Predicting Bitcoin price using technical indicator data features in Long short-term Memory models

Ruud van der Hagen
STUDENT NUMBER: 2031321

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:
Dr. Gonzalo Napolès
Dr. Bruno Nicenboim

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
May 2021

Predicting Bitcoin price using technical indicator data features in Long short-term Memory models

Ruud van der Hagen

Crypto currency has seen tremendous growth over the last few years. Growing to a near multi trillion dollar asset market, it is starting to become an asset class of its own. However, while predicting bitcoin price accurately can be very profitable, getting accurate predictions is difficult due to the volatile nature of the price action. This research attempts to leverage a Long short-term memory model to predict bitcoin prices. By using hourly price data, volume data and technical indicators as features, the best performing model outperformed the baseline model significantly. However, price predictions made by the model still are not overly accurate. This is largely due to imbalance in the data set caused by a recent surge in growth.

1. Introduction

Bitcoin and crypto currency has gained a lot of attention as an asset class over the last years. These digitised assets have created new financial channels through which peer to peer transactions can be constructed without the need of a centralized third party (Hileman and Rauchs 2017). Furthermore, they have introduced a new way of storing value. While originally designed as a digital currency meant for usage in transactions, bitcoin is often seen more as a speculative investment instrument nowadays (Yermack 2015). This change in use case is largely due to high price volatility. Inherently, bitcoin as an asset does not have value. It is not backed by any institution, nor is it pegged to any other inherently valuable asset. Its' price reflects the investors' confidence in the asset (Chen, Li, and Sun 2020). This attribute in turn leads to volatility. Being a measure of risk, volatility is a key variable when designing investment or trading strategies (Chun, Cho, and Ryu 2020). Being able to predict highly volatile price fluctuations can be very valuable.

Using machine learning and neural networks for asset price prediction has seen a rise in popularity over the last decade. In traditional stock markets, previous studies have shown that using these models outperform traditional linear models (Hiransha et al. 2018; Pang et al. 2020). However, there are some key differences between traditional markets and bitcoin. Firstly, bitcoin can be traded 365 days a year, 24 hours a day. Secondly, research has shown that Bitcoin can act as a hedge against equities and currencies or commodities (Bouri et al. 2017). Lastly, bitcoins' strong multifractality leads to the market being more inefficient than traditional equities (Al-Yahyaee, Mensi, and Yoon 2018). These differences imply that machine learning applications used for predicting traditional markets cannot simply be copied and used for bitcoin.

Due to the nature of bitcoin price data, recurrent neural networks such as long short-term memory perform better than traditional machine learning models such as multi level perceptrons (Pawar, Jalem, and Tiwari 2019).

An important aspect of using any machine learning application is feature selection. In traditional stock market prediction tasks, features such as volatility indices, interest rate spreads and foreign exchange rates are used (Chun, Cho, and Ryu 2020). In other research, technical indicators were used. Technical indicators are price data calculated by mathematical formulas during technical analysis, some of which have been proven to be solid indicators of price movement (Dai et al. 2020). In this paper, several different LSTM models are trained using a wide variety of technical indicators. Results are then compared to find which technical indicators are most robust when attempting bitcoin price prediction.

The main question this thesis answers is "What data features are most impactful when predicting Bitcoin price using time series data?". To assist in answering this question, several sub questions are formulated. Firstly, the Related Work section discusses several types of data features commonly used in financial prediction tasks. Using this information, experiments can be designed to determine which features are most useful for the problem of predicting bitcoin price movements. Secondly, literature is studied to find the most suitable prediction model for this task. As discussed earlier, recurrent neural networks are expected to perform best on this type of problem (Pawar, Jalem, and Tiwari 2019). This is dissected further in the Related Work section. Lastly, to find the most suitable time frame for this task, experiments are conducted using different time frame settings. This is discussed in the Experimental Setup section.

2. Related Work

As stated before, price prediction for publicly traded assets is nothing new. Several researches from the nineties show usage of neural networks and back propagation networks to predict stock prices (Kimoto et al. 1990; Freisleben 1992; Mizuno et al. 1998; Schumann and Lohrbach 1993). Since then, machine- and deep learning models have gotten a lot more complex and capable. Recent studies use high-dimensional data and features to predict future price movements (Pang et al. 2020). Furthermore, they use more than just historical price data. Features such as market conditions and sentiment, news articles, social media data and technical indicators are used to predict prices more accurately (Chun, Cho, and Ryu 2020; Vargas et al. 2018; Dai et al. 2020).

Technical indicators are derived from technical analysis. Technical analysis is a method of predicting future price movements using historical price data such as opening price, closing price and trading volume (Nazário et al. 2017). Previous research has shown that technical analysis can in fact be used to add value to investment strategies, and can outperform a simple buy-and-hold strategy (Lo, Mamaysky, and Wang 2000; Dai et al. 2020; Shynkevich et al. 2017). This method of analysing historical price movements can be used to form data features that in turn can be used as inputs for prediction models (Shynkevich et al. 2017). In essence, technical indicators are mathematically derived tools that for example indicate whether an asset is in a strong up or down trend, or whether it is overbought or oversold (Shynkevich et al. 2017).

Technical indicators can generally be divided in 5 main categories (Barone and Potters 2021). Firstly, trend indicators show in which way the asset is trending. Secondly, mean reversion indicators attempt to show when a price trend will reverse. Thirdly, trend strength indicators measure how strong a price trend is. This is done through calculating oscillations in price activity (Barone and Potters 2021). Momentum indicators measure how fast prices are changing over a certain time period. Lastly, volume indicators indicate whether there are more sellers or buyers. Some examples of often used indicators include relative strength index (RSI), simple and exponential moving averages (SMA, EMA) and stochastic %K and %D (Nazário et al. 2017).

