[Team 6] Project F4: Deep Learning for Bitcoin Prediction

Aditya Kadole (akadole), Shreyas Chikkballapur Muralidhara (schikkb), Shrihari Sundararaj (ssunda23)

***Abstract*—Cryptocurrency has grown exponentially over the past decade, with the most rapid advances seen in the past few years. Bitcoin has been one of the top hit in social media and search engines recently. Cryptocurrency due to their high volatility leads to the great potential of high profit if intelligent inventing strategies are taken. Unfortunately, due to their lack of indexes, Cryptocurrencies are relatively unpredictable compared to traditional financial instruments. The goal for this project is to show how a trained model can predict the price of a cryptocurrency using Bitcoin if we give the right amount of data and computational power using Long Short Term Memory (LSTM). The trends for Bitcoin prices are available for a considerable number of years, we will develop a model based on the Kaggle Historical Bitcoin dataset. We have built the model using LSTM and predicted the price of bitcoin.**

***Index Terms*—Long Short Term Memory (LSTM), Cryptocurrency, Bitcoin**

I. Model Training and Selection

Our approach is to verify if time series data is stationary before the pre-processing step. To check for data stationarity and using that data for forecasting time series, we performed the Augmented Dickey-Fuller Test, Seasonal Decomposition test, and Autocorrelation with differencing.

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test. The intuition behind this test was to determine how strongly a time series is defined by a trend. We have also implemented seasonal decomposition with an additive model in order to know whether the data is stationary or not. Time series forecasting splits the time series data into Trend, Seasonal, Residual and observed. It checks on annual seasonality trend in the provided time series data. Along with these checks,we performed autocorrelation test. Autocorrelation function plots identify the order of differencing required to make the time series stationary along with the similarity of observations as a function of time lag between them.

For training of the model, we have used "bitstampUSD 1-min data 2012-01-01 to 2020-04-22" available in the Kaggle website[1]. Upon performing Exploratory Data Analysis we got to know about fields, values in the dataset and we preprocessed the data in order to suit our model for prediction. The dataset has fields including Timestamp, Open, High, Low, Close, Volume (BTC), Volume (Currency), Weighted Price. Timestamp field has time represented in seconds. It's hard to understand the predicted value based on seconds so we convert those values to date.

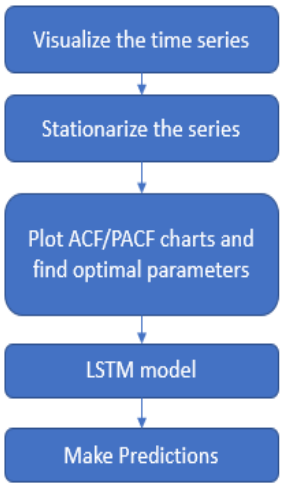

Fig. 1: Proposed Model

In Dataset we have transactions equal to 4.3 million at a frequency recorded per minute. It is hard to process data for each and every minute so we have imputed mean values to generate data on day, week, month frequency. The date level transactions are considered for prediction and further analysis. The fields in the dataset have NaN values in a considerable amount of numbers due to sampling data for each and every minute. These values can't be used for prediction, so rows with NaN are dropped. The pre processed data is separated into train and test data for training and evaluating the model. It is split on the proportion of 80:20 data.Figure below shows distribution of train and test data in the dataset.

The data that is available has values which are of wide range, Hence it is necessary to normalize the data. By using Min-Max scaling the data values in the dataset are mapped on a number from 0 to 1. The train and test data are reshaped to suit the LSTM model.

A. The Model

In this section, we have shown the methodology that has been used to predict the Bitcoin price. We have single out Long Short Term Memory as the best reliable model. Since the Prediction of bitcoin price requires remembering the previous outcome. LSTM is a variant of Recurrent Neural

