Universiteit van Amsterdam

Sparse trading strategies for high-frequency bitcoin trading

April 11, 2022

Student:
Caspar Hentenaar
12209066

Supervisor:
Noud van Giersbergen

Abstract

While bitcoin prices are on the rise, the bitcoin market is as volatile as ever. These wildly fluctuating prices might exhibit patterns to be detected using machine learning. In hopes of finding and profiting off such patterns, a Long Short-Term Memory (LSTM) network and a random forest are trained on recent, high-frequency, price data. Evaluating these predictions showed promise that the bitcoin market is indeed predictable to some degree. Combining the predictions of the random forest and LSTM network in a so-called ensemble, made trading based on these predictions more profitable when accounting for transaction fees. Introducing a threshold, which should be exceeded for trading to occur, further improved the profitability under transaction fees. However, at a realistic level of transaction fees, the ensembles were barely profitable and were outperformed by the buy and hold strategy.

1 Introduction

The inception of bitcoin came by the whitepaper of Nakamoto, 2009, a pseudonym for an author or group of authors who remain anonymous to this day. This paper laid the fundamentals for bitcoin as a digital, decentralized ledger and with that the fundamentals for all cryptocurrencies after it. The interest for bitcoin seems to be steadily increasing among the general public, as is implied by the increasing amount of search queries for the word "bitcoin", especially during price rallies (Google, 2021). Similarly, there seems to be an increase in academic attention as can be inferred from the work of Fang et al. (2020). They showed that over 85% of papers regarding cryptocurrency trading have been written since 2018.

The largest difference between bitcoin and everyday currencies might be the decentralized nature of bitcoin. Monetary policy, fraud detection, and facilitating (digital) payments are all tasks of a central authority, usually banks and governments in the case of regular currencies. For bitcoin, monetary supply increases at a publicly known geometrically decreasing pace (Browne, 2020). Fraud detection and prevention are replaced by proof of work requiring fraudsters to do an infeasible amount of computational work. Payments are verified by a decentralized network of "miners", computers working to generate a specific hash. The decentralized nature of the cryptocurrency market implies that there is no downtime, miners across the world are online 24/7, confirming transactions. This is in stark contrast to the strictly regulated stock markets which open and close during set times, only on weekdays.

Various aspects of bitcoin and bitcoin trading might make it an interesting subject of study. As Fang et al. (2020) note, the frequently occurring bitcoin price bubbles are a popular subject of interest. Furthermore, due to cryptocurrencies being a relatively new phenomenon institutional investors seem mostly absent yet. This absence of institutional investors could mean that there are or were large inefficiencies present in the cryptocurrency market. Past tense might be more appropriate as institutional investors are taking ever more interest in cryptocurrencies. The recent acquirement of bitcoin by Tesla (Tesla, 2021) is a great example of such an institutional investor, taking an interest.

Fang et al. (2020) show that 38.1% of papers in their survey are focused on predicting returns for cryptocurrencies. Note that predicting returns would only be possible in a market that is not efficient. Such a possibly inefficient market sparks interest in the application of machine learning when constructing a trading strategy. Machine learning might be able to derive patterns present in the erratic price behaviour of bitcoin, possibly non-perceptible to the human trader.

With the rise of computational power came advancements in machine learning. Exemplary of this are Long Short-Term Memory (LSTM) networks which are neural networks consisting of a sequence of cells. This structure seems to allow LSTM networks to better detect patterns in sequences. These networks have seen a much-increased use in various disciplines, for instance in the speech-to-text service: Google Voice (Beaufays, 2015). In addition to estimating more complex models, it is nowadays feasible to utilize ensemble learning. Ensemble learning combines machine learning methods by introducing voting principles to obtain a single classifier. This way, model performance increases through replication or combination of well-known methods, random forests for example are built by a large number of decision trees fitted to bootstrapped samples.

Although research on cryptocurrency trading is becoming ever more extensive some blanks are still observed in current literature. Bitcoin being the first and currently most established cryptocurrency seems to imply that most attention and funds of (institutional) investors are dedicated to bitcoin. Other crypto-currencies could therefore be promising for a trading strategy based on machine learning, due to a possible lack of professionalism in the market.