Bitcoins' rise in popularity has logically brought along more interest in solving the problem of price prediction. Previous research has attempted to predict bitcoin price using historical price data, blockchain features or both (Ji, Kim, and Im 2019; Chen, Li, and Sun 2020). Huang, Huang, and Ni (2019) used several high-dimensional technical indicators, however they exclusively used a classification tree-based model for predictions. This choice of model is in conflict with findings from previous studies, all of which state a significant difference in performance between tree-based models and RNN models (Chun, Cho, and Ryu 2020; Pawar, Jalem, and Tiwari 2019; Nabipour et al. 2020). Other researches aimed at predicting traditional asset prices also show significant difference in performance between LSTMs and other architectures (Selvin et al. 2017; Li, Shen, and Zhu 2018; Roondiwala, Patel, and Varma 2017). Therefore, this paper will compare LSTMs with different hyper parameter settings and data features in order to predict bitcoin price movements.

3. Experimental Setup

For this research, bitcoin price data was gathered from the exchange Binance. Binance is the largest crypto currency exchange by volume and they offer a free API for registered users. Using the API, historical price data can easily be retrieved for many different crypto currencies (Bin 2021). The data is explored in section 3.1. Then, extra features are engineered from this data. This process and the resulting features are further discussed in section 3.2.

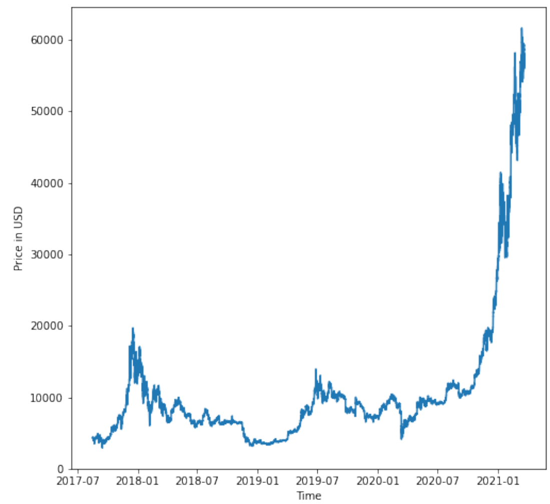
3.1 Data

As stated before, all price data was gathered from the crypto currency exchange Binance. In particular, historical bitcoin price data was gathered starting from the 17th of October 2017 up to the 22nd of March. Using the API, it is possible to select different time frames for the data. For experimental purposes, data was gathered per 1 minute, 5 minutes, 1 hour and 1 day. The number of instances in the data set thus depends on the chosen time unit. For reference, retrieving hourly data from Binance between the above dates results in 31411 instances. The features for each instance are shown in Table (1). After experiments with different time frames, the hourly data was chosen for the definitive training. Lower time frames produced much bigger data sets which made it impossible to train enough models for comparison due to time constraints. Daily data in turn created a smaller data set, which performed worse than hourly data.

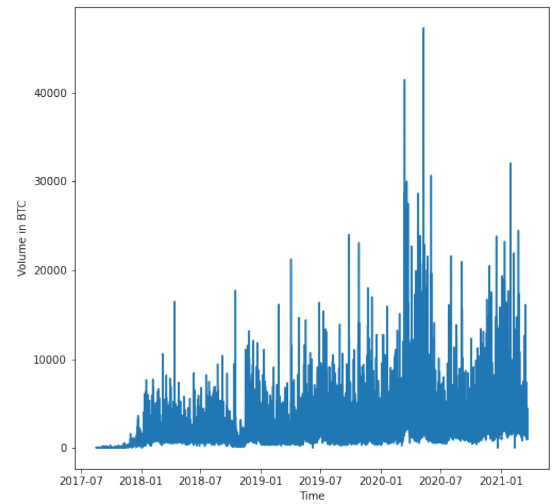
Table 1: Data features

Data feature	Description
Open time	Encoded timestamp of the start of the time frame
Open price	Price in United States Dollar at the start of the time frame
Highest price	Highest observed price in USD in the time frame
Lowest price	Lowest observed price in USD in the time frame
Closing price	Price in USD at the end of the time frame
Close time	Encoded timestamp of the end of the time frame
Quote asset volume	Volume in quote asset (USD)
Number of trades	Total number of executed trades
Taker buy base asset volume	Volume of taker in base asset (BTC)
Taker buy quote asset volume	Volume of taker in quote asset (USD)

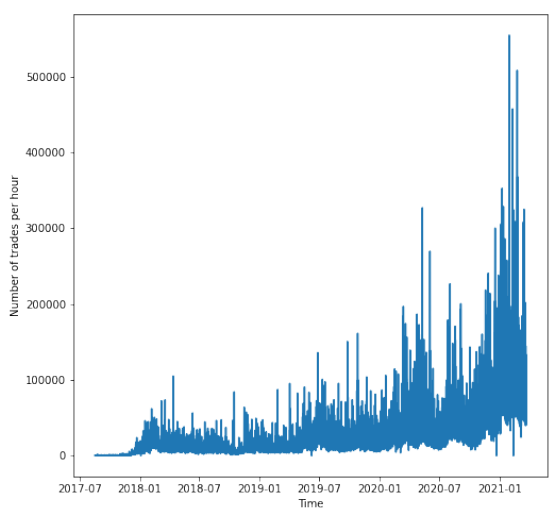
3.1.1 Data exploration. At first glance, it is clear that Bitcoin has seen strong growth over the past 4 years. Figure (1a) shows an explosive growth in asset pricing. Trading volume and raw number of trades also see a lot of growth over the same time period. Figure (1b) shows a very large peak in volume around march 2020. A likely explanation is the large covid-19 related sell-off, which also occurred in traditional markets (Daube 2020). Much like the bitcoin price graph however, growth is the clear trend with volume. The same goes for the number of trades as seen in Figure (1c). This growth does lead to a seemingly unbalanced data set, however. Most of the rapid movements in price occur in the last year.