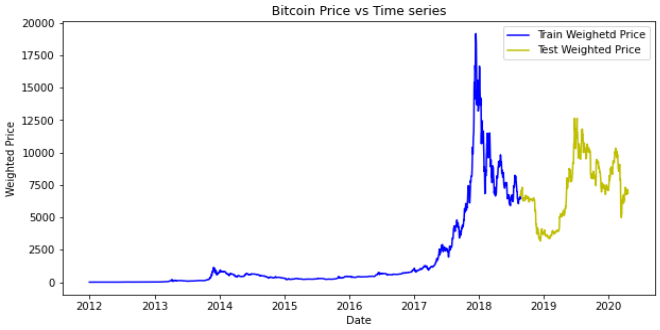

Fig. 2: train and test data split

Network. LSTM layer consists of gates including forget gate, input gate, output gate, and cell state which helps us in remembering the previous outcome as in Figure 2. The cells carry relevant information required for prediction over the time step. The cells states are updated by means of an input gate. As the process goes on the information is added and remembered by cell states. Remembering data which is stored long back is not going to add any significant value for prediction. Therefore LSTM has Forget gate which determines what needs to be remembered and removes the old data. The output gate determines what information should be there in the next hidden state. The architecture of the model includes the LSTM layer at the beginning followed by the Drop out layer, Dense layer, Flatten layer[1].

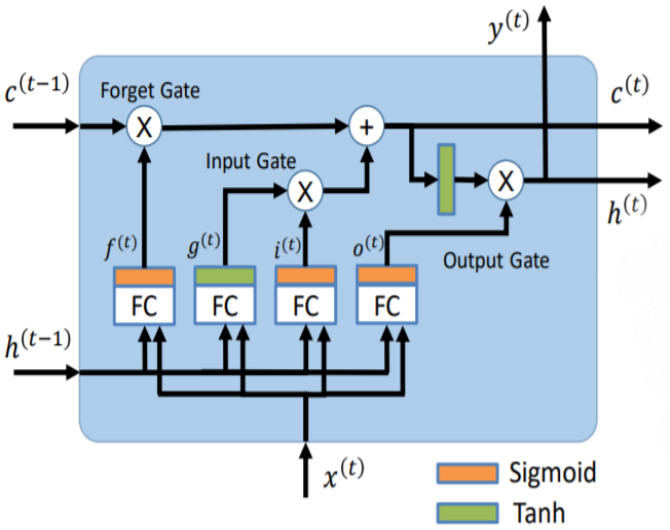

Fig. 3: LSTM unit

The input gates in the LSTM layer performs sequential forecasting for Weighted price attribute. The time step function creates the sequential inputs for the current label of the model. Bitcoin price for the current day is determined by Bitcoin price from previous X days. So if we need to specify the weighted price on X day based on the values from values from X-1 days as train input. Hence the time step is considered as hyper parameter in structuring the sequence forecasting model.

The LSTM model which we have developed has 5 layers. It has 2 LSTM layers with each layer having 256 neurons. Along with this, we have other layers including 1 drop out layer, 1 dense layer, 1 flatten layer as in Figure 3. Here the flatten layer transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.

B. Baseline

Our test data spans for a frame of 18 months between 2018-08-25 to 2020-04-22. Since we are considering the Imputed mean for daily basis, having the Mean absolute error and Root mean mean square error less than $500 USD would be considered as baseline for our model.

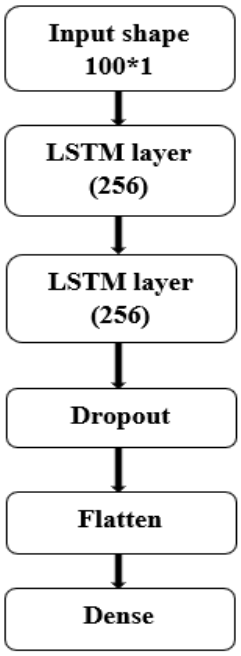

Fig. 4: Model Architecture

II. Experimental Section

A. Metrics

We have considered metrics including Mean absolute error(MAE), Root Mean Square Error(RMSE), Mean Square Error(MSE). These metric parameters are compared with baseline for concluding the accuracy of results obtained. RMSE provides an offset range for the actual value from the predicted value, indicating the extent of error.

Mean Square Error is the average squared difference between the estimated values and the actual value.

Mean Absolute Error measures the average magnitude of the errors in the test set predictions. It doesn't consider direction. It is average over the absolute difference between

actual and predicted values, with each value given equal weightage.

B. Stationary Data check results

We have implemented the Dickey-Fuller test to know about the stationarity of time series data. By implementing so we have witnessed p-value to be 0.4901 which is greater than 0.05. So it is not possible to reject the null hypothesis. Not rejecting null hypothesis implies that the dataset may or may not be stationary.

Then we performed seasonal decomposition with the additive model. The data is found to be stationary as trend and seasonal is the same as residual and observed as shown in Figure 4.

