Bitcoin Price Forecast

Markus Köfler

February 23, 2023

1 Motivation

Many people have tried to beat the market and become rich by forecasting stock prices and recently, prices of cryptocurrencies. Due to the cryptocurrencies high volatility and sometimes instability, they are seen as extremely speculative assets to invest in. The economic theorem Efficient Market Hypothesis (*EHM*) claims that no gains can be made from predicting financial assets because the prices already reflect current circumstances and are thus representing their fair valuation. [1] Although it might seem intuitive, my motivation to do this project is not to predict the bitcoin price for future investments, but rather to challenge myself to come up with the most effective model in terms of efficiency (simplicity) and accuracy using the knowledge I acquired from the lectures. In the same token, I can implement my entry-level Data Science skills to collect, clean and prepare the data and after having final results, visualizing the outcomes. Therefore, I choose this type of project as I believe it will improve my technical capabilities while having joy along the process. The aim of the project is to prove the efficient market hypothesis (EHM) wrong, through the use of machine learning.

Linear Regression *LR* is a popular and straightforward technique used to model the relationship between a dependent variable and one or more independent variables, whereas long short-term memory *LSTM*, on the other hand, is a specialized type of recurrent neural network *RNN* that can process and predict sequences of data. Although both models fundamentally differ in terms of architecture, heuristics and complexity, I wanted to compare them and analyse, which one is better suited.

1. Karlsson, "Comparison of linear regression and neural networks for stock price prediction"

2 Data

2.1 Multivariate Data

The first dataset, which I initially tried to create myself, has been retrieved from GitHub, created and published by Pratikkumar Prajapati.[2] The dataset consists of a total of 23 columns and a index in hourly timestamp format, whereby the first 9 columns are the open, high and close prices of the cryptocurrencies Bitcoin, Litecoin and Ether. The remaining 14 columns are comprised of numerical sentiments of social media posting on Reddit and news articles on Google News regarding those cryptocurrencies for the given hour. The data was scraped within a period from January 1, 2018 00:00:00 to November 11, 2019 23:00:00.

2.2 Univariate Data

This dataset was downloaded from the website Yahoo! Finance, which is widely known to provide historical stock market data organized in a total of 7 columns, whereby the first column represents the time series index in daily frequency.[3] For the proceedings I only extracted the *Close* column and the time-series index. For my experiments, I manipulated the univariate dataset in different ways described in the following two subsections.

2.2.1 Windowed Data

Windowed time series data is a popular technique used in time series analysis to improve forecasting accuracy by breaking down a time series dataset into smaller, more manageable chunks. To prepare windowed time series data, the data is divided into windows of a fixed size, where each window contains a subset of consecutive data points. The size of the window is typically chosen based on domain knowledge and the characteristics of the data. These windows can then be used as inputs to a time series forecasting model, where the model is trained to predict the next data point based on the values within a given window. By iteratively sliding the window through the time series and forecasting the next data point, a complete forecast can be generated for the entire time series.

I applied the windowing concept only on the univariate data with the Close price of Bitcoin by shifting rows down for the input window such that it consists of previous observations. Regarding the value to predict (I refer

2. Prajapati, *Predictive analysis of Bitcoin price considering social sentiments*
3. , *Bitcoin USD (BTC-USD), CCC - CoinMarketCap. Currency in USD*

to this as true output), I experimented with single value true outputs (*autoregression*) and also a windowed true outputs, whereby I shifted the close price of Bitcoin up such that output window consists of future observations. In the context of this project, the input windows iteratively passed onto a model are the row dimension of the *X_train* variable and the true output values are the row dimension of the *y_train* variable.

Note that depending on the window size, a given number of rows, must be dropped from the either top or the bottom or both from the windowed dataset as they contain missing values. As an example, predicting a single value of a given day based on the values of the past 7 days, we must get rid of the last 7 rows. On the other hand, predicting multiple days ahead based on the values of the past week, we must disregard the first and last 7 rows.

2.2.2 Recursion Technique

Recursive predictions is a time series forecasting technique where the model utilizes previous values of the time series to predict future values repeatedly. I implemented this approach in a way that the predicted values are used in the next iteration as inputs, until the model uses its own predictions to predict the another subsequent value. This process is can be replicated to the point where the desired forecast horizon is reached.