(a) Bitcoin price in USD



(b) Volume in Bitcoin



(c) number of Trades per hour

Figure 1: Data exploration graphs

3.2 Feature engineering

A number of features are engineered from the historical price data above. These features are all technical indicators as discussed in the Related Work section. The technical indicators that are used are found in Table (2). The features were extracted from the data set with the TALib python library. Section (3.3) discusses how the indicators work and how they are engineered from the data.

Table 2: Technical indicators

Abbreviation	Description
SMA	Simple Moving Average
EMA	Exponential Moving Average
RSI	Relative Strength Index
Stoch_K	Stochastic Oscillator
Stoch_D	Moving average of Stochastic Oscillator
AD	Chaikin Accumulation Distribution line
CCI	Commodity Channel Index
MACD	Moving Average Convergence/Divergence

3.3 Technical indicator features

In this section, the aforementioned engineered features are further explained. All technical indicators are mathematically derived from the price data (Nazário et al. 2017). For some indicators, more than just price values are required. For example, trading volume and differences between opening and closing price are used.

A commonly used simple trading strategy involves use of moving averages. Using this strategy, an investor is supposed to buy an asset whenever its price is above its average price over a given time frame (Zhu and Zhou 2009). In this research, two types of moving averages are used. Firstly, the Simple Moving Average (SMA) is an arithmetic moving average, calculated according to Formula (1) (Ilomäki, Laurila, and McAleer 2018).

$$SMA = \frac{C_1 + C_2 + ... + C_n}{n} \tag{1}$$

where C is the close price and $_n$ is the time period. The second type of moving average used in the training of the models is the Exponential Moving Average (EMA). The EMA is similar to the SMA, however smoothing is applied giving more recent data points more weight. Because of this, the EMA is more sensitive to recent price movements (Nakano, Takahashi, and Takahashi 2017). The $EMA(n)$ is calculated as shown in Equation (2).

$$EMA(n)_t = \frac{2}{n+1} \times (C_t - EMA_{t-1}) + EMA_{t-1} \tag{2}$$

where C_t is the closing price on day t, n is the number of time periods. For the first EMA calculation , SMA_n is taken instead as there is no initial value for EMA_{t-1}.

One of the most widely used technical indicators is the Relative Strength Index (RSI). The RSI is an indicator which shows the relative strength of an asset relating to the market it is traded on (Taran-Morosan 2011). In order to determine the RSI value, firstly the increase (I) or decrease (D) of the closing price for each day have to be calculated. This is done with Formulas (2) and (3) respectively.

$$I_{close} = close_{today} - close_{yesterday} \tag{3}$$

$$D_{close} = close_{yesterday} - close_{today} \tag{4}$$

When the price goes up for a certain day, I will be positive while D will be negative. In this case, D is set to 0. At the same time, when the price decreases, I will be set to 0 (Taran-Morosan 2011). Secondly, the EMA of I and D is required to calculate the relative strength (RS). These EMAs are calculated using Formula (2). The RS is then calculated using Formula (5).

$$RS = \frac{EMA_{Increase}}{EMA_{Decrease}} \tag{5}$$

Lastly, this value is converted to an index which has values ranging from 0 to 100. This is done with Formula (6).

$$RSI = 100 - 100 \times \frac{1}{1 + RS} \tag{6}$$

When interpreting RSI values, there are two main levels to look for. Generally, when the RSI value of an asset goes above 70, it indicates that said asset is overbought. This in turn means that it is expected that the assets' price will come down. Secondly, if the RSI value goes below 30, it means that the asset is oversold. Logically, it is then expected that the asset price will go up (Gumparthi 2017).

The stochastic oscillator is another type of momentum indicator, which in essence gives an indication of an assets' closing price relative to a range of recent price range (Ijegwa et al. 2014). Similar to the RSI, the stochastic oscillator returns an index value between 0 and 100. Unlike the RSI however, the overbought and oversold values are 80 and 20 respectively. The stochastic oscillator is measured with two values, $\%K$ and $\%D$ where $\%K$ is calculated as shown in Formula (7).

$$\%K = 100 \times \frac{C - L_n}{H_n - L_n} \tag{7}$$

where C is the closing price, L_n is the lowest observed price in the past n time periods and H_n is the highest observed price in the last n time periods. $\%D$, being a 3 day moving average of $\%K$, is calculated with Formula (8) (Thorp 2000).

$$\%D = 100 \times \frac{H_3}{L_3} \tag{8}$$

where H_3 is the highest observed price value in the last 3 time periods and L_3 is the lowest observed value in the last 3 time periods.

Chaikin's accumulation/distribution, part of the Chaikin Money Flow indicator, is based on the principle that if an assets price closes above its midpoint, that there was accumulation of the asset that day. Vise versa, if the asset closes below the mid point, the asset was distributed (Kannan et al. 2010). The AD is calculated as shown in Formula (9).

$$AD = M(t-1) + M(t) \tag{9}$$

where M is the money flow multiplier, defined as

$$M = N * V(t)$$

and N is the money flow volume, defined as

$$N = \frac{(C-L)-(H-C)}{H-L}$$

where t is the time period, V is the trading volume, C is the close price, L is the lowest price and H is the highest price.

The commodity Channel Index, or CCI, is a momentum indicator that aims to identify trend reversals (Maitah, Prochazka, and Cermak 2016). Similarly to the RSI and Stochastic $\%K$ and $\%D$, the CCI is an oscillator which identifies overbuying and overselling. Unlike the other two oscillators, the CCI is an unbound index which means that historic price action has to be taken into account when identifying overbought and oversold areas. Formula (10) illustrates how the CCI is calculated.

$$CCI = \frac{TP - SMA}{.015 \times MD} \tag{10}$$

where TP is the Typical Price, defined as

$$TP = \sum_{i=1}^{n}((H+L+C) \div 3),$$

where H, L and C are the highest, lowest and close price per time period respectively, n is the number of time periods, and where MD is the mean deviation, defined as

$$MD = (\sum_{i=1}^{n}|TP - SMA|) \div n.$$

The constant value $.015$ is chosen for stability according to Maitah, Prochazka, and Cermak (2016).

