Analysis and Prediction of Cryptocurrency Prices

Péter Garamvölgyi, Federico Zaiter
Department of Computer Science and Technology
Tsinghua University, Beijing, China
garamvoelgyip10@mails.tsinghua.edu.cn
zaitertf10@mails.tsinghua.edu.cn

Abstract

We evaluate the use of deep learning models for cryptocurrency price forecasting. On multi-step prediction, LSTM offers a 15% improvement in prediction accuracy compared to our baselines (linear regression, SVM regression, ARIMA). LSTM does not outperform the baseline for single-step prediction. We also present a detailed description of data preprocessing, where we show how a more representative training dataset improves model performance.

1 Introduction

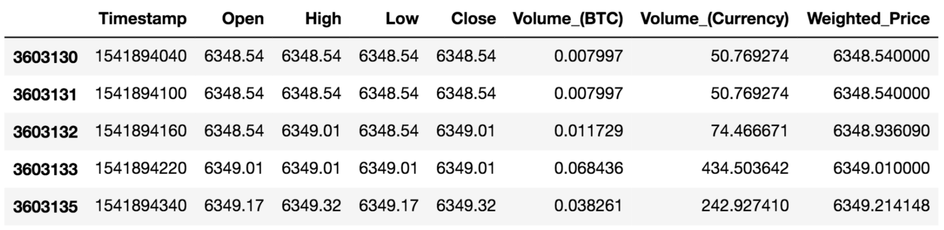
1.1 Problem motivation

Stock market price prediction is a well-established area of study with numerous applications of machine learning models on financial time series. In this work, we evaluate the use of these models for the related problem of cryptocurrency price prediction. The unique characteristics of these time series being non-stationary and volatile introduce challenges. Deep learning models are a potential candidate to tackle these.

1.2 Data description and insights

While our goal is to derive general insights applicable to all cryptocurrencies, we chose to limit our evaluations to Bitcoin. This is justified by the following reasons.

- Bitcoin has over 10 years' worth of price data available, surpassing all other cryptocurrencies.
- Cryptocurrencies tend to have similar price profiles (high volatility, sudden drops, etc.).
- Other cryptocurrencies tend to follow Bitcoin's trend, e.g. drops in Bitcoin price are mirrored in prices of alternative cryptocurrencies.

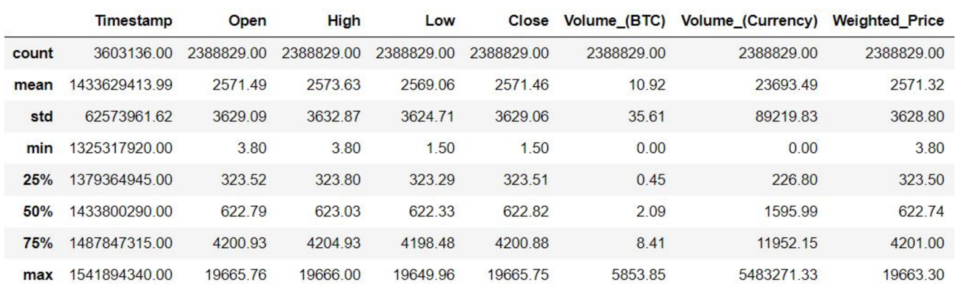


	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
3603130	1541894040	6348.54	6348.54	6348.54	6348.54	0.007997	50.769274	6348.540000
3603131	1541894100	6348.54	6348.54	6348.54	6348.54	0.007997	50.769274	6348.540000
3603132	1541894160	6348.54	6349.01	6348.54	6349.01	0.011729	74.466671	6348.936090
3603133	1541894220	6349.01	6349.01	6349.01	6349.01	0.068436	434.503642	6349.010000
3603135	1541894340	6349.17	6349.32	6349.17	6349.32	0.038261	242.927410	6349.214148

Figure 1: Example entries from Bitstamp's BTC-USD dataset

For working with Bitcoin price data, there are multiple freely available datasets online. We decided to use Bitstamp's BTC-USD dataset for the period between 1st January 2012 and 11th November

2018 (figure 1) [1]. This dataset contains features like the current exchange rate and trading volume on a minute basis for which insights can be seen in figure 2.



