Comparative Study of Prediction Models in Bitcoin Price Forecasting and Resolve of The Bounds Limitation

Chenjie Yang , Elarbi Amraoui , Zepu Wang
cy141@duke.edu, ea137@duke.edu, zw163@duke.edu

I) Abstract

Due to the relative youth of cryptal currency, there is a lack of research in forecasting its price. Many machine learning models are applied to make such predictions. However, each of those applied models have their hidden limitations. In this research, we attempt to uncover limitations of some common ones (Linear Regression, Decision Tree Regression, Random Forest, XGBOOST, Support Vector Machines and Recurrent Neural Networks) in the context of bitcoin price forecasting. Bearing those limitations in mind, investors can make better informed decisions. In this research, grid search was used to optimize the performance of each non-computationally expensive model and it successfully uncovered some of those limitations including the "Bounds problem" of tree models and the inability to predict sudden changes, etc. The research also proposes an updated method based on an altered version of the arc formula to enhance the predictions.

II) Introduction

Cryptocurrencies are electronic and decentralized alternatives to government-issued money, with Bitcoin as one of the best-known examples. In recent years, cryptocurrencies have been considered lucrative investments. The total market value of the cryptal currency's circulating supply has reached more than one trillion dollars ("Cryptocurrency Market Capitalizations "CoinMarketCap" 2010), for reference, the GDP of the USA, as of 2020, was $20.93 trillion ("Gross Domestic Product, 4th Quarter and Year 2020 (Advance Estimate) U.S. BEA" 2021). The current market cap of bitcoin makes up to twenty percent of the US GDP, this alone is a strong incentive to forecast bitcoin prices. However, literature failed to compare them, find their limitations, and solve their respective issues. This research covers that using a comparative method. The dataset used includes historical bitcoin prices from January 1st, 2012 to December 31st, 2020 within a minute interval. Each model was studied based on selected metrics. Overall, when dealing with different situations in investment, a highly dynamic environment, it is imperative to know which model to apply. In this case, learning about the limitation of each model becomes extremely valuable. In this research, the main focus was the bounds problem which naturally occurs in tree algorithms. It also occurs due to normalization techniques that set an upper and lower bound in the input space, it, consequently, affects models that require normalization such as SVMs and LSTMs. We proposed a solution using an altered version of the arc formula. Detailed methodology will be introduced later in this research. The remainder of this research is organized as follows; *Literature Review* will summarize recent achievements in related area, *Proposed Method* will introduce all models in detail along with the evaluation method, *Performance Evaluation* will evaluate the models' performances, and the finally, we will propose some final work.

III) Literature Review

Bitcoin price forecast is a relatively new topic, in other words, the related literature work that is directly related to it is, therefore, limited. Treating this problem as a general financial time series problem opens up more resources to our disposition. Among them, stock price forecasting is a well-developed field; its methods were used as the main foundations of our research.

Linear regression, often considered as the standard of comparison with more advanced models, uses a simplistic algorithm. Its use in stock predictions, however, is not very promising due to the linearity assumptions (Altay, Satman, and Hakan 2005). Support vector regression (SVR) is another widely used method; however, it is rarely implemented directly to predict stock market prices and is often combined with other machine learning techniques such as artificial neural networks to improve its accuracy (Patel et al. 2015). We opted to use SVR, with no ulterior help, to analyze its performances and limitations. To improve upon the traditional decision trees, researchers implemented random forest, XGBOOST and other techniques which improve the accuracy of the predictions (Basak et al. 2019). Those models have gained traction within the data science community, as they are increasingly cited in price forecasting articles (Ng 2021). Recurrent Neural Networks (RNNs) have also gained popularity in time series analysis. LSTM, a type of RNN, tackles the vanishing gradient issue that recurrent networks suffer from when dealing with long data sequences; it is, consequently, a good way of manipulating stock prices (Ghezelbash 2012). Further analysis of these methods would be included in section *Proposed Method*.

IV) Proposed Method

1) Flow Chart

In the chart below, we will illustrate the milestones of our research process. We conduct a related literature review, study relevant models, preprocess the data, and, finally, provide empirical results on the tested models using both the traditional and the newly proposed method to display its enhanced performance.