Moving average convergence divergence, or MACD, is another popular technical indicator often used to identify buy or sell signals. It is calculated using 2 EMAs with different time periods (Hung 2016). The MACD value represents the distance between the two used EMAs. In general, when the MACD line crosses above 0, this generates a buy signal and vise versa. The MACD is calculated as:

$$MACD_t = EMA(s)_t - EMA(l)_t \tag{11}$$

where s is the short term EMA and l is the long term EMA. Default values for s and l are 12 and 26 respectively (Hung 2016).

3.4 Method / Models

As found in the literature review, a long short-term memory model is expected to perform best on bitcoin price data, as it is sequential in nature (Chun, Cho, and Ryu 2020; Pawar, Jalem, and Tiwari 2019; Nabipour et al. 2020). Other than an LSTM, a recurrent neural network was considered. Similarly to LSTM, RNNs perform well on time series data. However, RNNs run into the problem of gradient vanishing (Wang et al. 2019; Sherstinsky 2020). This problem arises due to the short term memory nature of the RNN. Information processed early on in the data set will not be remembered (Chen, Li, and Sun 2020).

To combat this problem, LSTMs have memory cells that store valuable information for the longer term, but forget less relevant information. The internal memory cell, or its' long term memory, is the most important (Wang et al. 2019). What information gets stored in the long term memory cell is decided by 3 different gates (Sherstinsky 2020).

Firstly, the forget gate decides what information in the model is kept from the last block h_{t-1}. The mathematical formula for the forget gate is illustrated in Equation (12).

$$ft = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{12}$$

where f_t represents the forget gate, σ is the sigmoid function, w_f represents the weight for forget gate f, h_{t-1} is the output from the LSTM block at $t-1$, x_t is the current input and b_f represents the bias for forget gate f. By passing $w_f[h_{t-1}, x_t] + b_f$ through sigmoid σ, f_t outputs values between 0 and 1 where 0 represents forget and 1 represents remember. Jozefowicz, Zaremba, and Sutskever (2015) found that an increased bias of the forget gate leads to a generally increased performance of the model.

The input gate works in a similar way to the forget gate. The weights are different from the ones used in the forget gate, however (Yu et al. 2019; Sherstinsky 2020). The input gate decides what fresh information gets added to the long term memory. Equation (13) shows the mathematical representation of the input gate.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{13}$$

where i_t is the input gate, σ is the sigmoid function, w_i represents the weights for input gate i, h_{t-1} is the output from the previous block at $t-1$, x_t is the current input and b_i is the bias for input gate i.The input gate only sees information from the short term memory h_{t-1} and the current input x_t. Similarly to the forget gate, the output of $w_i[h_{t-1}, x_t]$ is passed through a sigmoid function σ. As before, by having values between 0 and 1 the gate either allows information to pass through or not according to how close the value is to 0 or 1 (Wang et al. 2019). In order to determine what information actually gets put through to the long term memory cell, a vector of new candidate values $\tilde{C}_t$ is generated with Equation (14).

$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{14}$$

where $tanh$ is the activiation function, W_C is the weight for layer $\tilde{C}_t$, h_{t-1} and x_t are the previous output and current output respectively, and b_C is the bias for layer $\tilde{C}_t$.

With the above formulated filters, the old long term memory or cell state $C_t - 1$ is updated to C_t via Formula (15).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{15}$$

where f_t is the forget gate as defined in Equation (12), C_{t-1} is the cell state from the previous step, i_t is the input gate as defined in Equation (13), and $\tilde{C}_t$ is the filter layer as defined in Equation (14). In essence, the previous cell state is filtered by multiplying it by f_t, and the new information which is filtered by the input gate i_t is added.

Lastly, the cell state is filtered one more time before it becomes the output. Firstly, the cell state C_t is put through another sigmoid activation function σ. This sigmoid layer is the output gate, as defined in Formula (16).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{16}$$

where σ is the sigmoid activation function, W_o represents the weights for the output gate o_t, h_{t-1} and x_t are the previous output and current output respectively, and b_o is the bias for output layer o_t. With the the output gate as defined in Equation (16), the final output h_t is generated as illustrated in Formula (17).

$$h_t = o_t * tanh(C_t) \tag{17}$$

3.5 Baseline

A baseline model was used to compare the results of the LSTM models to. This model is represented by

$$p_t = \frac{p_{t-1} + p_{t-2} + p_{t-3}}{3} \tag{18}$$

where p_t is price at time step t. In essence, the baseline model predicts the next price value by taking the mean of the previous 3 prices. The predicted values on the test set are plotted in Figure (2). Using the root mean square error metric to evaluate, the baseline model achieves a value of 19199. The LSTM models will also be evaluated by RSME. This metric will then be compared with the baseline RMSE.

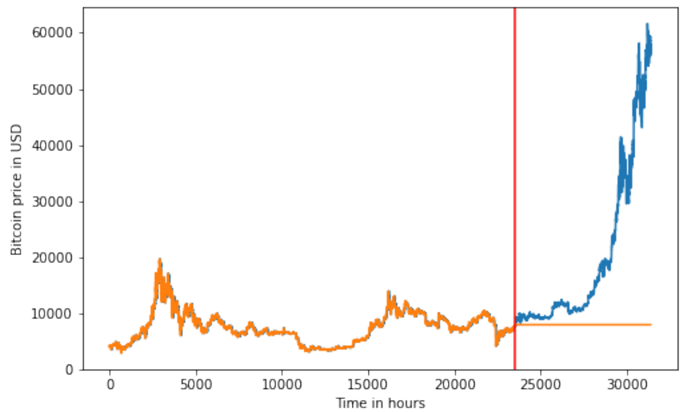


Figure 2: Baseline prediction plot. The red vertical line represents the start of the test data.