	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
count	3603136.00	2388829.00	2388829.00	2388829.00	2388829.00	2388829.00	2388829.00	2388829.00
mean	1433629413.99	2571.49	2573.63	2569.06	2571.46	10.92	23693.49	2571.32
std	62573961.62	3629.09	3632.87	3624.71	3629.06	35.61	89219.83	3628.80
min	1325317920.00	3.80	3.80	1.50	1.50	0.00	0.00	3.80
25%	1379364945.00	323.52	323.80	323.29	323.51	0.45	226.80	323.50
50%	1433800290.00	622.79	623.03	622.33	622.82	2.09	1595.99	622.74
75%	1487847315.00	4200.93	4204.93	4198.48	4200.88	8.41	11952.15	4201.00
max	1541894340.00	19665.76	19666.00	19649.96	19665.75	5853.85	5483271.33	19663.30

Figure 2: Bitstamp's BTC-USD dataset details

We turn the data into a univariate time series. Then, we can draw further insights by decomposing the time series (figure 3). Our assumptions on the characteristics of the data can be confirmed by our visualization of its decomposition. As it can be seen in figure 3, there is a very clear positive trend in the data with a small seasonality and a lot of noise because of its price volatility by the end of the analyzed period, which makes it look almost random. These are among the factors that make the time series non-stationary and therefore harder to predict.

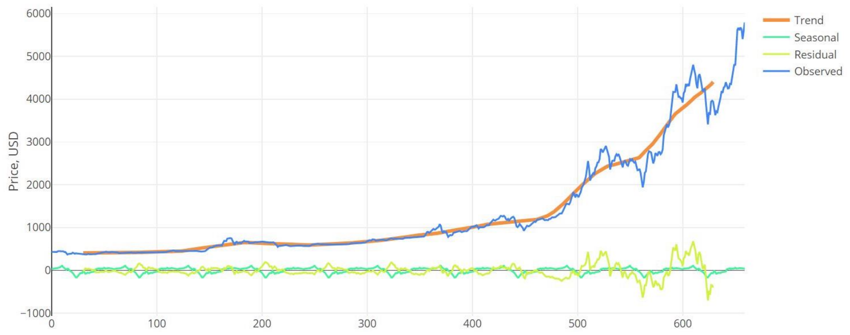


Figure 3: Seasonal decomposition of target data

The rest of the paper is organized as follows. Section 2 gives a high-level overview of our approach and introduction to the models we used. In section 3, we discuss the problem of single-step prediction, comparing LSTMs with a simple lag predictor (to be defined in 3.3). Section 4 turns to multi-step prediction. We compare and evaluate multiple baseline candidates in 4.3, and then proceed to apply LSTM to multi-step forecasting in 4.4. Finally, we present related work in section 5 and summarize our results in section 6.

2 Our approach

Our work focuses on two distinct problems: single- and multi-step time-series forecasting. For both problems, we first define a simple baseline solution. Then, we proceed to implement a more complex, deep learning based approach using recurrent neural networks (RNNs). We compare the performance of these models with that of the baseline and discuss implementation details and challenges.

For the implementation, we use common Python frameworks including numpy, pandas, sklearn, and keras. We rely on plotly for plotting.

We decided to use the Root-Mean-Square-Error (RMSE) metric to evaluate and compare our models. This metric is defined as the square root of the average squared error between the predicted values and the true labels of the training set.

$$\text{RMSE}(\mathbf{pred}, \mathbf{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{pred}[i] - \mathbf{y}[i])^2} = \sqrt{\frac{1}{N}(\mathbf{pred} - \mathbf{y})^2}$$

2.1 Models

We used the following models for our evaluations:

- **Linear regression**: In linear regression, we attempt to model the time-series using a linear model, basically fitting a trend line. It is a simple and fast model with limited generalization ability.
- **SVM regression (SVR)**: SVR is an extension of the SVM approach used for regression instead of classification. SVR maximizes the margin to derive a good linear regression model. Non-linearity can be introduced using kernel functions. [2]
- **ARIMA**: Autoregressive Integrated Moving Average models are statistical models often used for modeling non-stationary time-series.
- **LSTM**: Long Short-Term Memory is a type of Recurrent Neural Network (RNN), composed of cells including input, output and forget gates regulating the flow and retention of information. [3]

A detailed theoretical discussion of these models is out of the scope of this paper.