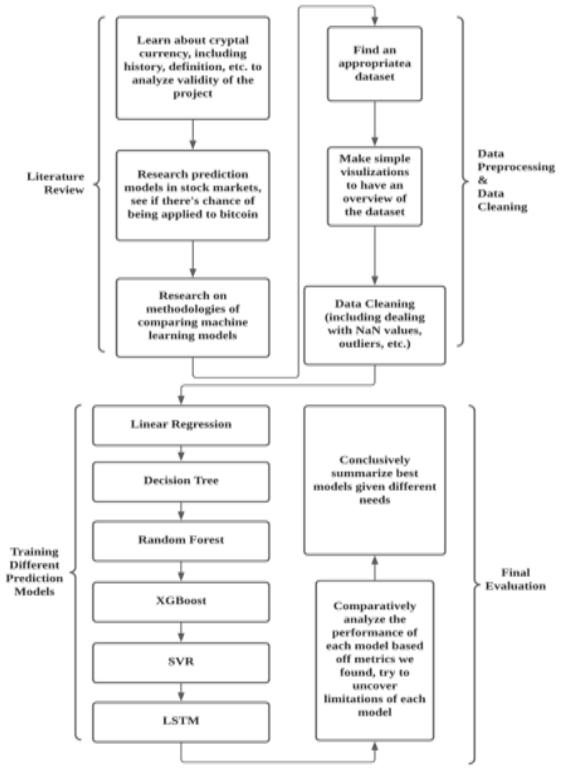

Figure 1

2) Model Description

This sub-section will introduce all the implemented models.

A) Decision Tree Regression (DTR)

Decision trees, whether it be classifier, regressor, or else, are mainly composed of a root node that leads to subsequent decision nodes, which in turn lead to two or more leaf nodes. A general rule of thumb is that, assuming the model isn't overfitting, the more nodes the better the accuracy and the worse the comprehensibility and readability. J. R. Quinlan introduced few methods to simplify decision trees, making them more intelligible, without extensively compromising their accuracy. Quinlan covered a list of four non-exhaustive methods, one of them was "reduced error pruning" which, put simply, looks for sub-trees that share the same properties with other existent sub-trees and turns them into leaf nodes, some of its handicaps, however, is that it requires a separate test set and do not take into consideration the special cases not seen in the testing data (Quinlan 1987).

B) Random Forest (RF)

Leo Breiman first proposed random forest as a general-purpose and noise-robust classification and regression algorithm. It works by putting together an ensemble of randomized decision trees and aggregating their predictions using majority voting (polling), weighted voting, simple averaging, or another ensemble combining methods (Zhang 2017). The random Forest algorithm can be applied to a variety of prediction problems and is especially thriving, compared to other models, in lack of observations in contrast to variables (Biau and Scornet 2016). Unlike the popular belief, more decision trees do not necessarily imply better predictions, it might imply, however, unnecessary wasted computational power (Oshiro et al. 2012). Alternatively, if one has access to more computational power, grid search is an adequate technique for finding the best number of trees among a given set.

C) XGBOOST

The XGBOOST algorithm was developed as a research project at the University of Washington (Chen and Guestrin 2016). Boosting is considered to be one of the most powerful ensemble methods introduced in the last two decades. Originally designed for classification problems, it was profitably extended to regression. It combines the outputs of many "weak" classifiers to produce a single powerful "committee" (Hastie, Tibshirani, and Friedman 2004). It has also been extensively used in price forecasting in scientific articles and amateur's notebooks.

D) Linear Regression (LR)

Linear Regression is the base model in machine learning while assuming linear relationship. It is simple enough to provide an adequate and interpretable description of how the inputs affect the output, and surprisingly, in terms of prediction, they can outperform fancier nonlinear models (Hastie, Tibshirani, and Friedman 2004).

E) Support Vector Machine (SVR)

A version of Support Vector Machine was first proposed by Drucker· et al. in 1996. In most linear regression models, the goal is to minimize the sum of squared errors. (GOLDBERGER 1964) SVR provides the flexibility to define an acceptable range and finds the appropriate line (hyperplane in higher dimensions) to fit the data. In contrast to ordinary least square, the objective function of SVR is to minimize the L2-norm of the coefficient vector. The error is handled by the constraints of the optimization problem, where the absolute error is set to be less than or equal to a specified maximum error, ε. The value of ε can be tuned according to the accuracy level of our choosing. The optimization problem is explicitly defined as:

$$MIN\ \frac{1}{2}\|w\|^2.$$
$$S.T.\ |y_i - w_i x_i| \leq \varepsilon.$$

F) Long Short-Term Memory (LSTM)

RNN is a very popular neural network structure in deep learning where neurons can accept the output from both the previous and current layer. This creates a so-called "short-term memory". This recurrent structure has, therefore, the potential to solve the time related modeling tasks (Elman 1990). The newly introduced multi-layer RNN can capture more abstract features among the dataset (Schaefer, Udluft, and Zimmermann 2008). To enhance the long-term memory ability of the repeat module in RNNs, Long-Short-Term Memory (Kandel 2009) were introduced. The structure of the repeating model in the LSTM are shown in Figure 2. The key feature distinguishing LSTMs is that there are three gates, compared to the GRU which has two.