3.6 Model architecture

While training the models in order to find the optimal parameter and feature settings, the same general architecture is used. This architecture is illustrated in Table (3).

Table 3: Model architecture

Layer type	Description
LSTM	Long Short-Term Memory layer with n nodes
Dense	Fully connected dense layer with x nodes and l2 kernel regularisation
Dropout	Dropout layer with d dropout rate
Activation	ReLU activation layer

Other tested architectures had multiple LSTM layers. However, this lead to over fitting issues due to the model being too complex for the data set. Furthermore, architectures without normalization were also trained. This also led to the model over fitting and having little to no predictive power and RMSE values on the test set that were higher than the baseline. Before settling on L2 regularisation, batch normalisation was also applied in an experiment. This led to the model not learning anything from the data, as the training error stayed flat during the training process. Training models with multiple dense layers led to similar issues as using multiple LSTM layers did. The dense layer has a set number of dense nodes, which is 1. The reason for this parameter setting is that the model is trying to predict a single value, which is the closing price.

At the start of the process of fine tuning the architecture, a dropout layer was introduced to combat over fitting (Baldi and Sadowski 2013). Section (3.6) discusses the different parameter settings used for this layer. Lastly, an activation layer with the ReLU activation function is implemented. ReLU was chosen due to its reputation as a well performing activation function for LSTM networks (Ang-bo and Wei-wei 2018).

3.7 Hyperparameters and features

As stated before, this research is aimed at predicting bitcoin price using long short-term memory models. In order to find the optimal parameters and features to achieve this, numerous models were trained with different hyper parameter settings and different sets of features. The tested hyper parameter settings are stated in Table (3). All features are compiled in Table (4).

Table 4: Hyperparameter tuning

Hyperparameters	Description	Tested values
Time step	Number of n data points to predict $n+1$	20, 30, 40, 50
LSTM nodes	Number of nodes in LSTM block	7, 14, 21, 28
Batchsize	Number of training examples per iteration	3, 5, 10, 15
Dropout	Dropout for dropout layer	0.1, 0.15, 0.2, 0.3

Table 5: Features

Features
Open price
Highest price
Lowest price
Closing price
Quote asset volume
Number of trades
Taker buy base asset volume
Taker buy quote asset volume
SMA
EMA
RSI
$Stoch_K$
$stoch_D$
AD
CCI
MACD

The hyper parameter tuning process was done on hourly data as stated in the Data Exploration section. Furthermore, the model used the features as formulated in Table (5). The model was trained with 4 settings for each of the 4 hyper parameters. This means that 256 models were trained in total during hyper parameter tuning. Each model is trained for 30 epochs, while no early stopping rule is used. The optimizer used is Adam with a learning rate of 0.0005 , one of the most popular training algorithms (Bock and Weiß 2019). Each model is then evaluated using MSE, where the model with the lowest MSE performs best.

The tested values for time step are arbitrarily chosen. While LSTM models are very suitable for training on sequences of data, they also take relatively long to train (Sherstinsky 2020). Due to time constraints, only the 4 values as formulated in Table (3) were tested. Given more time, other time step values could be tried as well. As for the number of nodes in the LSTM layer, multiples of 7 were chosen as the first model was being trained with 7 features initially. The chosen batchsizes are relatively small, as lower batch size allows networks to train better (Kandel and Castelli 2020). Lastly, the values for the dropout layer are chosen based on common practise (Baldi and Sadowski 2013).

After training the model with the above mentioned hyper parameter values, the optimized settings in Table (6) were found. The model with these settings achieved the lowest test MSE score of all the trained models.

Table 6: Hyperparameter optimization

Hyperparameters	Values
Time step	40
LSTM nodes	28
Batchsize	10
Dropout	0.2

4. Results

With the above mentioned hyper parameter settings, models with different sets of features were trained and compared. The RMSE is used to evaluate all the hyper parameter optimized models. As stated in the Baseline section, each model is then compared to the baseline RMSE. The features for each model are visible in Table (7). Feature abbreviations can be found in Tables (1) and (2).

Table 7: Tested models

Included features	Normalization	RMSE	Number
Close, volume	None	18153.33	1
Close, volume, rsi, sma, ema, $stoch_k$, $stoch_d$	None	24292.86	2
Close, volume, rsi, sma, ema, $stoch_k$, $stoch_d$	L2	15596.44	3
Close, volume, rsi, sma, ema, $stoch_k$, $stoch_d$, ad, cci, macd	L2	18290.07	4
Close, volume, quote av, trades, tb base av, tb quote av, rsi, sma, ema, $stoch_k$, $stoch_d$	**L2**	**12835.51**	**5**
Close, volume, quote av, trades, tb base a, tb quote av, rsi, sma, ema	L2	13390.82	6
Close, volume, quote av, trades, tb base av, tb quote av, rsi, sma, ema, $stoch_k$, $stoch_d$, ad, cci, macd	L2	13042.53	7
Close, volume, quote av, trades, tb base av, tb quote av, rsi, sma, ema, $stoch_k$, $stoch_d$, ad, cci, macd	None	22602.08	8

The simplest model in Table (7) is one that only takes the close price and the volume as features. This model achieved a RMSE of 18153.33 on the test set, which slightly outperforms the baseline RMSE which is set at 19199. Models (2) and (3) are the first to have technical indicators implemented as features. When not using normalization, this leads to a higher RMSE on the test set than model (1). Furthermore, this model does worse than the baseline. Implementing L2 normalization for the same model leads to a lower RMSE, however. Model (3) achieves an RMSE of 15596.44, beating both model (1) and the baseline. Including rsi, sma, ema and the stochastic oscillators like in model (3) improve the models' predictive power.