3 Supervised Learning Models

Supervised machine learning models are an appropriate choice when the goal is to recognize patterns and make predictions as it is the case with natural language processing (NLP) or time series forecasting. In supervised learning, the machine learning algorithm is trained using labeled data, meaning that the desired output is already known. The model learns to map inputs to outputs based on the labeled data, and can then be used to make predictions on unseen data. For this project, I dealt with Linear Regression (LR) and Long Short-Term Memory ($LSTM$) which are a memory-bound subclass of Recurrent Neural Networks (RNN).

3.1 Linear Regression

The idea behind LR is to model the linear relationship between a dependent variable and independent variables x_i with $i \in \{0, n\}$ and n observations y fitting a straight line to the data points that best describes the relationship between these variables. The model estimates the values of the coefficients

β_i for for the line based on the data, and can then be used to make predictions for new data points. In general, β_0 is treated as a constant, indicating the intercept with the y-axis whereas the other remaining coefficients are responsible for the slope of the regression line. Each independent variables coefficient can be interpreted as the individual *ceteris paribus* effect on the dependent variable, holding the other variables constant. The goal of linear regression is to minimize the distance between the predicted values $\hat{y}$ and the true values of the dependent variable y, also known as the residuals $\hat{u}$, in order to obtain an accurate model. This is also known as ordinary lest squares (OLS) minimization or minimizing the sum of squared residuals (SSR).[4]

Assuming we have k independent variables and given that

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 * x_{i1} + \hat{\beta}_2 * x_{i2} + ... + \hat{\beta}_k * x_{ik}$$

we aim to minimize

$$\sum_{i=1}^{n}(y_i - \hat{y})^2 \quad = \quad \sum_{i=1}^{n} {\hat{u}_i}^2$$

and therefore obtain a linear combination of parameters of the best fit.

3.1.1 Evaluation: R-squared

A common evaluation criterion is the R-squared (R^2), or coefficient of determination. The closer the R^2 is to one, the more variation is explained by the regression, meaning that the regression line fits the data well. The R^2 can be derived by dividing the sum of squares explained (SSE) by the sum of squares total (SST).[5]
With

$$SSE = \sum_{i=1}^{n} \hat{y}_i - \bar{y}^2$$

$$SST = \sum_{i=1}^{n} y_i - \bar{y}^2$$

whereby

$$SST = SSR + SSE$$

the R^2 can be obtained through the the formulas

$$R^2 = \frac{SSE}{SST} = 1 - \frac{SSR}{SST}$$

4. Wooldridge, *Introductory econometrics: A modern approach*
5. Hill, Griffiths, and Lim, *Principles of econometrics*

Alternatively, the coefficient of determination can also be computed by squaring the coefficient of correlation between y and $\hat{y}$

$$R^2 = (\rho_{y,\hat{y}})^2 = \frac{\widehat{Cov}^2(y,\hat{y})}{\widehat{Var}(y) * \widehat{Var}(\hat{y})}$$

which also implies that if $\rho = 0$ then the $R^2 = 0$.

3.2 Long Short-Term Memory

LSTM are a special type of RNN. Unlike their parent class, predictions by LSTM are influenced by both long-term and short-term information, whereas RNN are only able to account for accumulated long-term knowledge. LSTM solve a fundamental issue of RNN, especially for time series data: vanishing or inflating gradients. [6]

In basic *vanilla* RNNs, the gradients can either become tremendously small or large as they are backpropagated, making it difficult to learn long-term dependencies. The root cause of this issue is that if the weight matrix has small or big eigenvalues, the gradients will vanish or explode respectively. LSTM uses a gated cell structure to control the flow of information and gradients, allowing the model to keep information in memory, read, write and forget information, and thus, to remain stable over long sequences.[7]