2.2 Preprocessing

One well-known fact about cryptocurrency prices is their significant short- and long-term volatility. To have a stable basis to build on, given recent spikes and dips at the start of the year 2018, we chose to evaluate our models on the time period between 1st January 2016 and 20th October 2017. From this dataset, we only keep the price data daily aggregated, turning this into a univariate time series modeling and forecasting problem.

For separating the data into training and test sets, we tried two approaches.

1. Choose a date and use it to cut the dataset into two (figure 4). This is a common approach; however, the significant differences during different periods in our dataset might have a negative impact on the generalization ability of this approach.

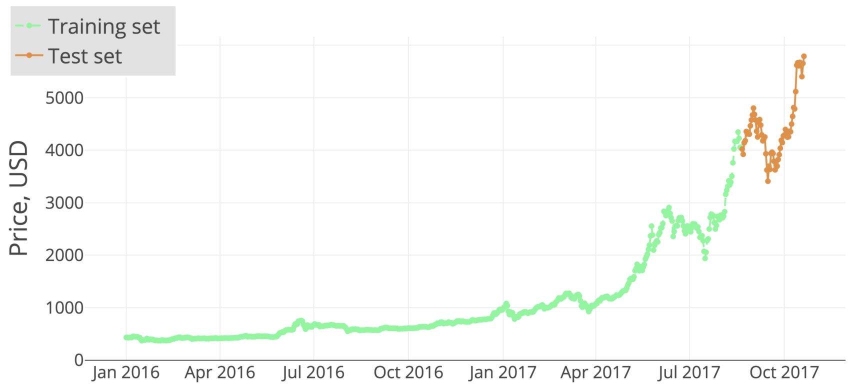


Figure 4: Training and test sets during the chosen time period

2. Cut the original dataset into *chunks* of equal duration and then separate each of these into training and test sets (figure 5). This approach will result in a more representative collection of data points.

The goal of the second approach is to give a more general *view* of the data for the model to train on. At the same time, it makes the testing data more representative of the whole time series, thus

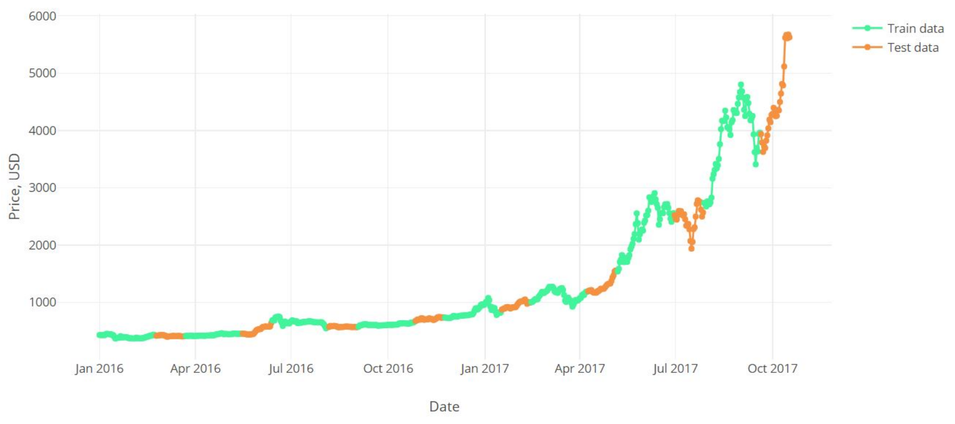


Figure 5: Training and test sets from chunks separations

making predictions more *fair*. In contrast to the first approach where the testing data is different from the training data, approach 2 evens this difference out. This can be clearly seen in the following two boxplot comparisons in figure 6 and figure 7 of training and testing data from approach 1 and 2 respectively.

