how well the Bitcoin volume per USD can be predicted. The Bitcoin Price Index contains price information. The work is accomplished to varying degrees of success by employing the Bayesian optimised recurrent neural network (RNN) and the Long Short Term Memory (LSTM) network. LSTM achieves a maximum accuracy of 52% and an RMSE of 8%. The popular ARIMA model for time series is used to compare with in-depth learning models. In-depth offline learning methods outperform ARIMA's poor performance forecast, as expected. Finally, both in-depth learning models are marked on both GPU and CPU, with GPU training time improving CPU implementation by 67.7%. Bitcoin, Deep Learning, GPU, Recurrent Neural Network, Long-Term Memory, ARIMA are index terms.

1. INTRODUCTION

Bitcoin is the most valuable cryptocurrency in the world and is traded in more than 40 exchanges globally accepted in more than 30 different currencies. It has a current market capitalization of 9 billion USD according to https://www.blockchain.info/ and sees more than 250,000 transactions taking place per day. Like currency, Bitcoin offers a new opportunity to predict price due to its small age and which has led to its non-existence, which is much higher than fi in finance. It is also unique in relation to traditional fi in the nature of money in its open nature; no complete data is available for transactions or cash-in-transactions. Forecasts of mature financial markets such as the stock market have been studied for a long time. Bitcoin presents an interesting similarity to this as it is a problem to predict a series of time in the market that is in its last phase. Methods of predicting traditional time series such as Holt-Winters exponential smooth models are based on precise predictions and require data that can be reverted to practice, season and sound in order to work. This type of method is best suited for a task such as sales forecasting when the results of the season are available. Due to the temporary shortage of Bitcoin markets and its high volatility, these methods do not work well in this field. Given the complexity of the task, in-depth learning makes for an interesting technical solution based on its application in the same areas. Repetitive neural network (RNN) and short-term memory (LSTM) are more popular than traditional multilayer perceptron (MLP) due to the Bitcoin data structure. The purpose of this paper is to investigate the accuracy of the Bitcoin price that can be predicted using learning tools and to compare the simulation methods performed in many key areas and GPUs. This paper offers the following: of the approximately 653 papers published in Bitcoin, only 7 (at the time of writing) are related to machine learning predictions. To facilitate comparisons of many traditional approaches to financial forecasting, the ARIMA time series model is also designed to compare performance with neural network models. An independent variation in this study is the closing price of Bitcoin per USD taken from the Coin-desk Bitcoin Price Index. Instead of focusing on a single exchange, we take the price between the major Bitcoin exchanges: Bit-stamp,

Bit fi nex, Coin-base, OkCoin and it-Bit. If we were to use brand-based trading it would be helpful to focus on one exchange. To test the performance of the models, we use the square root meaning (RMSE) of the closing price and add the estimated values to the divisive outcomes of the categories: price increase, decrease or no change. This latest step allows additional performance metrics that can be useful to a trader in developing a trading strategy: group accuracy, specific city, sensitivity and accuracy. The variations that depend on this paper are from the Coin-desk website, and Block-chain.info. In addition to closing price, opening price, daily high and daily low are included along with Block-chain data, eg mining weight and hash rate. Featured features (considered technical analytical indicators) include two simple moving averages (SMA) and a fixed closing price.

2. LITERATURE SURVEY

Bitcoin pricing research using machine learning capabilities is especially lacking. Use a hidden source model as it was developed to predict the Bitcoin price which marks 89% return in 50 days with a Sharpe 4.1 rating. There is also the task of using text data from social media and other sources to predict Bitcoin prices. investigate emotional analysis using vector ma support chines in line with the frequency of Wikipedia views, and the level of network hash. investigate the relationship between the price of bitcoin, tweets and Bitcoin views on Google Trends. they used the same method without predicting the value of Bitcoin they predicted trading volume using Google Trends views. However, one limitation of these studies is usually the small size of the sample, as well as the incorrect inclusions that spread through various (social) media channels such as Twitter or message boards such as Reddit, which are consumable / consumer prices. In parts of Bitcoin exchanges are very limited. As a result, the market is facing a high risk of fraud. For this reason, feelings from social media are not considered continuously analyzes Bitcoin Block-chain to predict Bitcoin pricing using vector support (SVM) and artificial neural networks (ANN) that accurately mark the price target of 55% with standard ANN. They concluded that there was only so much speculation in Block-chain data only. also used Block-chain data, using SVM, Random Forests and Binomial GLM (standard line model) which noted the accuracy of the forecast by more than 97% but without ensuring that their models reduced the occurrence of their results. Wavelets are also used to predict Bitcoin prices, by noting the positive correlation between search engine views, network hash rating and mining difficulty with Bitcoin price. Building on these results, data from Block-chain, which is the hash and difficulty level included in the analysis as well as data from major exchanges provided by Coin-Desk.