Model (4) saw the addition of more technical indicator features. Seemingly, adding the ad, cci and macd indicators did not improve the models' performance. While trained on mostly the same features as model (3), it achieved a higher RMSE of 18290.07. The model did have L2 normalization. Adding more of the base price data features alongside close and volume decreases the RMSE of the model, however. Model (5) has the same features as model (3), with quote av, trades, tb base av and tb quote av as extra features. This model performs better than model (4), achieving an RMSE of 12835.51. This makes it the best performing model in this series. Model (5) also had L2 normalization implemented in the model. Model (6) implemented the same features as (5), however without stochastic oscillators. This led to a slightly higher RMSE of 13390.82. Model (6) also had L2 normalization implemented.

The last two models have all of the available features included. Model (7) did have L2 normalization, while model (8) did not. There is a clear difference in RMSE scores for these models. The model with normalization performed significantly better with an RMSE of 13042.53, while the model without any normalization achieved a score of 22602.08. The latter fails to improve over the baseline, while the former shows a significant increase in performance. Similarly to the difference in performance between models (3) and (4) however, model (8) does not perform better than model (5).

As stated in the experimental setup section, models with batch normalization were also trained. However, these results are not included in the research due to extremely bad performance compared to models with L2 normalization. Furthermore, an attempt was made to train a binary classification LSTM model. The goal for this model would be to predict an increase or decrease in price for time step $t+1$ instead of predicting a price value. Experiments with this setup produced bad results, often no better than a coin flip. Partly due to time constraints, and to stay within the scope of this research, no further experiments into binary classification for this problem were conducted. Some other model architectures were also experimented with, however these models produced more results not worthy of mentioning. Lastly, models were trained on a larger data set. This data set included 2 more years of data, starting from 2015 up to 2021. However, the training process for the models became unpractical when taking time constraints into account. Furthermore, these models showed no significant increase in predictive performance.

5. Discussion

In the introduction, several sub research questions were introduced to assist in answering the main research question "What data features are most impactful when predicting Bitcoin price using time series data?". These questions are answered in this section, starting with "What data features are commonly used in financial prediction tasks?". In the Related Work section, it was found that several different data features are used in predicting asset prices. Firstly, historical price data is often used as the base data (Chun, Cho, and Ryu 2020; Vargas et al. 2018; Dai et al. 2020). Often however, more features are introduced. These other features include but are not limited to market sentiment, news articles, social media data and technical indicators. Only technical indicators fell into the scope of this research, however future research could include a wider variety of variables to predict asset prices. While use of technical indicators do improve performance, they are inferred from price data. This price data is already fed to the model, limiting their usefulness. Utilizing other mentioned features such as sentiment analysis or social media data could improve predictability. Due to time constraints however, this was not possible for this research.

As discussed in the Related Works section, technical indicators have proven to be solid features for prediction tasks (Lo, Mamaysky, and Wang 2000; Dai et al. 2020; Shynkevich et al. 2017). A handful of technical indicators used in previous research were selected for this prediction task. Not all indicators improved model performance, however. From experiments it is evident that the Chaiking accumulation/distribution, the commodity channel index and the moving average convergence/divergence do not offer an increase in performance. Other technical indicator features such as the simple moving average, exponential moving average, relative strength index and the stochastic oscillators did show a significant increase in performance. Not every single possible combination of features was tested due to time constraints, however. Furthermore, a lot more technical indicators exist that were not included in this research. In order to fully understand the predictive power of technical indicators, more research has to be conducted. This was not possible for this research due to both time constraints and computational limitations. Outside of the scope of this research, models can be trained utilizing both technical indicator as well as other aforementioned features.

The answer to the second sub question, "Which prediction model is most suitable for financial price prediction tasks?" was answered through literary review. It was found that LSTM models are expected to perform best on financial prediction tasks due to their sequential nature. Other options were multi level perceptron, decision tree and a recurrent neural network. Through literary review it was found that MLPs and decision trees almost always under performed LSTMs. Therefore, during this research, these models were not tested. Given more time, these models could be trained on the same set of features and compared to the LSTM performance. This could give more insight into how big the difference between these architectures truly is. It was also concluded that there was little reason to try recurrent neural networks, due to their inherent problem of vanishing gradient (Wang et al. 2019; Sherstinsky 2020). Another architecture that could be compared to the currently used LSTM are gated recurrent units (GRU). Future research can compare performance of this model with LSTMs for this prediction problem.

Most literature reviewed in the Related Works section was research done in the traditional finance sector. Here, it was found that LSTM models offer a significant advantage over other architectures (Chun, Cho, and Ryu 2020; Pawar, Jalem, and Tiwari 2019; Nabipour et al. 2020). Previous research aimed at predicting bitcoin price used tree-based models instead, which as stated in the Related Works section is in conflict with earlier mentioned results. This is another reason why this research utilized an LSTM model instead of other options.

The last sub question, "Which time frame is most suitable for financial asset price prediction tasks?", was answered with experiments. This was made possible with the flexibility of the Binance API, which allows for different time frames when extracting price data. As stated in section (3.1), creating data sets on lower time frames such as 1 or 5 minutes leads to a very bloated data set with a lot of instances. This in turn causes model training times to heavily increase. Higher time frames such as 4 hours or 1 day lead to data sets that are too small, however. The hourly data proved to be a good equilibrium, creating a manageable data set with sufficient instances for training.

According to the results of the research, numerous added data features add predictive power to the LSTM model. Especially comparing to a model only trained on close price and volume, models trained with more features perform notably better. In correspondence with the literature, technical indicator features increase the predictive power of the model significantly. However, seemingly not all of the indicators lead to better performance. Models which include original data features other than pure price data also perform better than models which do not. Number of trades and the taker buy base and quote asset volumes lead to better performance. The best performing model outperforms the baseline model by a significant amount.

6. Conclusion

From the results of the experiments, model (5) as seen in Table (7) performs best according to RMSE score. There is a significant difference in accuracy between models that have technical indicator features and models that do not. Findings from the experiments suggest that RSI, moving averages and stochastic oscillators increase model performance. This is in agreement with the reviewed literature. The best performing model also significantly outperforms the baseline model. However, the predictions made by the model are not great. This is clearly visible when plotting the predicted price and comparing it to the real data. The prediction plot can be seen in Figure (3).