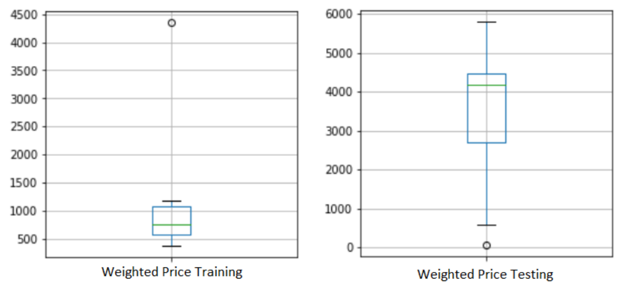


Figure 6: Training versus testing data comparison for preprocessing approach 1

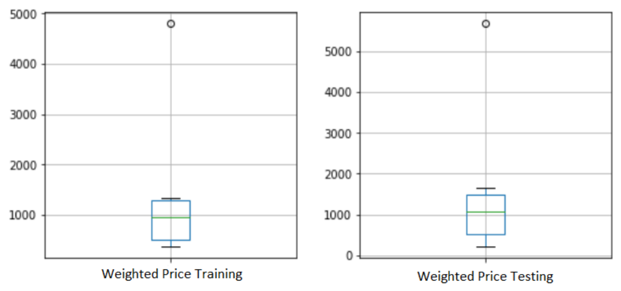


Figure 7: Training versus testing data comparison for preprocessing approach 2

3 Single-step prediction

3.1 Problem definition

First, let us define the problem of single-step prediction. In this scenario, we are given a time window of the last n values of a time-series. Our goal is to predict the value in the next time step immediately after this window.

$$\text{single-pred}(a_1, a_2, ..., a_n) \approx a_{n+1}$$

An example of this approach is predicting tomorrow's price based on the last 7 days.

3.2 Preprocessing

As the problem definition suggests, single-point prediction is a supervised learning task, where our dataset consists of input windows and the corresponding labels.

$$\mathbf{x_i} = [a_{i+1}, a_{i+2}, ..., a_{i+n}] \qquad y_i = a_{i+n+1} \qquad i \in [0, N-n-1]$$

As our predictions are based solely on the price, $\mathbf{x_i}$ is a vector and y_i is a scalar. (Another option would be to use more features, in which case $\mathbf{x_i}$ would become a matrix and y_i would be a vector.)

We use a simple Python function to decompose our original time-series into a series of such $\langle \mathbf{x_i}, y_i \rangle$ pairs. This is basically a sliding window approach, with overlaps between the windows.

3.3 Baseline

Our baseline for single-step prediction is a lag predictor: We simply assume that the price has not changed since the last day.

$$\text{baseline}_{single}(\mathbf{x_i}) = \mathbf{x_i}[n] = a_{i+n}$$

Figure 8 shows the baseline for our sample dataset. At first sight, the predictions seem quite accurate. This is even more so if we evaluate this model on a bigger dataset: the two curves seem to fit perfectly. However, such visual evaluation is misleading, and that is why we rely on the RMSE instead. The RMSE of our baseline model on the test set is 161.024.

$$\text{RMSE}(\text{baseline}_{single}) = 161.024$$

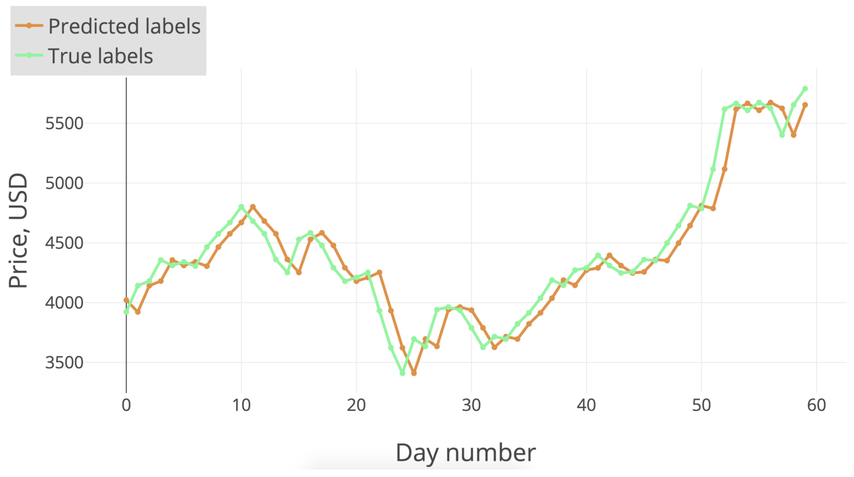


Figure 8: Single-step lag predictor output

3.4 Advanced models

This first model we applied for the single-step problem was ARIMA. We tried different parameter configurations. The best results were delivered by an ARIMA(1,0,0) first-order autoregressive model.

$$\text{RMSE}(\text{ARIMA}_{single}) = 159.945$$

Next, we moved on to applying LSTM. After trying multiple network architectures, we found the best results were delivered by a network with 2 LSTM layers of 128 neurons each, with a densely connected output layer. (This network is very similar to the one used for multi-step prediction, illustrated by figure 12.)