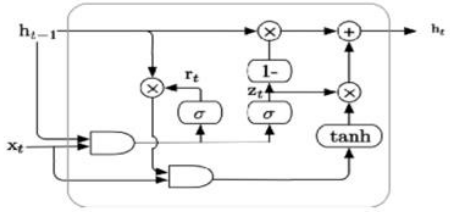

Figure 2

3) Data Description and Preprocessing

The Bitcoin Historical data contains a total of 8 features (Timestamp, Open, High, Low, Volume Bitcoin, Volume Currency, and Weighted Price) and over four million examples from which one million is missing. The examples represent the values of each feature within one-minute intervals starting from January 2012 to March 2021. The Weighted Price of bitcoin started fluctuating after 2017, it was, hence, agreed upon to discard any value before then. Considering the values from 2017 onwards, the dataset still includes missing data, dropping them will render the previously continuous time series into a discrete one which might impede on the predictions. Multiple imputation methods were considered, such as Bayesian inference, the use of a proxy variable, and the use of simple neighboring value, etc. the use of spline interpolation and one-dimensional interpolation minimized the MSE for the features which display a strong trend but weak seasonality, while the average of the last observation carried forward (LOCF) and the next observation carried backward (NOCB) minimized it for the remaining features which display weak trend and seasonality (*volume_(BTC)* and *volume_(Currency)*).

4) Training Requisites

Before training a model, certain prerequisites should be met. A metric and training method need to be defined. The Feature choice is also important, Pearson correlation was used as an indicator, and it shows that all but two features (*volume_(BTC)* and *volume_(Currency)*) are highly correlated with the weighted price. After extensive testing, it became clear that the poorly correlated features only slightly improve the results while significantly increasing the training time, they will be discarded. Furthermore, the option to use the weighted price alone instead of the other miscellaneous features is also tempting. In view of the fact that our main goal is to compare the existing models and uncover their limitations, and that our resources are limited, training RNNs with multiple features is not doable. It has, consequently, been decided to use the price alone for all the models, including the less computationally expensive algorithms (decision trees, XGBOOST, linear regression, and SVR), even if using all the features improves the results (their values will, however, will still be included).

A) Training Method

In this paper, we are going to treat this problem as a financial time series analysis problem.

With $X = [X_{t_1}, X_{t_2}, \dots, X_{t_n}]$ and $Y = [Y_{t_1}, Y_{t_2}, \dots, Y_{t_n}]$, where X_{t_i} represents the input at t_i , Y_{t_i} represents the bitcoin price at t_i, and $[t_1, t_2, \dots, t_n]$ represent the continuous, evenly spaced time series. We can choose k continuous inputs to predict the next price, which is equivalent to:

$$f(X_{t_a}, \dots, X_{t_{a+k-1}}) = \widehat{Y_{t_{a+k}}}.$$
$$with\ 1 < k\ < n-1\ ;\ a \geq 1$$

In this case, f represents the prediction methods and $\widehat{Y_{t_{a+k}}}$ represents the predicted value.

Within the existing implementation, $k$is set to 1 by default but its value can be altered by passing the parameter $--shift\_param$.

B) Evaluation Metrics

The significance of a model is defined by the gap or distance between the predicted and actual price. Some of the best metrics that achieve it are MSE and MAE, the difference between the two is that MAE doesn't consider the direction of the error. The R^2 metric will also be included for reference.

$$\text{MAE} = \frac{1}{n}\sum |y_i - \hat{y}_i|.$$
$$\text{R}^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}.$$
$$\text{MSE} = \frac{1}{n}\sum(y_i - \hat{y}_i)^2.$$

Wherey_i represents the real data, $\hat{y}_i$ represents the predicted data, and $\bar{y}$ represents the average real data.

5) Model Preparation

Independently of the chosen model, there are multiple parameters available; $--price\_only$, True by default, works as a switch that defines whether to use all the highly correlated features or the price only; $--shift\_param$, 1 by default, refers to the number of past examples to consider (referred to as k above). For instance, if set to 3, the model will require the previous three instances to predict the future price. There are more options available, a more detailed documentation can be accessed by running the following prompt on Linux or Mac *python main.py -h* and *py main.py -h* on windows.