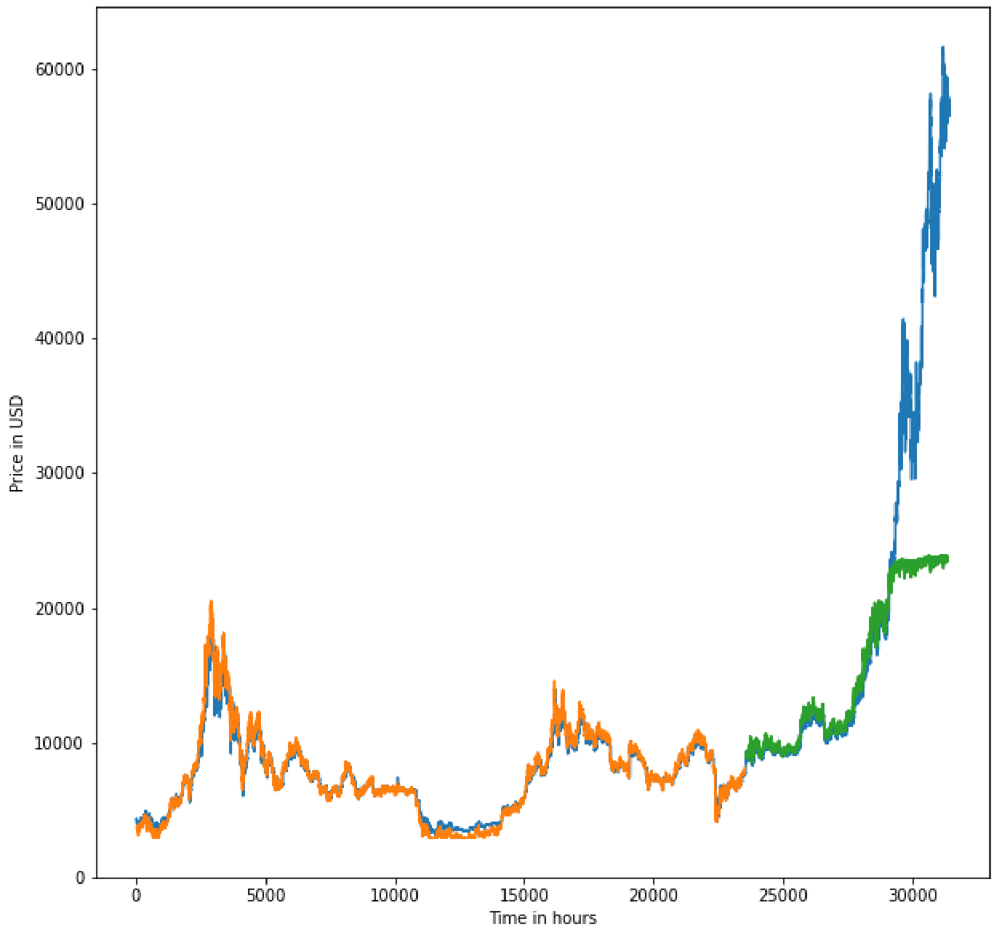


Figure 3: BTC Price prediction in USD. The blue line is the true price, the orange line is the prediction on the training set, the green line is the prediction on the test set.

As is visible in Figure (3), predictions on the test set are not very accurate. This could be accounted to imbalance in the data set. As is also visible in Figure (3), the test set does not represent the training set very well. As mentioned before, models were trained on more data as well. However, this did not increase performance of the model. Model performance could be increased by including more data features, either more technical indicators or other aforementioned features. With the results of the experiments, the conclusion can be made that making use of technical indicators in training can improve predictive performance of LSTM models. As for the other features such as sentiment or news articles, models can also be trained having these in addition to the tested features. The results showed that data that is not inherently related to the price value also increases model performance. Examples of these features include amount of trades and different volume data.

One of the limitations of this research is that only a small number of technical indicators were tested. Given more time and computational resources, more models could be trained with a higher number of combinations of technical indicators. This could give a better indication of the predictive power of technical indicators as a whole. Another limitation in this research is the data. As discussed before, the data seems imbalanced. Future research could attempt to modify the data with re balancing strategies. Due to time constraints, this was not included in this research. Future research could also attempt to create a trading strategy according to model results, in order to make results useful in practical scenarios. This was not in the scope of this research.

References

2021. Binance api documentation.
https://binance-docs.github.io/apidocs/spot/en/#change-log.

Al-Yahyaee, Khamis Hamed, Walid Mensi, and Seong-Min Yoon. 2018. Efficiency, multifractality, and the long-memory property of the bitcoin market: A comparative analysis with stock, currency, and gold markets. *Finance Research Letters*, 27:228–234.

Ang-bo, JIANG and WANG Wei-wei. 2018. Research on optimization of relu activation function [j]. *Transducer and Microsystem Technologies*, 2.

Baldi, Pierre and Peter J Sadowski. 2013. Understanding dropout. *Advances in neural information processing systems*, 26:2814–2822.

Barone, Adam and Charles Potters. 2021. Top technical indicators for rookie traders.

Bock, Sebastian and Martin Weiß. 2019. A proof of local convergence for the adam optimizer. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, IEEE.

Bouri, Elie, Rangan Gupta, Aviral Kumar Tiwari, and David Roubaud. 2017. Does bitcoin hedge global uncertainty? evidence from wavelet-based quantile-in-quantile regressions. *Finance Research Letters*, 23:87–95.

Chen, Zheshi, Chunhong Li, and Wenjun Sun. 2020. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied Mathematics*, 365:112395.

Chun, Dohyun, Hoon Cho, and Doojin Ryu. 2020. Economic indicators and stock market volatility in an emerging economy. *Economic Systems*, 44(2):100788.