3. METHODOLOGY

This paper follows the CRISP data mining methodology.1 The motivation for CRISP-DM over the more traditional KDD [26] revolves around the business setting of the prediction task. The dataset Bitcoin dataset used, ranges from the 19th of August 2013 until the 19th of July 2016. A time series plot of this can be seen in Figure 1. Data from previous to August 2013 has been excluded as it no longer accurately represents the network. In addition to the Open, High, Low, Close (OHLC) data from CoinDesk, the diffificulty and hash rate are taken from the Blockchain. The data was also standardised to give it a mean of 0 and standard deviation of 1. Standardisation was chosen over normalisation as it better suits the activation functions used by the deep learning models. Figure 1: Decomposition of the Bitcoin Time Series Data

A. Feature Engineering and Feature Evaluation

Feature engineering is the art of extracting useful patterns from data to make it easier for machine learning models to perform their predictions. It can be considered one of the most important parts of the data mining process in order to achieve good results in prediction tasks [27], [28]. Several papers in recent years have included indicators including the Simple Moving Average (SMA) for machine learning classification tasks [29], [30]. An example of an appropriate technical indicator is a SMA recording the average price over the previous x

days, and is correspondingly included. 1CRISP-DM 1.0: ttps://www.te-modeling-agency.com/crisp-dm.pdf 340To evaluate which features to include, Boruta (a wrapper built around the random forest classification algorithm) was used. This is an ensemble method in which classification is performed by voting of multiple classifiers. The algorithm works on a similar principle as the random forest classifier. It adds randomness to the model and collects results from the ensemble of randomised samples to evaluate attributes and provides a clear view on which attributes are important [31]. All features were deemed important to the model based on the random forest, with 5 day and 10 days (via SMA) the highest importance among the tested averages. The de-noised closing price was one of the most important variables also.

B. Deep Learning Models

Appropriate design of deep learning models in terms of net work parameters is imperative to their success. The three main options available when choosing how to select parameters for deep learning models are random search, grid search and heuristic search methods such as genetic algorithms. Manual grid search and Bayesian optimisation were utilised in this study. Grid search, implemented for the Elman RNN, is the process of selecting two hyperparameters with a minimum and maximum for each. One then searches that feature space looking for the best performing parameters. This approach was taken for parameters which were unsuitable for Bayesian optimisation. This model was built using Keras in the Python programming language [32]. Similar to the RNN, Bayesian optimisation was chosen for selecting LTSM parameters where possible. This is a heuristic search method which works by assuming the function was sampled from a Gaussian process and maintains a posterior distribution for this function as the results of different hyperparameter selections are observed. One can then optimise the expected improvement over the best result to pick hyperparameters for the next experiment [33]. The performance of both the RNN and LSTM network are evaluated on validation data with measures to prevent overfitting. Dropout is implemented in both layers, and we automatically stop model training if its validation loss hasn't improved in 5 epochs.

4. IMPLEMENTATION

A *RNN*

The first parameter to consider was the temporal length window. As suggested by supporting literature [34] these types of networks may struggle to learn long term dependencies using gradient based optimization. An auto correlation function (ACF) was run for the closing price time series to assess the relationship between the current closing price and previous or future closing prices. While this is not a guarantee of predictive power for this length, it was a better choice than random choice. Closing price is correlated with a lag of up to 20 days in many cases, with isolated cases at 34, 45 and 47 days. This led the grid search for the temporal window to test from 2 to 20, 34, 45 and 47 days. To ensure a robust search, larger time periods of up to 100 days were also tested in increments of five. The most effective window temporal length was 24. In addition to the temporal window, some hyper parameters also need tuning: Learning rate is the parameter that guides stochastic gradient descent (SGD), i.e. how the network learns. Similarly, momentum updates the learning rate to avoid the model falling into local minima (in terms of error) and attempts to move towards the global minimum of the error function [35]. We used the RMS prop optimiser to improve on SGD, as it keeps a running average of recent gradients and as a result is more robust against information loss [36]. According to Heaton [37], one hidden layer is enough to approximate the vast majority of non-linear functions. Two hidden layers were also explored and were chosen as they achieved lower validation error. Heaton also recommends for the number of hidden nodes to select between the number of input and output nodes. In this case, less than 20 nodes per layer resulted in poor performance. 50 and 100 nodes were