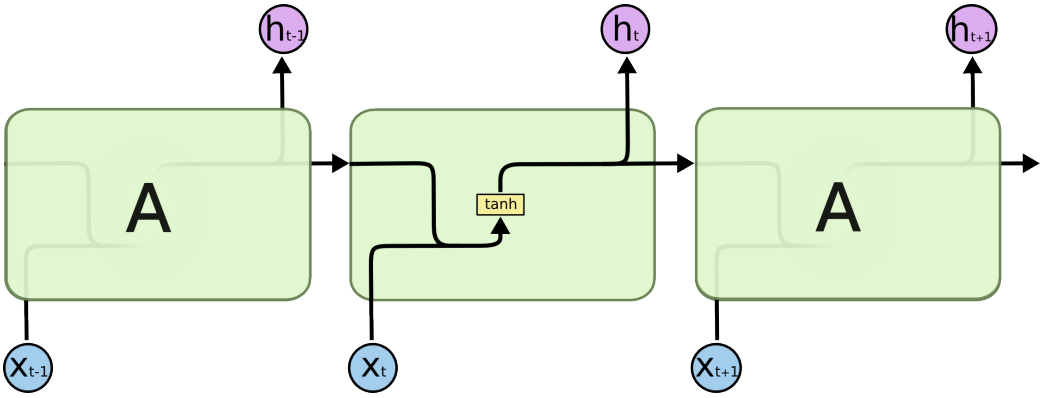


Figure 1: sequenced RNN-Cells

6. Wikipedia contributors, *Long short-term memory - Wikipedia, The Free Encyclopedia*
7. Starmer, *Long Short-Term Memory (LSTM), Clearly Explained*

The LSTM cell contains three gates, which are responsible for controlling the information flow: the input gate i_t, the forget gate f_t and the output gate o_t. The input gate determines how much new information should be added to the cell state, while the forget gate decides which information should be discarded from the cell state. The output gate determines how much of the cell state should be outputted to the next time step.

With the sigmoid function σ compressing inputs into the interval $[0, 1]$, the hyperbolic tangent function $tanh$ mapping inputs on the interval $[-1, 1]$ the element-wise multiplication $\odot$, the new candidate cell state $\tilde{C}_t$, the updated cell state C_t, with h_t as the output at time t , x_t as the input at time t, h_{t-1} as the hidden state at time $t-1$, and W and b as the weights and biases, respectively, we can derive the following equations:

$$
\begin{aligned}
input\ gate: &\quad i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
forget\ gate: &\quad f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
output\ gate: &\quad o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
new\ candidate\ cell\ state: &\quad \tilde{C}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
new\ cell\ state: &\quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
new\ hidden\ state: &\quad h_t = o_t \odot \tanh(C_t)
\end{aligned}
$$

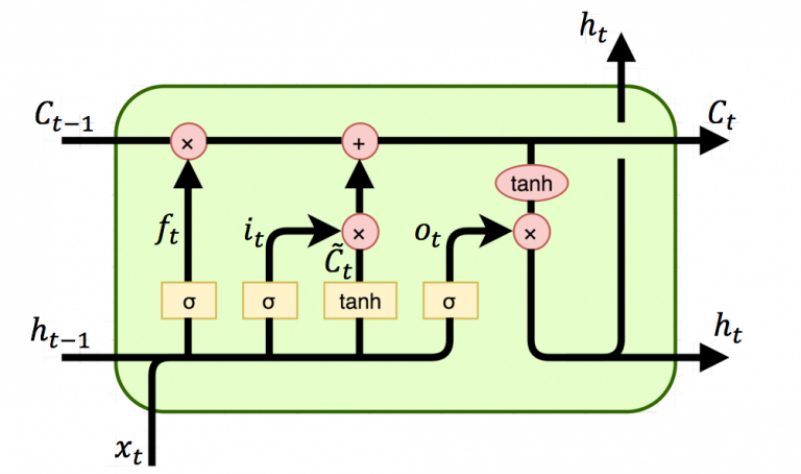


Figure 2: LSTM Cell Gates

The process at the input i_t, forget f_t and output o_t gates follows the same pattern, only differing in the weights and biases: the input x_t and the hidden

state of the previous cell h_{t-1} are multiplied with their respective weights $W_{x[i,f,o]}$ and $W_{h[i,f,o]}$, summed up with bias $b_{[i,f,o]}$, evaluated by the sigmoid function σ returning a value between 0 and 1 (the closer the value to 1, the higher the impact). i_t determines how much the new input can impact the cell state f_t determines how much of the old cell state should be retained and o_t determines the impact of the current cell state on new hidden state h_t.

The process for the new candidate cell state $\tilde{C}t$ is the exact same as for input, forget and output, however, instead of sigmoid σ the hyperbolic tangent function *tanh* is used, resulting in a value between -1 and 1. The value represents the new information added to the cell state C_t.