Dai, Zhifeng, Xiaodi Dong, Jie Kang, and Lianying Hong. 2020. Forecasting stock market returns: new technical indicators and two-step economic constraint method. *The North American Journal of Economics and Finance*, 53:101216.

Daube, Carl Heinz. 2020. The corona virus stock exchange crash.

Freisleben, Bernd. 1992. Stock market prediction with backpropagation networks. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 451–460, Springer.

Gumparthi, Srinivas. 2017. Relative strength index for developing effective trading strategies in constructing optimal portfolio. *International Journal of Applied Engineering Research*, 12(19):8926–8936.

Hileman, Garrick and Michel Rauchs. 2017. Global cryptocurrency benchmarking study. *Cambridge Centre for Alternative Finance*, 33:33–113.

Hiransha, M, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2018. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132:1351–1362.

Huang, Jing-Zhi, William Huang, and Jun Ni. 2019. Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3):140–155.

Hung, Nguyen Hoang. 2016. Various moving average convergence divergence trading strategies: a comparison. *Investment management and financial innovations*, (13, Iss. 2 (contin. 2)):363–369.

Ijegwa, Acheme David, Olufunke Rebecca Vincent, Olusegun Folorunso, and Olusola Olasunkanmi Isaac. 2014. A predictive stock market technical analysis using fuzzy logic. *Computer and information science*, 7(3):1–17.

Ilomäki, Jukka, Hannu Laurila, and Michael McAleer. 2018. Market timing with moving averages. *Sustainability*, 10(7):2125.

Ji, Suhwan, Jongmin Kim, and Hyeonseung Im. 2019. A comparative study of bitcoin price prediction using deep learning. *Mathematics*, 7(10):898.

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International conference on machine learning*, pages 2342–2350, PMLR.

Kandel, Ibrahem and Mauro Castelli. 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4):312–315.

Kannan, K Senthamarai, P Sailapathi Sekar, M Mohamed Sathik, and P Arumugam. 2010. Financial stock market forecast using data mining techniques. In *Proceedings of the International Multiconference of Engineers and computer scientists*, volume 1.

Kimoto, Takashi, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. 1990. Stock market prediction system with modular neural networks. In *1990 IJCNN international joint conference on neural networks*, pages 1–6, IEEE.

Li, Hao, Yanyan Shen, and Yanmin Zhu. 2018. Stock price prediction using attention-based multi-input lstm. In *Asian Conference on Machine Learning*, pages 454–469, PMLR.
Lo, Andrew W, Harry Mamaysky, and Jiang Wang. 2000. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(4):1705–1765.
Maitah, Mansoor, Petr Prochazka, and Michal Cermak. 2016. Commodity channel index: Evaluation of trading rule of agricultural commodities. *International Journal of Economics and Financial Issues*, 6(1).
Mizuno, Hirotaka, Michitaka Kosaka, Hiroshi Yajima, and Norihisa Komoda. 1998. Application of neural network to technical analysis of stock market prediction. *Studies in Informatic and control*, 7(3):111–120.
Nabipour, Mojtaba, Pooyan Nayyeri, Hamed Jabani, Amir Mosavi, Ely Salwana, et al. 2020. Deep learning for stock market prediction. *Entropy*, 22(8):840.
Nakano, Masafumi, Akihiko Takahashi, and Soichiro Takahashi. 2017. Generalized exponential moving average (ema) model with particle filtering and anomaly detection. *Expert Systems with Applications*, 73:187–200.
Nazário, Rodolfo Toríbio Farias, Jéssica Lima e Silva, Vinicius Amorim Sobreiro, and Herbert Kimura. 2017. A literature review of technical analysis on stock markets. *The Quarterly Review of Economics and Finance*, 66:115–126.
Pang, Xiongwen, Yanqiang Zhou, Pan Wang, Weiwei Lin, and Victor Chang. 2020. An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, 76(3):2098–2118.
Pawar, Kriti, Raj Srujan Jalem, and Vivek Tiwari. 2019. Stock market price prediction using lstm rnn. In *Emerging Trends in Expert Applications and Security*. Springer, pages 493–503.
Roondiwala, Murtaza, Harshal Patel, and Shraddha Varma. 2017. Predicting stock prices using lstm. *International Journal of Science and Research (IJSR)*, 6(4):1754–1756.
Schumann, Matthias and T Lohrbach. 1993. Comparing artificial neural networks with statistical methods within the field of stock market prediction. In *[1993] Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences*, volume 4, pages 597–606, IEEE.
Selvin, Sreelekshmy, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647, IEEE.
Sherstinsky, Alex. 2020. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.
Shynkevich, Yauheniya, T Martin McGinnity, Sonya A Coleman, Ammar Belatreche, and Yuhua Li. 2017. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, 264:71–88.
Taran-Morosan, Adrian. 2011. The relative strength index revisited. *African Journal of Business Management*, 5(14):5855.
Thorp, Wayne A. 2000. Id'ing when to buy and sell using the stochastic oscillator. *AAII Journal*, pages 24–34.
Vargas, Manuel R, Carlos EM dos Anjos, Gustavo LG Bichara, and Alexandre G Evsukoff. 2018. Deep leaming for stock market prediction using technical indicators and financial news articles. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, IEEE.
Wang, Shouxiang, Xuan Wang, Shaomin Wang, and Dan Wang. 2019. Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting. *International Journal of Electrical Power & Energy Systems*, 109:470–479.
Yermack, David. 2015. Chapter 2 - is bitcoin a real currency? an economic appraisal. In David Lee Kuo Chuen, editor, *Handbook of Digital Currency*. Academic Press, San Diego, pages 31–43.
Yu, Yong, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270.
Zhu, Yingzi and Guofu Zhou. 2009. Technical analysis: An asset allocation perspective on the use of moving averages. *Journal of Financial Economics*, 92(3):519–544.