A) Classical Models

Two classic algorithms were used: Linear Regression and SVR. There was no normalization for LR because there is no set upper and lower limit for bitcoin prices and imposing them will impede on the results. As for SVR, it won't be able to train efficiently without normalization, it is necessary to use it. (Shanker, Hu, and Hung 1996):

For a value X,

$$X_{std} = \frac{(X - X_{min})}{(X_{max} - X_{min})}.$$

Where X_{max} and X_{min} represent the maximum and minimum value, respectively.

B) Tree Models

Three tree algorithms were used: random forests, XGBOOST, and decision tree regressor. Grid search was used to find the best tuning parameters. The results are depicted below.

Model	RF	XGBOOST	DTR
Max_depth	15	7	18
min_samples_leaf	1	N/A	N/A
min_samples_split	7	N/A	N/A
n_estimators	50	100	N/A
alpha	0	10	N/A
Learning Rate	N/A	0.1	N/A
max_leaf_nodes	N/A	N/A	50

Table 1

C) Modern Models:

Using a complex LSTM structure might improve the results compared to a one-layered LSTM. However, due to the limited computing resources, it is impossible to try multiple layers and multiple epochs in order to tune the parameters and find the best structure. Therefore, we only trained the LSTM with one layer and 10 epochs.

V) Performance Evaluation

1) Traditional Method:

Traditional method refers to preprocessing using bounded normalization, a bound inducing algorithm, or no normalization.

A) *Tree Methods:*

For computationally cheap tree models, namely, decision tree regressor and XGBOOST, an extensive grid search method was used to find the best set of parameters and analyze the variations of the predictions with regards to changes in parameters. For random forests that are relatively more expensive in terms of our computational resources, we limited the search based on the previous results. The best set of parameters will be highlighted and followed by an analysis of the changes based on the train_size, price_only, and shift_param.

(a) General Results of the Decision Tree Regressor:

Price only	Shift param	Train size	MSE F	**MSE T**	MAE F	**MAE T**	R2 F	**R2 T**
False / **True**	1	0.8	1692454.28	**1690417.51**	375.88	**375.52**	0.91	**0.91**
False/ **True**	1	0.5	680046.07	**680187.09**	204.11	**207.27**	0.96	**0.96**
False / **True**	1	0.3	492817.21	**492838.47**	175.51	**175.49**	0.96	**0.96**
False / **True**	3	0.8	1692462.31	**1690425.53**	375.88	**375.52**	0.91	**0.91**
False / **True**	3	0.5	680047.35	**680188.37**	204.11	**207.27**	0.96	**0.96**
False / **True**	3	0.3	492817.88	**492839.14**	175.51	**175.49**	0.96	**0.96**
False / **True**	7	0.8	1692478.36	**1690441.56**	375.88	**375.53**	0.91	**0.91**
False / **True**	7	0.5	680049.92	**680190.93**	204.11	**207.27**	0.96	**0.96**
False / **True**	7	0.3	492819.22	**492905.38**	175.51	**175.59**	0.96	**0.96**

Table 2

The following points can be drawn from the table above:

- Using all features instead of just the price usually slightly worsens the MAE and improves the MSE.
- Increasing the number of instances considered usually doesn't affect the MAE and improves the MSE.
- Increasing the training size worsens both the MAE and MSE.

(b) General Results of XGBOOST:

Price only	Shift param	Train size	MSE F	**MSE T**	MAE F	**MAE T**	R2 F	**R2 T**
False / **True**	1	0.8	1323405.3	**1320953.97**	244.14	**243.73**	0.93	**0.93**
False / **True**	1	0.5	528777.02	**528862.01**	101.01	**101.02**	0.97	**0.93**
False / **True**	1	0.3	377802.66	**377566.11**	73.94	**73.93**	0.97	**0.97**
False / **True**	3	0.8	1323439.87	**1323130.98**	244.19	**244.03**	0.93	**0.93**
False / **True**	3	0.5	529252.59	**529174.81**	101.08	**101.18**	0.97	**0.97**
False / **True**	3	0.3	378027.51	**378008.31**	74.01	**74.06**	0.97	**0.97**
False / **True**	7	0.8	1322923.41	**1324275.25**	244.17	**244.23**	0.93	**0.93**
False / **True**	7	0.5	529363.54	**529642.2**	101.2	**101.27**	0.97	**0.97**
False / **True**	7	0.3	378175.87	**378191.75**	74.1	**74.2**	0.97	**0.97**

Table 3

The following points can be drawn from the table above:

- Using all features instead of just the price *usually* slightly improves the MAE and worsens the MSE.
- Increasing the number of instances considered worsens both the MAE and MSE.
- Increasing the training size worsens both the MAE and MSE.