With the new values resulting from the gates, the new cell state C_t and hidden state h_t can be updated for the next cell in the sequence: To obtain C_t, f_t is element-wisely multiplied with the old cell state C_{t-1} gate and added to the product of the element-wise product of i_t and $\tilde{C}t$. Finally, the hyperbolic tangent function is applied on the updated cell state C_t and multiplied by the output gate's value o_t resulting in the new hidden state h_t.

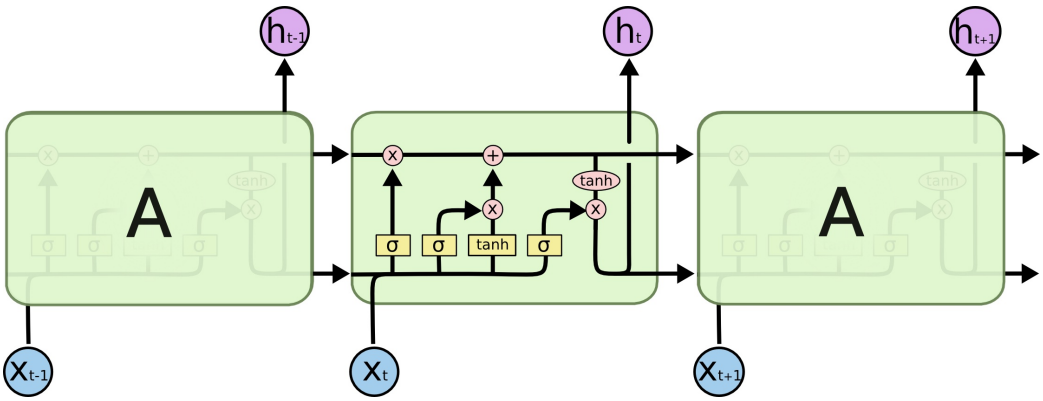


Figure 3: sequenced LSTM-Cells

Note that there are many other derivatives of this basic LSTM cell construct in practice such as *peephole connections*. In summary, the LSTM solves the vanishing and exploding gradient problem of RNNs by using a gated cell structure that controls the flow of information and gradients, allowing them to remain stable over long sequences.[8]

8. Olah, Christopher, *Understanding LSTM Networks*

3.2.1 Evaluation: MAE, MSE and RMSE

Common metrics to evaluate the performance of NN are the mean squared error (MSE), mean absolute error (MAE) or root mean squared error ($RMSE$).

MSE is a popular choice for regression problems because it penalizes large errors more heavily than small errors, which can be useful in situations where the goal is to minimize the overall error. However, it can be sensitive to outliers and may not provide a clear sense of how well the model is performing across the full range of target values.

$$MSE = \frac{SSR}{n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

MAE, on the other hand, provides a measure of the average error and is less sensitive to outliers, making it a good choice in situations where the data contains extreme values or when the goal is to minimize the average error.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

RMSE is the square root of MSE and provides a measure of the standard deviation of the error. This metric particularly useful in situations where the y has a large range of values. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable. It is a good choice when the goal is to minimize the overall error while also accounting for the variability in the data.[9]

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

Note that the formulas stated above do not account for the degrees of freedom, which depend on the number of independent variables k.

9. Chai and Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature"

4 Literature

4.1 LR vs. LSTM

LR and LSTM models are both commonly used for predicting financial asset prices, but they differ in their approach and performance. LR models rely on a linear relationship between the input variables and the output, and thus, are suitable for simple, linear datasets. However, financial asset prices often exhibit complex, nonlinear behavior, ultimately making the simple but effective concept of LR useless. In contrast, LSTM models are designed to handle sequences of data, making them well-suited for predicting financial time series. Instead of establishing a linear line of best fit, neural networks such as are trained to approximate a function $f(x)$ that maps an input x to a specific output or category y.[10] LSTM, in addition, exhibit advantages from its specific construction to capture long-term dependencies and nonlinear patterns in the data, which can lead to more accurate predictions. Still, regression analysis is a popular approach for prognosis long-term trends. For the task of predicting cryptocurrency or stock prices, LSTM models generally perform the best, also amongst other neural network architectures or classification models, considering their ability to capture complex, nonlinear relationships.[11]

4.2 Hyperparameters and Architecture

Due to its construction, LR does not allow for hyperparameter tuning. Solely modifications on the data or reformulation of the task can impact the result of LR such as taking *logarithms* or removing a certain percentage of outliers. In contrast, LSTM have many hyperparameters that can potentially be optimized:

1. Layers: deeper networks (several LSTM layers) can potentially learn more complex patterns, but may also be more prone to overfitting

2. Units: the more units inside a layer, the better the model's ability to capture different patterns in the data. Hence, more units may lead to better performance but may also increase the training time and computational complexity

10. Jaquart, Dann, and Weinhardt, "Short-term bitcoin market prediction via machine learning"

11. , "Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach"

3. Batch size: larger batches, number of samples that are processed in each training iteration, usually lead to faster convergence of training and validation loss (faster learning), but may also increase the risk of overfitting

4. Learning rate: the learning rate determines how much the weights of the model are updated during training, implying that higher learning rates can result in the model overshooting the optimal weights, while lower learning rates can lead to slow convergence or even getting stuck in local minima

5. Sequence length: the length of the input sequence can also influence the performance as longer sequences capture more information, but should be customized based on the task

6. Epochs: the number times that the algorithm will work through the input data. Too many epochs lead to overfitting, too less epochs to chronic generalization

7. Activation functions: several types of activation functions can be chosen, commonly *ReLU*, or *linear* for time series forecasts [12]

Experiments demonstrate the power of deep LSTM models, using 3 to 5 hidden layers with LSTM unit numbers in a range between 16 to 512. It is even recommended to use the *Adam* optimizer in combination with a larger batch size to minimize the binary cross-entropy loss during the training process.[13] Another factor worth mentioning is that the predominant feature for all regression-type models is Bitcoin price itself. [14] I chose a simplistic, one-layered architecture of the model (no deep learning). I then improved the performance by manipulating mainly the epochs, and batch size.

12. Buslim et al., "Comparing Bitcoin's Prediction Model Using GRU, RNN, and LSTM by Hyperparameter Optimization Grid Search and Random Search"

13. Jaquart, Dann, and Weinhardt, "Short-term bitcoin market prediction via machine learning"

14. , "Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach"

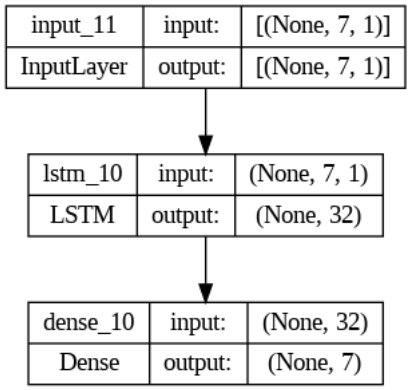


Figure 4: Example Architecture of LSTM model (7-7)

5 Implementation in Practice

All the experiments have been implemented using *Python 3.8.10* in conjunction with *Google Colaboratory Notebooks*. The LR models have been created with *Sci-Kit Learn* and *Pycaret*, whereas the LSTM has been built with the *Tensorflow Keras* infrastructure. Note that the split of data into training, testing and if applied validation follows a consecutive pattern, meaning that the time series stays intact.

5.1 Prediction on Multivariate Data with LR

For this experiment, I used the package *Pycaret* which has a reputation for simplifying machine learning. I performed a 80% training and 20% test split on the multivariate data described in section 2.1 and regressed the variable *close_BTCUSDT* on all 22 remaining cryptocurrency and media sentiment variables. A feature-importance plot gave impactful insights in what variables have been influential for the results. Consequently, I performed a t-test to test for significance of each individual variable, and a F-test for the significance of the regression between an unrestricted model (all 22 variables as regressors) and a restricted model (only 3 variables as regressors).

5.2 Prediction on Univariate Data with LR

In this experiment, I created the used the dataset from section 2.2 and created a windowed dataframe as explained in section 2.2.1. This essentially allows for *autoregression*. Autoregression describes the modelling of a relationship between the current value of the time series and a certain number of lagged

values (past days) of the same series. [15] Again, a 80/20 train-test-split has been used. The models were established using *Sci-Kit Learn*. Eventually, I applied the recursion technique from section 2.2.2 which uses a predicted value to predict the next value of a time series.

5.3 Prediction on Univariate Data with LSTM

Here, I applied the same scheme as for the previous subsection 5.2, but with different input window sizes of 3 days and 7 days and one true output value. Unlike as with the LR, I allocated the data with 80% to training, 10% to validation and 10% to testing. The LSTM models were coded with *Tensorflow Keras*. messing around with different recursion horizons. Also, I created a simple LSTM model with an input window of 7 days and an output window of 7 days, meaning that this model can predict the next week based on the last week's prices without recursion.