tested with good performance. However, too many nodes can increase the chances of overfitting, and significantly increase the time needed to train the network. As 20 nodes performed sufficiently well this was chosen for the final model. An activation function, a nonlinear stepwise equation that passes signals between layers, is also needed. The options explored were Tanh, ReLu, and Sigmoid. Tanh performed the best but the differences were not significant. The final parameters for selection are batch size and number of training epochs. Batch size was found to have little effect on accuracy but considerable effect on training time when using smaller batches in this case. The number of epochs tested ranged from 10 to 10000, however, too many training epochs can result in overfitting. To reduce the risk of overfitting, dropout was implemented as discussed above. Optimal dropout between 0.1 and 1 was searched for both layers with .5 dropout the optimal solution for both layers. A Keras callback method was also used to stop the training of the model if its performance on validation data did not improve after 5 epochs to prevent overfifitting. Generally, the RNN converged between 20 and 40 epochs with early stopping.

B. *LSTM*

In terms of temporal length, the LSTM is considerably better at learning long term dependencies. As a result, picking a long window was less detrimental for the LSTM. This process followed a similar process to the RNN in which autocorrelation lag was used as a guideline. The LSTM performed poorly on smaller window sizes. Its most effective length found was 100 days, and two hidden LSTM layers were chosen. For a time series task two layers is enough to find nonlinear relationships among the data. 20 hidden nodes were also chosen for both layers as per the RNN model. The Hyperas library2 was used to implement the Bayesian optimisation of the network parameters. The optimiser searched for the optimal model in terms of how much dropout per layer and which optimizer to use. RMSprop again performed the best for this task. In the LSTM model, activation functions weren't 2Hyperas:ttps://gitub.com/maxpumperla/hype as modified as LSTM has a certain sequence of tanh functions and sigmoid activation of different gates within the cell. LSTM models converged between 50 to 100 epochs in pre-existing conditions. Like RNN, batch size was found to have a greater influence on performance time than accuracy. This may be due to the small size of the database.

C. *Model Comparisons*

The confusion matrix representing the true / false and positive / negative collection is used to determine the metrics for measurements. Accuracy can be defined as the total number of well-planned predictions (price increases, decreases, and no change). Fighting class inequality (the price of bitcoin goes up a lot) metric sensitivity, specific city and accuracy are also analyzed. Sensitivity shows how good a model is in getting good. One city represents how good the model is in avoiding false alarms. Finally, accuracy indicates how many well-planned predictions were appropriate. Root Mean Square (RMSE) error is used to check and compare the reversal accuracy. To perform the model testing tool, 80/20 20 validation verification strategy is used. To facilitate the comparison of in-depth learning methods with the many traditional methods we have developed (and developed) the ARIMA model, as it has been widely used in price prediction problems (e.g. [38], [39]). ARIMA forecast is made by dividing data 5 times and predicting 30 days in the future. Data were segmented before acquiring several ARIMA models. Best for auto.arima from R forecast package.

5. EVALUATION

As can be seen in Table I, LSTM achieved the highest accuracy while the RNN achieved the lowest RMSE. The ARIMA prediction performed poorly in terms of accuracy and RMSE. Upon analysis of the ARIMA forecast, it predicted the price would gradually rise each day. There were no false positives from the model. One reason for this may be due to the class imbalance in predictive portion of the