(c) General Results of the Random Forest:

Both previous tree models maximized their metrics while considering a single previous price and training on 30% of the dataset, equivalent to 631k features. Using those same parameters for the random forest results in an MSE of 375058.66, an MAE of 73.45, and an R2 of .97. As can be noted from those raw numbers, it seems that it is performing better than the previous models. Nevertheless, the results of all of those tree algorithms are quite poor, to say the least.

(d) Limitations:

Considering the top right part of figure 3:

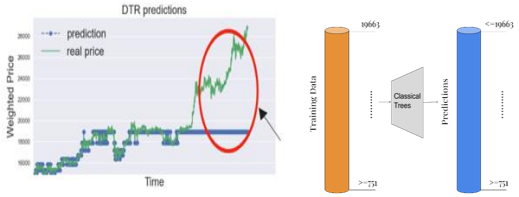

Figure 3

The predicted weighted price stagnates while the real price keeps increasing. The reason for this odd behavior is that classical trees-based models are unable to extrapolate, in other words, the range of their training data defines a floor and ceiling for their future predictions. The use of classical tree-based methods isn't the best choice when it comes to the forecast of values that do not have an upper and lower limit such as bitcoin price. We can further posit that in the case where those limits exist, it is necessary that they are part of the training data, failure to do so will result in erroneous predictions. It is also important to note that if the testing data is within the range of the training data, the predictions will be spaciously accurate, but it is a special case that one shouldn't rely on.

B) Linear Regression

(a) General Results

Price only	Shift param	Train size	MSE F	**MSE T**	MAE F	**MAE T**	R2 F	**R2 T**
False / **True**	1	0.8	353.94	**125**	6.06	**6.4**	1	**1**
False / **True**	1	0.5	175.92	**93.26**	5.05	**5.39**	1	**1**
False / **True**	1	0.3	133.29	**77.36**	4.55	**4.92**	1	**1**
False / **True**	3	0.8	327.87	**123.43**	6.07	**6.41**	1	**1**
False / **True**	3	0.5	167.3	**92.11**	5.06	**5.44**	1	**1**
False / **True**	3	0.3	127.57	**76.53**	4.57	**4.96**	1	**1**
False / **True**	7	0.8	324.03	**123.4**	6.07	**6.41**	1	**1**
False / **True**	7	0.5	169.73	**92.1**	5.05	**5.44**	1	**1**
False / **True**	7	0.3	129.12	**76.52**	4.56	**4.96**	1	**1**

Table 4

The following points can be drawn from the table above:

- Using all features instead of just the price slightly improves the MAE and worsens the MSE.
- Increasing the number of instances considered slightly worsens the MAE and slightly improves the MSE.
- Increasing the training size worsens both the MAE and MSE.

(b) Limitations

Figure 4

Looking at the MSE and MAE values, it is tempting to assume that linear regression is a decent predictor for Bitcoin prices. Looking more closely, it becomes clear that LR achieves those results because it keeps the predictions rather close to the previous price considered, and thus, the higher k, the more accurate it is. However, it is less resilient to sudden changes in price, so in case of an abrupt price increase or decrease, the LR will perform poorly. Some of those cases have been circled.

C) SVM

(a) General Results of SVR:

As mentioned in the proposed method, SVR requires normalization to train efficiently, but the price normalization engenders a similar issue as the one mentioned in the tree limitation. The normalization used was the min-max normalization with a set minimum and maximum previously extracted from the training data. The values that go beyond the pre-defined maximum (which will result in a normalized value greater than one) are set to one, similarly, the values that go lower than the pre-defined minimum are set to zero. In the provided implementation, SVR was designed to only work with the price only as a feature, future updates might include it. The general results are depicted below.