6 Results

6.1 Linear Regression

According to the feature plot, only the variables *high_BTCUSDT*, *close_LTCUSD* and *close_ETHUSD* have been used. To demonstrate, why all other 19 variables have been omitted without negatively affecting the R^2, I ran t-tests with the null hypothesis $H_0 : \beta_k = 0$ and a F-test for the significance of the unrestricted and the restricted regression model with $H_0 : \beta_1 = \beta_2 = ... = \beta_k = 0$ using the statistical programming language R and obtained the following results:

15. Wikipedia contributors, *Autoregressive model - Wikipedia, The Free Encyclopedia*

	Dependent variable:	
	close_BTCUSDT	
Variable	Unrestricted	Restricted
high_BTCUSDT	0.938*** (0.007)	0.994*** (0.0003)
low_BTCUSDT	0.743*** (0.006)	
open_BTCUSDT	−0.681*** (0.008)	
volume_BTCUSDT	−0.004* (0.002)	
close_LTCUSD	−0.044** (0.017)	−0.167*** (0.025)
volume_LTCUSD	0.002*** (0.001)	
close_ETHUSD	−0.001 (0.003)	−0.022*** (0.004)
volume_ETHUSD	0.0003* (0.0002)	
gnews_flair	1.789* (1.075)	
gnews_tb_polarity	−12.448 (11.405)	
gnews_tb_subjectivity	17.301* (9.150)	
gnews_sid_pos	−942.001 (2,176.416)	
gnews_sid_neg	−975.956 (2,176.448)	
gnews_sid_neu	−918.252 (2,176.008)	
gnews_sid_com	−1.420 (1.782)	
reddit_flair	1.187* (0.628)	
reddit_tb_polarity	0.454 (2.951)	
reddit_tb_subjectivity	0.766 (2.108)	
reddit_sid_pos	−4.216 (5.972)	
reddit_sid_neg	−10.245 (7.043)	
reddit_sid_neu	−2.380 (1.911)	
reddit_sid_com	−1.194 (2.105)	
Constant	922.205 (2,176.149)	26.834*** (1.428)
Observations	16,536	16,536
R^2	1.000	0.999
Adjusted R^2	1.000	0.999
F Statistic	3,093,307.000*** (df = 22; 16513)	10,358,237.000*** (df = 3; 16532)
Significance Code:		*p<0.1; **p<0.05; ***p<0.01

Table 1: Summary output of Linear Regression Models

Conducting the F-test using the R^2 approach with k independent variables and q restrictions imposed we get

$$F = \frac{R_U^2 - R_R^2}{1 - R_U^2} * \frac{n - k - 1}{q} = \frac{1 - 0.999}{1 - 1} * \frac{16536 - 22 - 1}{19} = 0$$

which falls into the non-rejection region. Thus, we prefer the simpler model, as it does not differ significantly from the full model.

This proves that especially the media sentiments are redundant in a LR model due to insignificance. Even though the critical value of the F-test

is considerably large in both models, the critical value is higher for the restricted model. The reason for the higher F statistic of the restricted model is the degrees of freedom, that is, accounting for the number of independent variables.

No matter how I reformulated the learning task, whether I only used today for tomorrow's forecast or the past 3 days, the R^2 of the prediction never fell below 0.98. The experiment with recursion from 2.2.2 served to demonstrate the limits of LR. As the LR's coefficients are fixed, they recursively predict the same slope. This, unfortunately is not useful to predict volatile asset classes.

6.2 LSTM

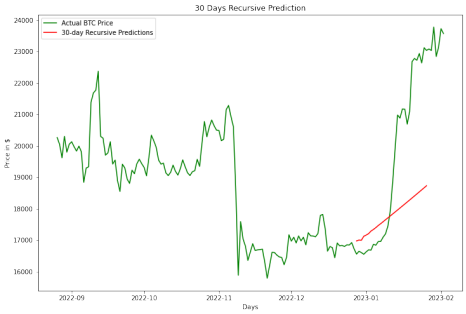
In the scope of the project, I used forecast horizons of 30, 60 and 100 days, however, only the 30 days forecast has been used for comparison.