As mentioned, institutional investors seem to be entering the market for cryptocurrencies. As a consequence, the market might have become more efficient. If that is indeed the case the efficient market hypothesis prescribes that all information would be encompassed in the current price. Prediction of bitcoin prices using machine learning and trading based on this prediction would then be rendered useless. To construct a strategy that could be used for live trading, it would therefore be fruitful to consider recent data. This leads to another gap to be filled in, a strategy can only be viable for use in live trading if transaction fees are accounted for during backtesting. Leaving trading fees out of consideration is especially unrealistic since profits on individual trades are usually small in HFT. As individual profits are small and trades are frequent, trading fees accumulate quickly. Following this, more research could be devoted to the amount of trading to be done, or more specifically, when not to trade. As the market becomes more efficient trading in every period might not remain profitable due to a vanishing signal-to-noise ratio, forcing traders towards sparse trading strategies in which trades don't occur in every period but only occasionally.

This piece will try to fill in the mentioned gaps and provide a current trading strategy by answering the question: "How can random forests and LSTM networks be used and combined to construct a profitable sparse trading strategy for cryptocurrency when facing transaction costs?"

Answering this question might prove more important than ever as cryptocurrencies

could provide an invaluable opportunity for further diversification of portfolios. To make use of this opportunity, however, investors should be wary of the many artifacts present in the cryptocurrency market. The price bubbles that seem to occur every so often, or the occurrence of pump and dump schemes in which coins are bought and then quickly sold by online communities. Hoping that the next person will buy their coins for a higher price. The construction of a trading strategy that manages to remain profitable or even profit from such artifacts might prove vital for investors looking to enter the cryptocurrency market.

2 Theoretical Framework

When constructing a high-frequency trading strategy researchers and investors are faced with a daunting amount of choices. Generally, three different types of decisions can be distinguished: feature selection, modelling decisions, and the choice of evaluation framework.

Regarding the decision of which features to use, as price movements are to be predicted, naturally, price data is included. Extending hereupon might lead to the inclusion of technical indicators derived from this price data. Technical analysis has seen widespread use by investors in constructing trading strategies even though the efficient market hypothesis would suggest that results in the past carry no predictive value. Still, technical indicators are often included when attempting to predict cryptocurrency prices. McNally et al. (2018) even identified the 5 and 10 day Simple Moving Average (SMA) as the most important features in their analysis. Vo and Yost-Bremm (2020) also made use of technical indicators by including the following indicators when fitting a random forest classifier: Relative Strength Index (RSI), stochastic oscillator, Williams %R, Moving Average Convergence Divergence (MACD), and On Balance Volume (OBV). By using these indicators in conjunction with price data they seemed able to consistently generate excess returns compared to buy and hold strategies. These high returns were likely due to very accurate predictions of price movements, as indicated by an F1 score of 97% for predicting whether prices go up or down at a 15-minute frequency.

Many authors consider different features for inclusion in their models, such as gold spot prices, blockchain network characteristics, or data regarding search queries for cryptocurrencies. This might be for good reason, as Kristoufek (2013) found a correlation between bitcoin prices and online sentiment as measured by Google Trends data and Wikipedia search queries. Similarly, he found a significant relation between bitcoin prices and two bitcoin network characteristics: mining difficulty and hash rate in succeeding

research (Kristoufek, 2015). Moreover, Chen et al. (2020) chose to include gold spot price, Baidu and Google search volume for the word "bitcoin", and multiple bitcoin network characteristics after analyzing feature importance using the Boruta package. Boruta works similarly to a random forest classifier and can be used to evaluate which attributes are important (Kursa, Rudnicki, et al., 2010). In brief, a large variety of features might carry predictive value for bitcoin prices but some features might not be applicable at every frequency. For example, usually mining difficulty and hash rate are only available at a daily frequency. Taking the modelling decisions into account when selecting features might therefore prove essential.