ARIMA forecast (the price tends to always increase). This contributed to the specificity and precision being so high (specificity, precision= 100%). This does not necessarily suggest good overall performance, but rather that it does a decent job at identifying price direction change(s). used VGG19 and ResNet50. This VGG19 and ResNet50 both are trained Convolution Neural Network (CNN) where 19, 50 represents layers in the ConvNet[8]. To implement the transfer learning we follow 2 major steps: Chose the pre-trained model accordingly: We need to choose which pre-trained model best suits our model. The problems must belong to a similar kind. For example for Image classification we use Image Net trained VGG19 which has several images [3] Restructure your CNN,after choosing the pre trained model we freeze all the layers and remove the last fully connected layer to add custom layers related to our problem. We extract the necessary bottleneck features for our mode. It helps us build more accurate models while saving time. For optimizing w have chosen SGD optimizer and adam optimizer. SGD maintains the same learning rate throughout the training for all the parameters, adam (Adaptive Moment Estimation) updates the learning rates accordingly [6]. A Each CNN is divided into a convolution base and a classifier. Whenever we what to use this as the pre trained model we just have to remove the last fully-connected classifier and build our own classifier.

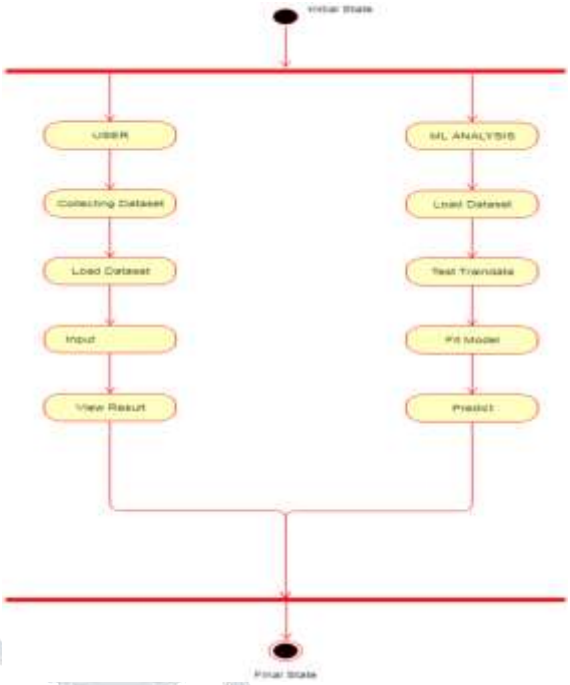

Fig 5.1: Activity Diagram.

A Each CNN is divided into a convolution base and a classifier. Whenever we what to use this as the pre-trained model we just have to remove the last fully-connected classifier and build our own classifier.

6. SYSTEM ARCHITECTURE

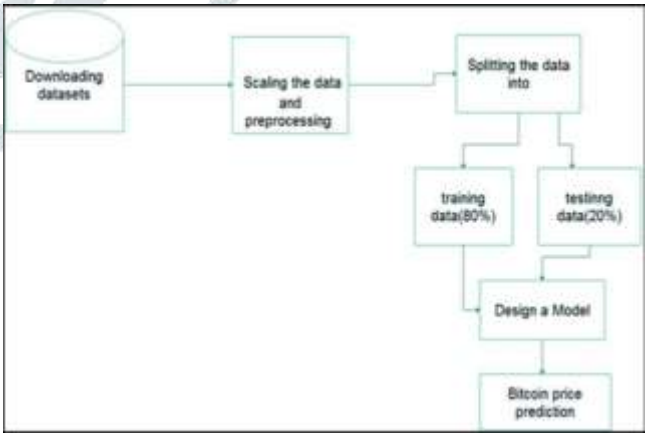

Fig 6.1: Architecture diagram

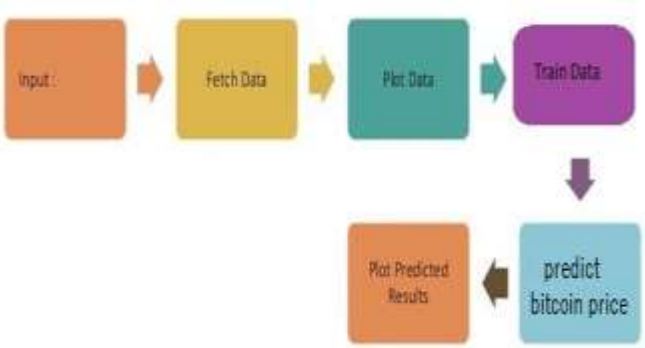

Fig 6.2: Data Flow Diagram

A system architecture fig 5.1 diagrams would be used to show the relationship between different components. Usually they are created for systems which include hardware and software and these are represented in the diagram to show the interaction between them. However, it can also be created for web applications. The Modules used

i) NumPy is a general-purpose array-processing package. It provides tools for working with these arrays.

ii) MatPlotLib: It is a plotting library used for graphics in python. It can be used in python scripts, shell, web applications.

iii) Sklearn: It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports libraries like NumPy and SciPy.