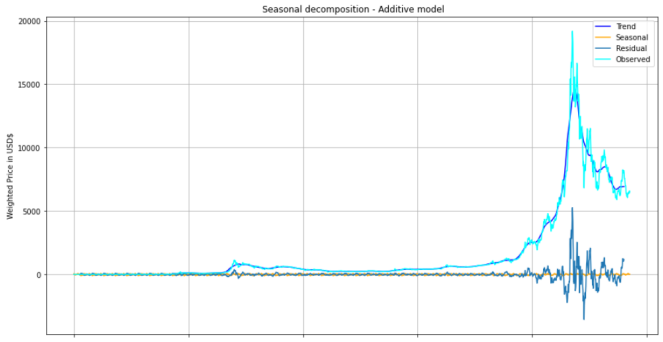

Fig. 5: Seasonal Decompositon

Finally, we performed the Autocorrelation test. Thus for the order of the first differential, the time series is stationary as it matches pacf results. Figures 5,6,7,8 below depicts the results we got upon performing autocorrelation.

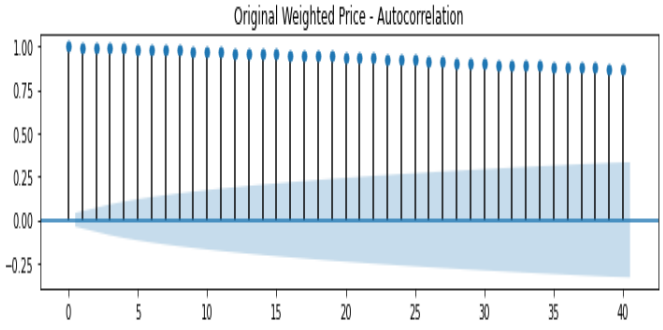

Fig. 6: Original weighted price

C. Model Selection

We have tuned hyper-parameter including Epoch, Learning rate, batch size, Number of neurons per layer. Initially, we tried assigning the number of epochs to 20, batch size as 16, and varied learning rate in order to get low mean absolute

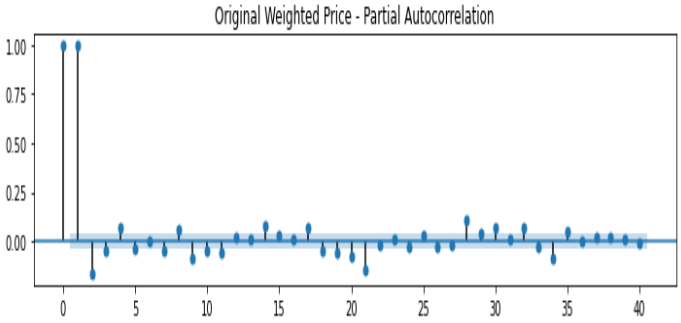

Fig. 7: original weighted price partial autocorrelation

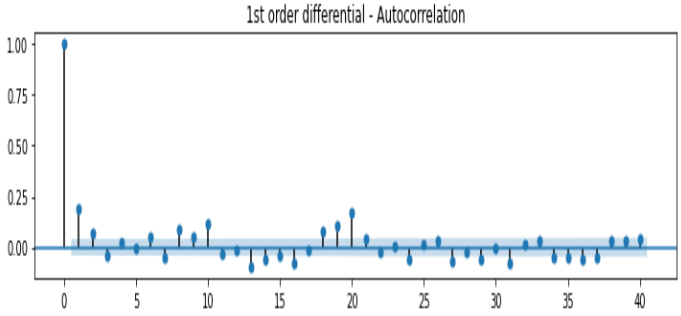

Fig. 8: First order differential correlation

error. We were not able to get encouraging results. So post that we have varied epochs from 20 to 50 keeping batch size and learning rate constant.Based on Figure 9 upon increasing number of epochs the train and validation loss decreases. On comparison train loss is less than validation loss. We have experimented increasing number of epoch and visualised the response of Mean Absolute Error(MAE). From Figure 10 we can infer that the validation loss increase and decrease upon increasing epochs and training loss was keep on reducing. Upon trying with all possible combinations we ended up with hyper-parameters epoch equal to 20 and batch size 16 and learning rate 0.001 which fetched us low validation loss equals 0.01599.