In Table 2, I compare the results from the experiment trials. The model (3-1) was trained on the previous 3 days to predict the next day's price, the model (7-1) was trained on the previous week to predict the next day's price and the model (7-7) was trained on the past week to predict the next week's prices. The letter R indicates recursive predictions as explained in 2.2.2, whereby predictions are made based on previous predictions.

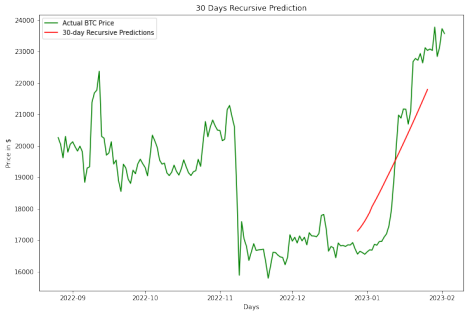
Input Days - Predicted Days	MSE	MAE	RMSE
3-1	1250611.5	852.99	1118.31
7-1	1365692.5	800.69	1168.63
7-7	93220.91	249.74	305.32
3-1 R	6077744.96	1885.69	2465.31
7-1 R	1685472.38	1218.76	1298.26
7-7 R	1438262.42	954.61	1199.28

Table 2: Results of LSTM predictions

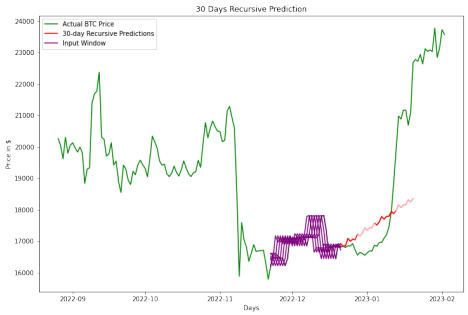
The results from the above table suggests that a 7-day input window leads to better results, especially when predicting the future 7 days.
The graphical representation of the performance measures from Table 2 are shown in the figure above. Based on those, the model using the past week as input to predict the next week (7-7) has the smallest errors. Also the model trained on the past 7 days to predict the next day (7-1) is superior to the model only trained on the past 3 days (3-1). Note the purple lines in subplot (c) are representing the previous week, that has been used as input array to generate the forecast for the next week.



(a) 3-1



(b) 7-1



(c) 7-7

Figure 5: 30 Days Recursive Predictions

7 Discussion

From the results obtained, I conclude that LR models are well suited for outlining an overall trend in the data based on the training observations. LR was initially designed to showcase, how variables are correlated to each other. This can be used to investigate long run relationships like the overall stock market prediction, which is clearly not as volatile as cryptocurrencies, for the next 10 years, given the last century. Unsurprisingly, LR predicts the next day very accurately. But this is somewhat misleading, as someone could simply multiply the last day's price by 0.99 and most of the time would end up at a value quite close to the actual one. Thus, there is very limited utility to predicting the next day's price as it will most likely not differ all that much from the current day. Additionally, predicting the next day based on the last days does not differ significantly from predictions, say, based on the past 3 days. To prove this empirically I ran a small simulation, running the regression of today's Bitcoin price to predict on multiple days in the past. The coefficient of yesterday ($Target - 1$) however, remains most influential.

For the context of this project, I conclude that a simple LR model is not the appropriate statistical tool to use. In contrast, LSTM is - due to

days in the past	$\beta_{Target-1}$
1	0.99870309
2	0.97373293
3	0.97383954
7	0.97231716

Table 3: Coefficient of *Target-1*

its architecture - capable of considering short and long term patterns. Even for the highly fluctuating Bitcoin price, the LSTM correctly picks up the upward trend of the recent weeks. Regarding the recursive predictions, we must pay attention to not view this as a promising result. The further the predictions move into the future, the higher the likelihood of severe failure. Eventhough LSTM can consider fluctuations, once it picks up a persistent upward or downward trend, it will recursively stick with this trend.