Price only	Shift param	Train size	MSE T	MAE T	R2 T
True	1	0.8	**3276567.78**	**1121.98**	**0.83**
True	1	0.5	**2107202.02**	**1045.89**	**0.86**
True	1	0.3	**2215386.49**	**1208.04**	**0.83**
True	3	0.8	**3114925.14**	**1021.57**	**0.84**
True	3	0.5	**2070605.94**	**1031.76**	**0.87**
True	3	0.3	**2188061.48**	**1201.13**	**0.83**
True	7	0.8	**3007692.84**	**964.1**	**0.85**
True	7	0.5	**2059939.05**	**1037.93**	**0.87**
True	7	0.3	**2140716.53**	**1180.95**	**0.84**

Table 5

The following points can be drawn from the table above:

- Increasing the number of instances considered improves both the MAE and the MSE.
- Using a balanced number in the training (50%) usually improves the MAE and the MSE, going above or below 50% usually worsens the results.

(b) Limitations

Figure 5

Normalization imposes an upper and lower limit on the price; the predictions can't go beyond or below a certain price level. This problem is similar to the tree limitation, but it isn't an inherent problem within the SVR.

D) RNN:

(a) General Results of LSTM:

Using a normalization similar to that of the SVM, the LSTM will output similar results to that of the tree and SVR models. For illustration purposes, the LSTM will be trained without normalization.

(b) Limitations:

Figure 6

Using a single layer and 10 epochs, the LSTM, fed with non-normalized features, outputs a constant value for all the predictions. Based on a research aiming to emphasize the importance of normalization in quantized LSTMs, we hypothesized that this behavior is either due to a gradient explosion or a saturated neutron which are direct consequences of not normalizing the data (Hou et al. 2019). It is hence necessary to use normalization for the LSTM, even if doing so will cause that upper and lower bound issue.

2) Proposed Method:

The tree algorithms and the algorithms that require normalization (LSTM, SVR) all suffer from the bounds problem, the limits of their training data define the limits of their testing data. This bounds problem results in the creation of constant values at the limits, our proposed method solves that issue altogether while also reducing the likeliness of hitting an upper or lower bound. If the bounds were to be reached, the predicted price will still increase or decrease which considerably improves the results. The proposed method is to predict on an altered version of the arc formula instead of on the weighted price. The formula used is:

$$change\ in\ price\ = \frac{future\ price\ -\ current\ price}{future\ price + current\ price}\ *\ norm.$$

If the norm is set to 100, the change in price will be equal to the percentage change, if it was set to 1000, it will be equal to the per

thousand change…The effect of the change in norm wasn't deeply considered in this research, however, it is advised to use lower norms if the interval between two observations is longer (because with greater intervals, the price changes are more drastic) and higher norms if the interval is shorter (our case) (because with smaller intervals, the price changes are less substantial).
Some of the results are depicted below. One can notice that even with a single layer on the LSTM, it produces significantly better results. The last figure depicts the improvements in terms of MSE and MAE for each model. One can notice that this updated method barely affects the results of linear regression which doesn't use normalization.