$$\text{RMSE}(\text{LSTM}_{single,lookback=1}) = 159.816$$

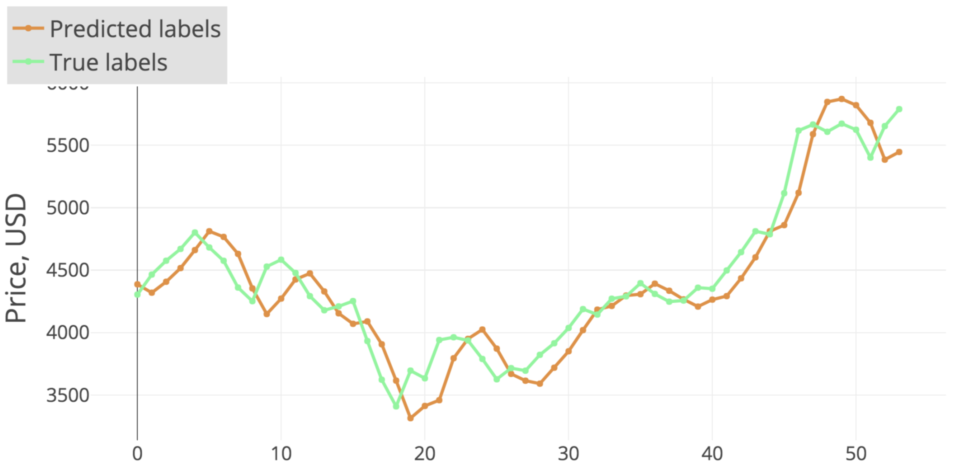


Figure 9: Example predictions using LSTM (look-back=7)

3.5 Results

The RMSE of the different models evaluated (lag predictor, ARIMA, LSTM) shows no significant difference. This suggests that LSTM approximated the baseline but was unable to learn a better prediction based on the limited data available.

Interestingly, single-step prediction does not benefit from increased look-back size, as illustrated by figure 10. The best prediction error was achieved with a look-back of size 1.

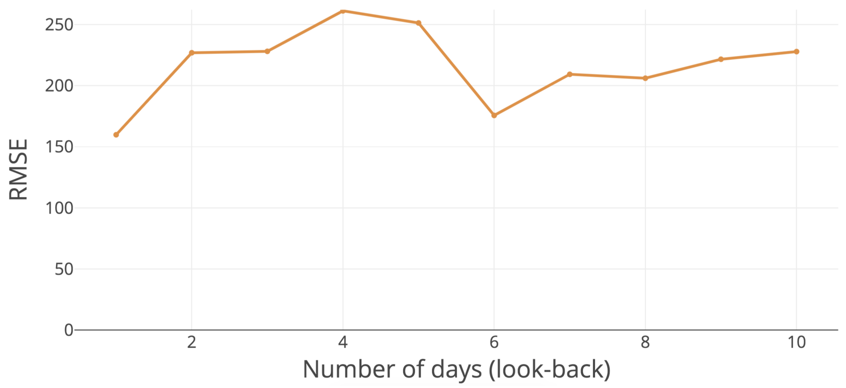


Figure 10: Single-step prediction RMSE for different look-back sizes

We briefly evaluated the impact of adding one additional feature (the daily trading volume) but this change had no impact on the prediction accuracy.

4 Multi-step prediction

4.1 Problem definition

In this section, we will present our results for multi-step prediction. In this scenario, we are given a time window (called *look-back*) of the last n price values. Our goal is to predict the price values for the next m steps immediately after this window (called *look-ahead*).

$$multi\text{-}pred(a_1, a_2, ..., a_n) \approx (a_{n+1}, a_{n+2}, ..., a_{n+m})$$

An example would be predicting the next seven days' prices based on data from the past three weeks.

4.2 Preprocessing

Similarly to the single-point prediction problem, our dataset consists of input windows and the corresponding labels. The difference is that the labels in this case are also vectors and not scalars.

$$\mathbf{x_i} = [a_{i+1}, a_{i+2}, ..., a_{i+n}] \qquad \mathbf{y_i} = [a_{i+n+1}, a_{i+n+2}, ..., a_{i+n+m}] \qquad i \in [0, N - n - m]$$

Again, we rely on Python to decompose our original time-series into a series of $\langle \mathbf{x_i}, \mathbf{y_i} \rangle$ pairs.