Existing literature exhibits examples of different models that were fitted to predict bitcoin prices. As mentioned the frequency of trading affects the set of features which can be used, it seems that a similar effect is present regarding the choice for which model to use. Statistical methods like logistic regression, linear discriminant analysis, or generalized linear models outperform machine learning strategies when using data at a daily frequency according to the results of both Madan et al. (2015) and Chen et al. (2020). It might therefore be unsurprising that Sebastião and Godinho (2021) obtained an accuracy of 0.57 when fitting a random forest to daily data whereas the accuracy of statistical methods in Chen et al. (2020) was equal to 0.65 on average. At a 5 minute frequency, however, Chen et al. (2020) showed that machine learning techniques outperformed statistical methods, classifying price movements correctly in 62.2% of cases versus 53% for statistical methods. In light of these results, it seems reasonable to only consider machine learning strategies when constructing a high-frequency trading strategy.

Even when excluding statistical methods, many machine learning strategies to choose from remain. Most authors seem to opt for either a random forest or a recurrent neural network like a long short-term memory network. Other viable options might be support vector machines (Sebastião & Godinho, 2021) or gradient boosting classifiers (Chen et al., 2020). Vo and Yost-Bremm (2020) opted to fit a random forest on price data and some technical indicators to classify price movements, i.e. prices go up or go down during the next period. Their random forest was trained on multiple frequencies, from 1 minute up to 90 days. Their classifier achieved the highest F1 score when predicting price movements at a 15-minute frequency. Moreover, this classifier based on a random forest outperformed a multilayer perceptron network when considering F1 metrics. This is not very surprising as multilayer perceptron (MLP) networks seem generally regarded as ill-adapted for time series. It could therefore be more appropriate to test against more sophisticated neural networks, recurrent neural networks would probably be better suited for time series data. An example of such a more sophisticated recurrent neural network is employed by Chen

et al. (2020) who built a classifier by fitting a long short-term memory network for 5-minute bitcoin price data. Although this classifier outperformed a random forest fitted to the same data, it performed worse than the random forest classifier as in Vo and Yost-Bremm (2020). At the same 5-minute frequency Vo and Yost-Bremm (2020) managed to achieve an F1-score of 0.902 in contrast to the F1-score of 0.776 as achieved by the LSTM network in Chen et al. (2020). This difference might be due to the increased efficiency of the bitcoin market as Chen et al. (2020) use data from 2017-2018 whereas Vo and Yost-Bremm (2020) use data from 2015 up to and including 2017. This increased efficiency might also be inferred from the returns as specified by Vo and Yost-Bremm (2020), interestingly the returns yielded by their classifier decreased substantially since 2015. Annual returns for the random forest classifier were equal to 932%, 47%, and 205% for 2015, 2016, and 2017 respectively. All in all, it is hard to pick the "best method", it seems however that results of the past do not automatically generalize to the future, possibly due to a more efficient market.

A changing bitcoin market might mean that interest should go out to the application of machine learning strategies to an efficient market. Nelson et al. (2017) do just that by employing an LSTM network to predict the prices of exchange-traded funds on the Brazilian Bovespa (nowadays: B3) exchange. Nelson et al. (2017) fitted an LSTM network to 15-minute candlestick (OHLC) data and technical indicators derived from these candlesticks. Surprisingly even in this presumably highly efficient market, an accuracy was achieved of up to 55.9%. Moreover, trading based on this classification yielded relatively high returns per trade compared to the included baselines, which suggests that their strategy might have remained profitable, even when including trading costs. As predictions and trades are done at a 15-minute frequency trading costs accumulate rather quickly if returns per trade are insufficient. In an attempt to overcome the hurdle that transaction costs pose Resta (2006) devised a model that aimed to trade only on strong signals. This model thus had the ability to do nothing, in contrast to the strategy utilized by Nelson et al. (2017) which took either a long or a short position in every period. Resta (2006) fitted a neural network to the OHLCV data for the NASDAQ index at a 30-minute frequency. Evaluation on the test set showed that the constructed strategy remained profitable for transaction costs of up to 0.02%. Transaction costs on the cryptocurrency market are usually a magnitude of order larger. Transactions costs of 0.2% for opening and subsequently closing a position are not uncommon (Binance, 2021). Sebastião and Godinho (2021) go as far as including transaction fees of 0.5% in their calculations. Just like Resta (2006), they built a strategy in which trades take place only in the presence of strong signals. To this end, they constructed an ensemble model which combined a linear model, a random forest,