7.RESULTS

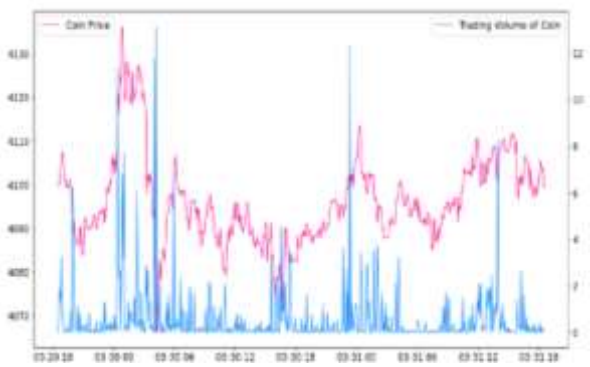

Fig 7.1: Bitcoin Data Graph

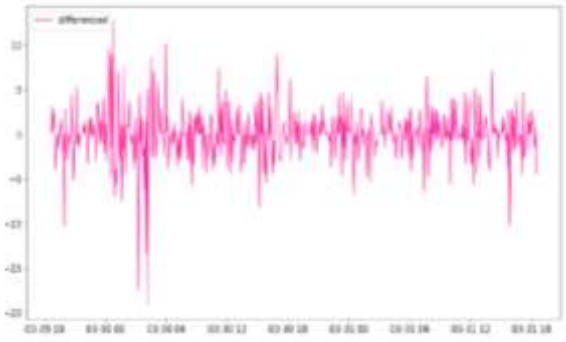

Fig 7.2: Difference

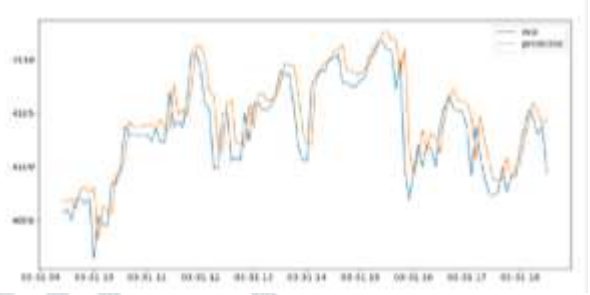

Fig 7.3: Prediction vs Actual

8. CONCLUSION

Deep learning models such as the RNN and LSTM are evidently effective for Bitcoin prediction with the LSTM more capable for recognising longer-term dependencies. However, a high variance task of this nature makes it difficult to transpire this into impressive validation results. As a result it remains a difficult task. There is a fine line between overfitting a model and preventing it from learning sufficiently. Dropout is a valuable feature to assist in improving this. However, despite using Bayesian optimisation to optimize the selection of dropout it still couldn't guarantee good validation results. Despite the metrics of sensitivity, specificity and precision indicating good performance, the actual performance of the ARIMA forecast based on error was significantly worse than the neural network models. The LSTM outperformed the RNN marginally, but not significantly. However, the LSTM takes considerably longer to train.

Future Enhancements:

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are: As the technology emerges, it is

possible to upgrade the system and can be adaptable to desired environment. Based on the future security issues, security can be improved using emerging technologies like single sign-on.

REFERENCES

[1] Peter Mel and Tim Grace, "The NIST Definition of Cloud Computing", NIST, 2010.

[2] Achill Buhl, "Rising Security Challenges in Cloud Computing", in Proc. of World Congress on Information and correspondence Technologies ,pp. 217-222, Dec. 2011.

[3] Srinivasarao D et al., "Breaking down the Superlative symmetric Cryptosystem Encryption Algorithm", Journal of Global Research in Computer Science, vol. 7, Jul. 2011.

[4] Tingyuan Nye and Tang Zhang "An investigation of DES and Blowfish encryption algorithm‟ , in Proc. IEEE Region 10 Conference, pp. 1-4 ,Jan. 2009.

[5] Jitendra Singh Adam et al.," Modified RSA Public Key Cryptosystem Using Short Range Natural Number.