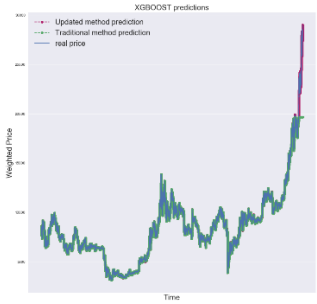

Figure 7

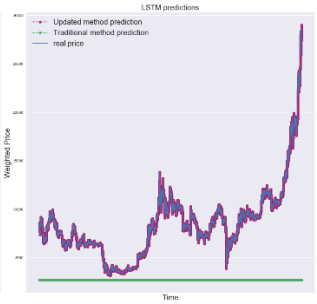

Figure 8

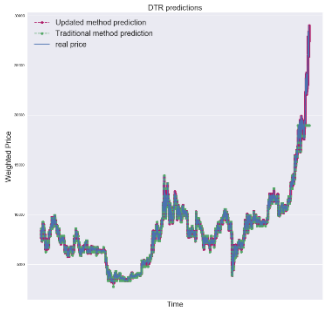

Figure 9

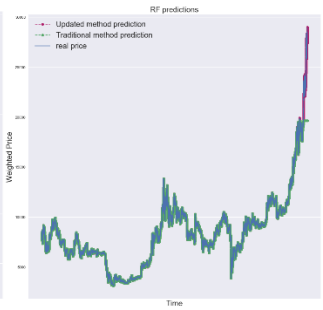

Figure 10

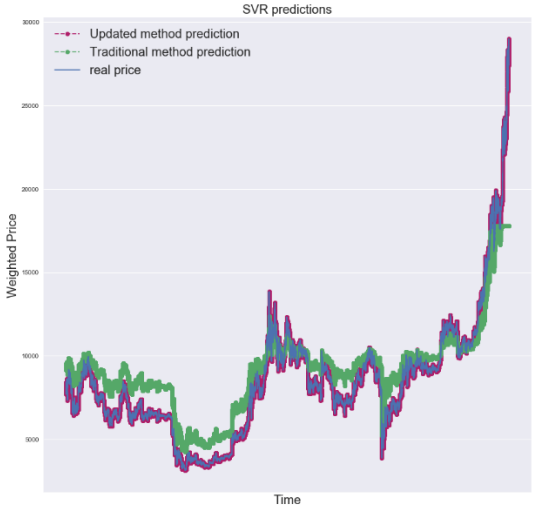

Figure 11

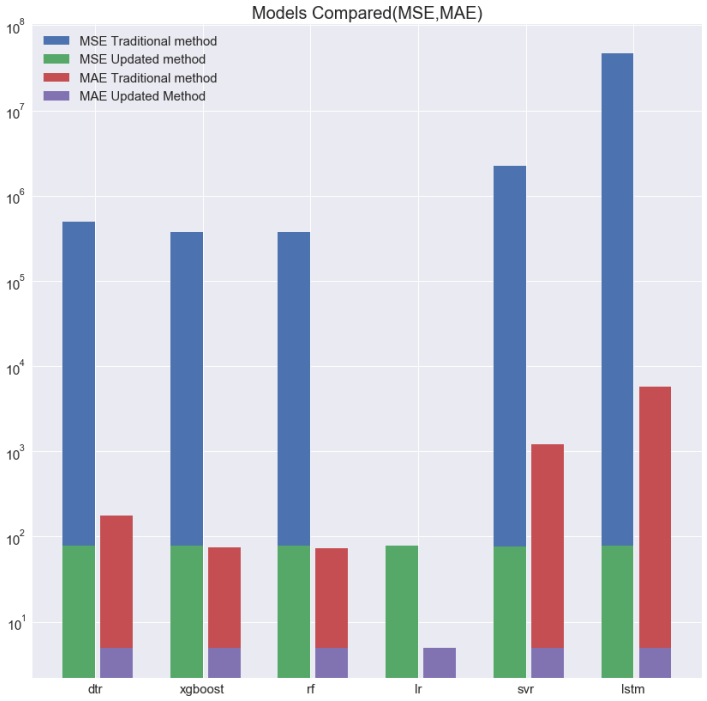

Figure 12

Within the provided implementation, the parameter $--per\_change$ can be used to choose between the traditional method (set to False) or the new proposed method (set to True). It is enabled by default.

VI)Conclusion and Future Work

In this paper, different models were applied to predict bitcoin price. All of the models covered suffered from either inherent or imposed limitations. Linear regression struggles with abrupt price fluctuations. Tree based algorithms face an inherent bounds problem, while SVRs and LSTMs face an imposed bounds problem due to the normalization. LSTMs, heavy on the CPU and GPU, also have a naturally high barrier of entry making it hard to use complex structures with limited resources. Moreover, we developed a new formula for the data manipulation, resulting in a higher performance in most models. This research can be applied to any forecasting problem whose target doesn't have existing bounds.
To further extend our work, we can study how a change in the norm affects the predictions. We can also introduce more models such as Gradient Boost and Artificial Neural Network (ANN), etc. This research did not include any other deep learning method (besides LSTM) due to the limited resources relative the dataset substantial size.

VII) References

[1] Altay, Erdinç, M Satman, and Hakan. 2005. "STOCK MARKET FORECASTING: ARTIFICIAL NEURAL NETWORK and LINEAR." Journal of Financial Management & Analysis 18 (2).

[2] Basak, Suryoday, Saibal Kar, Snehanshu Saha, Luckyson Khaidem, and Sudeepa Roy Dey. 2019. "Predicting the Direction of Stock Market Prices Using Tree-Based

Classifiers." The North American Journal of Economics and Finance 47 (January): 552–67. https://doi.org/10.1016/j.najef.2018.06.013.

[3] Biau, Gérard, and Erwan Scornet. 2016. "Rejoinder On: A Random Forest Guided Tour." TEST 25 (2): 264–68. https://doi.org/10.1007/s11749-016-0488-0.

[4] Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16. https://doi.org/10.1145/2939672.2939785.

[5] "Cryptocurrency Market Capitalizations | CoinMarketCap." 2010. CoinMarketCap. 2010. https://coinmarketcap.com/.