4.3 Baseline

For choosing the baseline for multi-step prediction, we evaluated three models:

- simple linear regression,
- SVM regression with radial basis function,
- ARIMA.

All of these models are applied to a single look-back window per prediction. Figure 11 illustrates some example predictions of these three models.

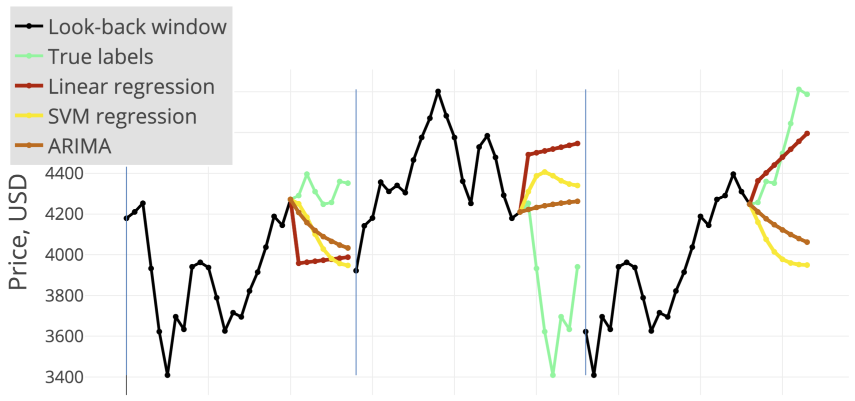


Figure 11: Example predictions using the three baseline candidates

ARIMA delivered the best results during our evaluation, so we chose it as our baseline. As uncertainty grows with every step in multi-step prediction, the corresponding RMSE values are significantly higher than the single-step ones (figure 14).

4.4 Advanced models

Our deep learning model for multi-step prediction is a deep LSTM network. The inputs are the look-back windows, while the outputs are the look-ahead windows.

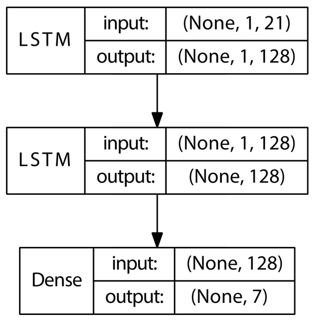


Figure 12: LSTM network parameters

The best performance was given by a network similar to the one used before: Two LSTM layers of 256 and 128 neurons each, followed by a single dense layer.

Figure 13 shows example predictions of our trained model.

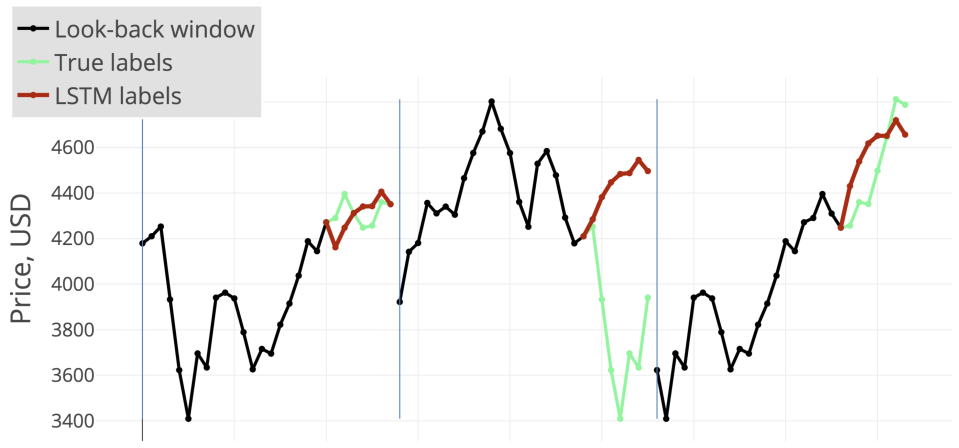


Figure 13: Example predictions using LSTM

4.5 Results

By using LSTM, we achieved a small but significant (about 15%) improvement in overall prediction error on our test data (figure 14). When using preprocessing approach 1, LSTMs already get a 13% improvement and the best performance is gotten using LSTM with preprocessing approach 2. This, as discussed in section 3.2 and visualized in boxplots from figures 6 and 7, is because the training data is more representative of the testing data in the latter than in the former approach. While these results rely heavily on the actual series being used, they suggest that LSTM models can learn some complex features automatically and thus outperform simple models.