and a support vector machine. This ensemble only made a buy decision if each individual method predicted the same upwards price movement (short-selling was excluded). This differs from Resta (2006) who defined the problem as a three-class classification: prices go up, nothing happens, or prices go down. Sebastião and Godinho (2021) achieved an annualized return of 1.62% when applying their best performing ensemble on recent bitcoin price data, this was also the only ensemble of the three that managed to yield a positive return. Note however that the ensembles were evaluated in a down-trending market as the buy and hold strategy resulted in a return of -54.9%, the restriction of no short selling was therefore especially harsh. When applying the same ensemble on the market for ethereum however they achieved an annualized return of 14.35% after subtraction of transaction fees. Overall, these results suggest that even if the bitcoin market has become more efficient positive returns could still be realised if transaction costs remain low enough.

The large effect that the inclusion of trading fees has on the profitability of a trading strategy emphasises the need for a solid evaluation framework. Precision, accuracy, and recall statistics based on a confusion matrix give insight into the statistical side of performance. However, these statistics don't take the magnitude of price movements into account. Improving hereupon could be done by including cumulative returns as done by Sebastião and Godinho (2021) for example. Using another strategy as a baseline to compare the strategy of interest against might be considered the next step up. The Sharpe ratio as included by Vo and Yost-Bremm (2020) is an example of this, it measures if and how reliably a strategy outperforms risk-free assets like bonds. One might suggest however that in a market that grows as rapidly as the cryptocurrency market has done recently, the comparison with the risk-free rate yields a rather optimistic result. To provide a more reliable evaluation Nelson et al. (2017) test their obtained results against multiple baselines. The performance of their LSTM network is tested against two other trading strategies based on machine learning methods, namely a random forest and a multilayer perceptron (MLP) neural network. Besides, they also test their preferred strategy against more simple investment strategies: a buy and hold strategy, a naive forecast (if prices went up last period they will go up again), and a pseudo-random strategy (prediction based on class distribution). By extensively comparing the performance of their LSTM network against these alternatives they managed to make their results better interpretable. Due to the vanishingly small signal-to-noise ratio in stock data such extensive comparisons are necessary to make substantiated claims about the performance of a strategy.

The cited literature will be used when devising a design for this research. To this end, ensembles similar to the ensembles of Sebastião and Godinho (2021) will be employed.

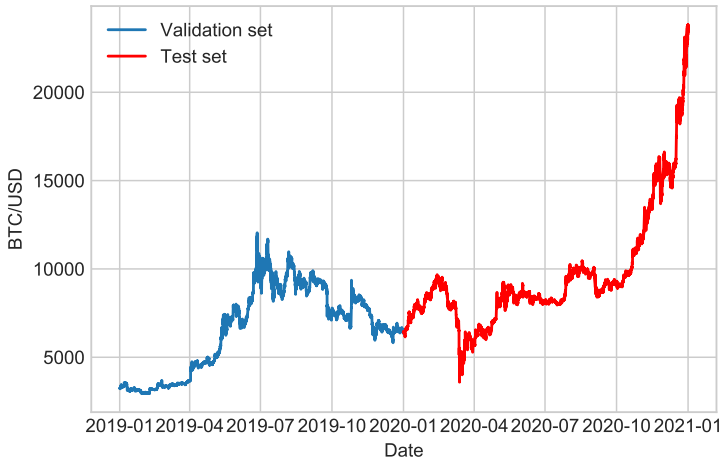
However, these ensembles will be built from different methods. As Vo and Yost-Bremm (2020) achieved a low classification error using random forests, these will be part of the ensemble. Furthermore, also an LSTM network will be included, as done by Chen et al. (2020), among others. Using these methods an ensemble is constructed to classify price movements in a way similar to Resta (2006). However, unlike Resta (2006), only long positions will be considered, much like Sebastião and Godinho (2021). The research design will be expanded upon further in the following section.