[6] Drucker·, Harris, Chris Burges, Linda Kaufman, Alex Smola··, and Vladimir Vapoik. 1996. "Support Vector Regression Machines." https://proceedings.neurips.cc/paper/1996/file/d38901788c533e8286cb6400b40b386d-Paper.pdf.

[7] Elman, Jeffrey L. 1990. "Finding Structure in Time." Cognitive Science 14 (2): 179–211. https://doi.org/10.1207/s15516709cog1402_1.

[8] Ghezelbash, Ali. 2012. "Predicting Changes in Stock Index and Gold Prices to Neural Network Approach." Journal of Mathematics and Computer Science 04 (02): 227–36. https://doi.org/10.22436/jmcs.04.02.12.

[9] GOLDBERGER, ARTHUR S. AUTOR. 1964. Econometric Theory. Google Books. John Wiley & Sons, Incorporated. https://books.google.com/books?id=KZq5AAAAIAAJ&pg=PA156.

[10] "Gross Domestic Product, 4th Quarter and Year 2020 (Advance Estimate) | U.S. Bureau of Economic Analysis (BEA)." 2021. Www.bea.gov. January 28, 2021. https://www.bea.gov/news/2021/gross-domestic-product-4th-quarter-and-year-2020-advance-estimate#:~:text=Current%2Ddollar%20GDP%20decreased%202.3.

[11] Hastie, Trevor, Robert Tibshirani, and J H Friedman. 2004. The Elements of Statistical Learning: Data Mining, Inference, and Prediction: With 200 Full-Color Illustrations. New York: Springer.

[12] Hein, Matthias, and O. Bousquet. 2005. "Hilbertian Metrics and Positive Definite Kernels on Probability Measures." Semantic Scholar. 2005. https://pdfs.semanticscholar.org/7822/69ba228ab3c245ea62f8da2fee141a40f486.pdf.

[13] Hou, Lu, Jinhua Zhu, James Kwok, Fei Gao, Tao Qin, and Tie-Yan Liu. 2019. "Normalization Helps Training of Quantized LSTM." Advances in Neural Information Processing Systems 32. https://papers.nips.cc/paper/2019/hash/f8eb278a8bce873ef365b45e939da38a-Abstract.html.

[14] Kandel, Eric. 2009. "The Long and Short of Long-Term Memory." GBM Annual Spring Meeting Mosbach 2009 2009 (Spring). https://doi.org/10.1240/sav_gbm_2009_m_002330.

[15] Ng, Yibin. 2021. "Forecasting Stock Prices Using XGBoost — a Detailed Walk-Through." Medium. April 23, 2021. https://towardsdatascience.com/forecasting-stock-prices-using-xgboost-a-detailed-walk-through-7817c1ff536a.

[16] Oshiro, Thais Mayumi, Pedro Santoro Perez, and José Augusto Baranauskas. 2012. "How Many Trees in a Random Forest?" Machine Learning and Data Mining in Pattern Recognition, 154–68. https://doi.org/10.1007/978-3-642-31537-4_13.

[17] Patel, Jigar, Sahil Shah, Priyank Thakkar, and K Kotecha. 2015. "Predicting Stock Market Index Using Fusion of Machine Learning Techniques." Expert Systems with Applications 42 (4): 2162–72. https://doi.org/10.1016/j.eswa.2014.10.031.

[18] Quinlan, J.R. 1987. "Simplifying Decision Trees." International Journal of Man-Machine Studies 27 (3): 221–34. https://doi.org/10.1016/s0020-7373(87)80053-6.

[19] Schaefer, Anton Maximilian, Steffen Udluft, and Hans-Georg Zimmermann. 2008. "Learning Long-Term Dependencies with Recurrent Neural Networks." Neurocomputing 71 (13-15): 2481–88. https://doi.org/10.1016/j.neucom.2007.12.036.

[20] Shanker, M., M.Y. Hu, and M.S. Hung. 1996. "Effect of Data Standardization on Neural Network Training." Omega 24 (4): 385–97. https://doi.org/10.1016/0305-0483(96)00010-2.

[21] Zhang, Jingyi, Masao Utiyama, Eiichro Sumita, Hai Zhao, Graham Neubig, and Satoshi Nakamura. 2016. "Learning Local Word Reorderings for Hierarchical Phrase-Based Statistical Machine Translation." Machine Translation 30 (1-2): 1–18. https://doi.org/10.1007/s10590-016-9178-7.

[22] Zhang, Yi. 2017. “Strategies for Combining Tree-Based Ensemble Models.” CCE Theses and Dissertations, January. https://nsuworks.nova.edu/gscis_etd/1021/.