5 Related work

ARIMA, SVM, neural networks, and various combinations of these approaches have been applied to stock price prediction several times in recent years. [4][5][6][7]

In the cryptocurrency space, there have been numerous papers attempting to derive knowledge and predictions from sentiment analysis and other social data. [8][9] Other approaches focus on implementing profitable trading algorithms. [10]

In addition, several recent blog posts and tutorials discuss the application of LSTMs to Bitcoin price prediction. [11][12][13]

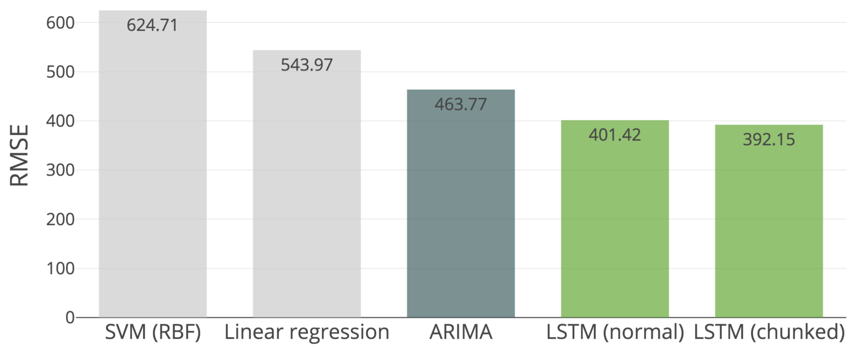


Figure 14: Comparison of prediction errors (RMSE)

6 Conclusions

We evaluated the use of LSTM for both single-step and multi-step time-series forecasting. For single-step prediction, using machine learning models offered no improvement compared to our simple baseline. For multi-step prediction, however, we achieved a ~15% improvement in prediction accuracy with LSTM compared to our best baseline ARIMA.

The results above suggest that the application of deep learning models for price prediction and other time series forecasting tasks is a promising area.

Additionally, we show how important it is to consider the similarity of the distribution between the training and testing data for validation when implementing and evaluating our models as experienced with the two preprocessing approaches used.

Possible future directions for research include a more thorough evaluation of models, hyper-parameter tuning, incorporating more features into the models, and trying the same approaches on other datasets.

References

[1] Kaggle Bitcoin Historical Data https://www.kaggle.com/mczielinski/bitcoin-historical-data

[2] Drucker, H., Burges, C.J., Kaufman, L., Smola, A.J., & Vapnik, V. (1996). Support Vector Regression Machines. NIPS.

[3] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9, 1735-1780.

[4] Ayodele A., Adebiyi et al. (2014). Stock Price Prediction Using the ARIMA Model. 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 106-112.

[5] Pai, P., & Lin, C. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting.

[6] Chen, Kai et al. (2015). An LSTM-based method for stock returns prediction: A case study of China stock market. 2015 IEEE International Conference on Big Data (Big Data), 2823-2824.

[7] Adebiyi, A.A., Adewumi, A.O., & Ayo, C.K. (2014). Comparison of ARIMA and Artificial Neural Networks Models for Stock Price Prediction. J. Applied Mathematics, 2014, 614342:1-614342:7.

[8] Kim, Y.B., Kim, J.G., Kim, W.K., Im, J.H., Kim, T.H., Kang, S.J., & Kim, C.H. (2016). Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. PloS one.

[9] Lamon, C., Nielsen, E., & Redondo, E. (2017). Cryptocurrency Price Prediction Using News and Social Media Sentiment.

[10] Amjad, M.J., & Shah, D. (2016). Trading Bitcoin and Online Time Series Prediction. NIPS Time Series Workshop.

[11] ActiveWizards, (2018). Bitcoin Price Forecasting with Deep Learning Algorithms https://activewizards.com/blog/bitcoin-price-forecasting-with-deep-learning-algorithms

[12] Sheehan, David (2017). Predicting Cryptocurrency Proces with Deep Learning https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-prices-with-deep-learning

[13] Schultze-Kraft, Rafael (2018). Don't be Fooled - Deceptive Cryptocurrency Price Predictions Using Deep Learning https://hackernoon.com/dont-be-fooled-deceptive-cryptocurrency-price-predictions-using-deep-learning-bf27e4837151