3 Research Design

3.1 Data and preprocessing

Tick by tick price data for bitcoin from 2014 until 2021 is obtained from the Kraken exchange. Kraken is a well-established exchange that has been online since 2011. Only one exchange is considered as Vo and Yost-Bremm (2020) show that ML methods are tolerant to differences between exchanges. As stated before, interest goes out to the profitability of constructed strategies in recent times due to the possibly increasingly efficient bitcoin market. Consequently, a subset of the data is made containing only the two most recent years, 2019-2021. Of these two years, the first one will be used as a validation set whereas the last year will be used for final evaluation. The bitcoin price development during the period from 2019 to 2021 is plotted in Figure 1.

Figure 1: 15-minute closing prices



Due to the use of a rolling window, not the entire last year is used for testing: the first

180 days of the data will only be used as the training set for the first iterations of the rolling window. This is illustrated in Figure 2.

Figure 2: First iteration of rolling window for evaluation

The tick by tick data is aggregated into Open-High-Low-Close-Volume (OHLCV) data at a 15-minute frequency. Returns are then derived as follows:

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}},$$

with p_t, the closing price at time t. A classifier is constructed which attempts to predict the direction of price movements and whether price movements are "large". For this purpose, the target variable is derived from the individual returns in a way similar to Resta (2006).

$$y_t = \begin{cases} 1 & \text{if} \quad r_t > \theta \\ 0 & \text{if} \quad r_t \leq \theta \end{cases}.$$

Here θ signifies a threshold value subject to $\theta \in \{0, 0.1, 0.2\}$. The inclusion of such a threshold is done in the hope that trades will only occur when price movements exceed the cost incurred from trading fees. Note that θ only affects the target variable on which the methods are trained. Here, τ is used for the level of transaction cost during profitability calculations. For example, the ensembles trained using $\theta = 0$ are evaluated for multiple values of τ. This approach allows conclusions regarding the profitability of the zero

threshold ensemble under various levels of transaction costs.

Multiple technical indicators are included as features, these technical indicators are derived from the OHLCV data by making use of the Pandas-TA package. Since McNally et al. (2018) identified the 5 and 10-day Simple Moving Average (SMA) as the most important features in predicting bitcoin prices, a simple moving average is included. Moreover, as Vo and Yost-Bremm (2020) constructed a profitable trading strategy by including the Relative Strength Index (RSI), stochastic oscillator, Williams %R, Moving Average Convergence Divergence (MACD), and On Balance Volume (OBV), these indicators are utilized here as well. Vo and Yost-Bremm (2020) mention that their classifier performs well under the default settings of these indicators, therefore, no further tweaking is done to indicator settings. Similar to Chen et al. (2020) gold spot price and trading volume is included. Lastly, all features are standardized as follows:

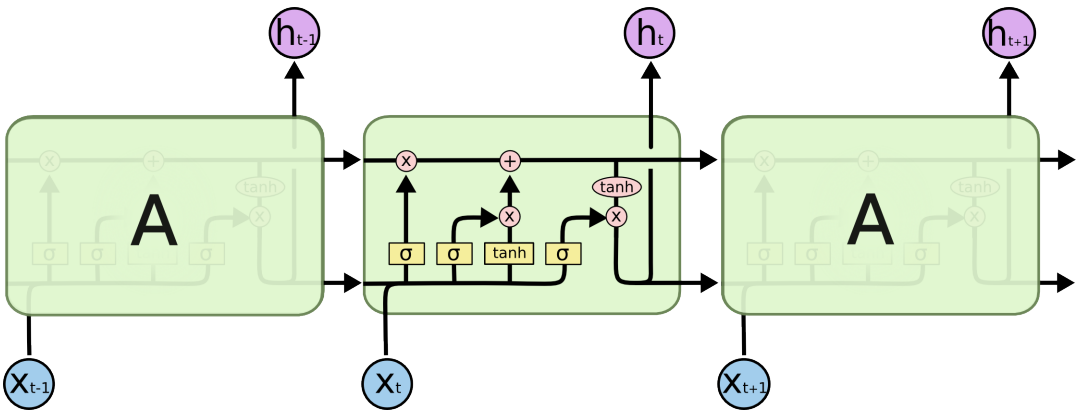
$$Z_{tj} = \frac{x_{tj} - \bar{x}_j}{s_j}, \quad t \in [1, n], j \in [1, k],$$