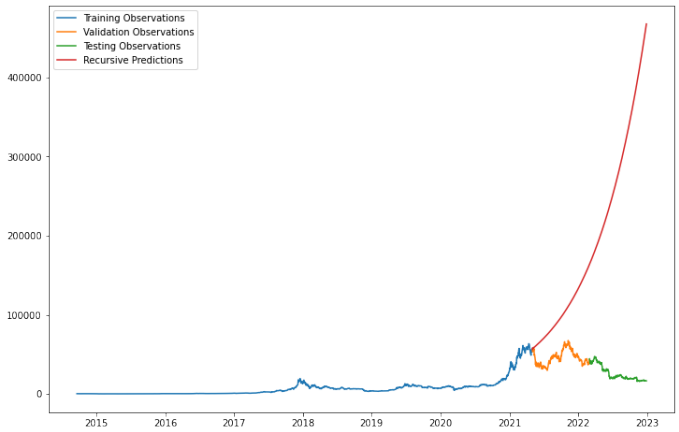


Figure 6: LSTM picking up exponential rise

To summarize, the choice of model heavily depends on the task we are trying to accomplish. LR may be a decent choice when modelling relationships among variables for decision-making, or measuring the consequences of a policy on society. With respect to time series forecasting of volatile assets, LR is not helpful. LSTM can model much more complex relationships, but can hardly be interpreted. For future projects, I intend to advance the construction of the model including reasonable hyperparameter optimization.

8 Resources

8.1 Online Resources

- Youtube
- GitHub
- Stack Overflow
- Papers with Code
- Kaggle
- Jupyter Notebooks
- Google Colaboratory
- RStudio
- Yahoo! Finance
- Towards Data Science

8.2 Libraries & Code Documentation

- python 3.8.10
- pandas
- numpy
- matplotlib
- seaborn
- pycaret
- R 4.2.1
- tensorflow/keras
- deepcopy
- pickle
- tidyverse
- stargazer

Bibliography

Bitcoin USD (BTC-USD), CCC - CoinMarketCap. Currency in USD. https://finance.yahoo.com/quote/BTC-USD. Accessed: 2023-02-07.

Buslim, Nurhayati, Imam Lutfi Rahmatullah, Bayu Aji Setyawan, and Aryajaya Alamsyah. "Comparing Bitcoin's Prediction Model Using GRU, RNN, and LSTM by Hyperparameter Optimization Grid Search and Random Search." In *2021 9th International Conference on Cyber and IT Service Management (CITSM)*, 1–6. 2021. https://doi.org/10.1109/CITSM52892.2021.9588947.

Chai, T., and R. R. Draxler. "Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature." *Geoscientific Model Development* 7, no. 3 (2014): 1247–1250. https://doi.org/10.5194/gmd-7-1247-2014. https://gmd.copernicus.org/articles/7/1247/2014/.

Hill, R Carter, William E Griffiths, and Guay C Lim. *Principles of econometrics*. John Wiley & Sons, 2018.

Jaquart, Patrick, David Dann, and Christof Weinhardt. "Short-term bitcoin market prediction via machine learning." *The Journal of Finance and Data Science* 7 (2021): 45–66. ISSN: 2405-9188. https://doi.org/https://doi.org/10.1016/j.jfds.2021.03.001. https://www.sciencedirect.com/science/article/pii/S2405918821000027.

Karlsson, Nils. "Comparison of linear regression and neural networks for stock price prediction," Uppsala Universitet, 2021. https://www.diva-portal.org/smash/get/diva2:1564492/FULLTEXT02.

"Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach." *Neural Comput & Applic*, 2020. https://doi.org/https://doi.org/10.1007/s00521-020-05129-6.

Olah, Christopher. *Understanding LSTM Networks*. [Online; accessed 20-February-2023], 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Prajapati, Pratikkumar. *Predictive analysis of Bitcoin price considering social sentiments*, 2020. arXiv: 2001.10343 [cs.IR].

Starmer, Josh. *Long Short-Term Memory (LSTM), Clearly Explained*, 2022. https://www.youtube.com/watch?v=YCzL96nL7j0.

Wikipedia contributors. *Autoregressive model - Wikipedia, The Free Encyclopedia*. [Online; accessed 17-February-2023], 2023. https://en.wikipedia.org/w/index.php?title=Autoregressive_model&oldid=1139020229.

———. *Long short-term memory - Wikipedia, The Free Encyclopedia*. [Online; accessed 17-February-2023], 2023. https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1139608545.

Wooldridge, Jeffrey M. *Introductory econometrics: A modern approach*. Cengage learning, 2015.

Link to Notebooks:

https://drive.google.com/drive/folders/1uUH71oD1fiAwIW-QB82fpvulDs2EpJYH?usp=sharing

Link to Data Sources:

https://github.com/pratikpv/predicting_bitcoin_market

https://finance.yahoo.com/quote/BTC-USD/history?p=BTC-USD