Epoch	Batch Size	Learning rate	val_loss
10	16	0.001	0.06833
20	16	0.001	0.01599
30	16	0.25	0.05320
30	32	0.001	0.01657
10	32	0.001	0.02209
20	32	0.001	0.01859

TABLE I: Model performance for various Hyper Parameter

D. Performance and Comparison with baseline

Performance is measured upon comparing predicting Bitcoin weighted price from August 2018 to April 2020. We have compared the weighted price of actual data and predicted data against the number of days between above

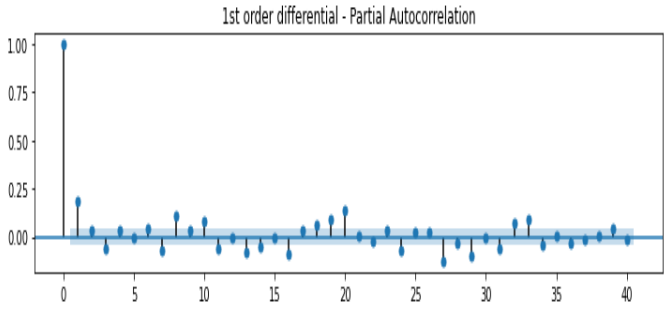

Fig. 9: First order differential partial correlation

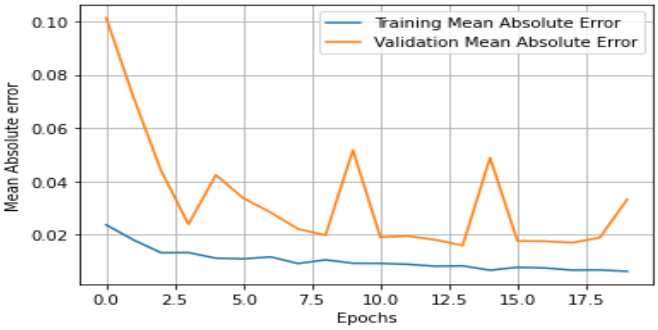

Fig. 11: Mean Absolute Error

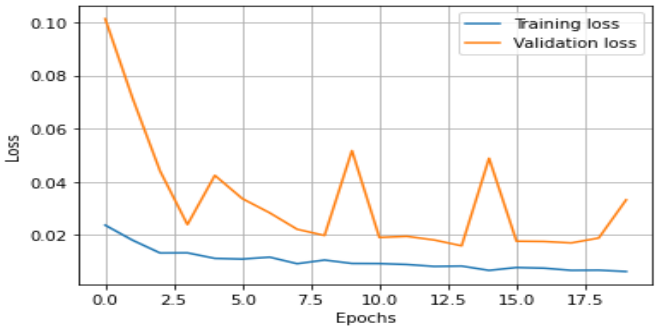

Fig. 10: Train and Validation loss

Fig. 12: Final result

mentioned months Fig 11. Upon implementing this we were able to achieve Root mean square value, Mean Absolute Error, Mean Square Error which matches with baseline. In Table III we have included values we got for these metrics. In Table II we included predicted and actual bitcoin price. From table we can infer that differences are lesser than baseline.

Timestamp	Predicted price	Actual price
04-12-2020	7213.916	6954.061
04-13-2020	7142.497	6751.871
04-14-2020	7031.413	6884.686
04-15-2020	6934.094	6798.847
04-16-2020	6846.189	6914.666
04-17-2020	6803.266	7071.902
04-18-2020	6817.889	7153.927
04-19-2020	6869.909	7166.968
04-20-2020	6934.893	7050.526
04-21-2020	6975.755	6867.155
04-22-2020	6952.915	6850.600

TABLE II: Actual and Predicted value table

MAE	RMSE	MSE
306.7201199344436	443.20439527851875	196430.1359941975

TABLE III: Metrics

REFERENCES

[1] A. Aggarwal, I. Gupta, N. Garg and A. Goel, "Deep Learning Approach to Determine the Impact of Socio Economic Factors on Bitcoin Price Prediction," 2019 Twelfth International Conference on Contemporary Computing (IC3), Noida, India, 2019, pp. 1-5.

[2] M. Wimalagunaratne and G. Poravi, "A Predictive Model for the Global Cryptocurrency Market: A Holistic Approach to Predicting Cryptocurrency Prices," 2018 8th International Conference on Intelligent Systems, Modelling and Simulation (ISMS), Kuala Lumpur, Malaysia, 2018, pp. 78-83.

[3] P. V. Rane and S. N. Dhage, "Systematic Erudition of Bitcoin Price Prediction using Machine Learning Techniques," 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 594-598.