where Z_{tj} is the standard score for entry x_{tj}, feature j at time t. $\bar{x}_j$ and s_j are the sample mean and deviation for feature j respectively. n is the observation count and k the feature count. All this is as implemented by the "StandardScaler" in scikit-learn.

3.2 Machine learning methods

First Long Short-Term Memory (LSTM) networks are discussed, then random forests. Lastly, attention is payed to the topic of combining these two methods in an ensemble. LSTM networks are a type of Recurrent Neural Network (RNN) typically used when analyzing sequence data. Moreover, an LSTM network consists of a chained series of gated cells. A chain of three gated cells is illustrated in Figure 3.

Figure 3: LSTM cells, retrieved from Olah (2015), section "LSTM networks"



Each cell consists of a forget gate, a memory gate, and an output gate. The forget gate is trained to select which parts of the previous cell state are forgotten, and which will be put forward by remaining in the cell state. As stated by Olah (2015), the current input and previous cell output are put through a sigmoid function, which outputs a number ranging from 0 to 1 for each value in the cell state. A zero means disregard this value in the cell state completely, a 1 means that the value is propagated forward through the cell state, unchanged. The input gate enables the network to add to the current cell state which is then used in future periods. Lastly, the output gate combines current cell inputs and cell state to generate an output, in this case, a class. This summarizes the article of Olah (2015) which explains the inner workings of the LSTM cells in more detail.

Keras is used to implement the LSTM network, which is a deep learning framework running on top of Google's TensorFlow. Using this toolkit the LSTM network is compiled with 2 hidden layers, each containing 36 hidden units. A dropout rate of 0.2 is used, which means that 20% of inputs are randomly selected and set equal to 0. Furthermore, the LSTM is trained on samples that contain 96 timesteps, therefore, each sample contains all inputs of the 24 hours leading up to the newest observation in the training set. Lastly, at each iteration of the sliding window, the network is trained over 10 epochs, as this resulted in the lowest validation loss, see Figure 11 through 13. The dropout rate, amount of timesteps, and number of hidden units are selected by trial and error over a small subset of the training data. A grid search over the full training set, or a full grid search on this subset even, was computationally infeasible. In light of these computational limitations, the default activation functions were used, namely a tanh function in the output gate and a sigmoid function in the other gates. Using these default activations allowed the Keras package to train the network using a cuDNN (CUDA Deep Neural Network library) implementation, utilizing the GPU and drastically reducing training times. Lastly, the output of the LSTM layers are put through a sigmoid function, which yields a probabilistic value to be used for classification.

Random forests are an ensemble learning method based on decision trees. When fitting an RF many decision trees are built on various bootstrapped samples using various random sets of features. Once trained, data is run through all decision trees, after which results are averaged to obtain a classification. Due to this innate randomization, random forests are less prone to overfitting, as described more in-depth by Breiman (2001). Various values for the number of trees are considered. According to Breiman (2001), changes to this parameter may have a noticeable effect on results. To utilize random forests the implementation of scikit-learn is used.

Following the work of Sebastião and Godinho (2021) an ensemble is constructed by

combining an LSTM network with a random forest. Both are fitted on the same data, predicting whether there will be a large price increase and trading should occur, or not. Note that both the LSTM network as the random forest output a value between 0 and 1. the prediction of the ensemble is then given as below

$$\hat{y}_{t,1} = \begin{cases} 1 & \text{if} \quad \hat{y}_t^{\text{rf}} > 0.5 \wedge \hat{y}_t^{\text{lstm}} > 0.5 \\ 0 & \text{else} \end{cases}.$$

As both the LSTM network and the random forest give probabilistic outputs, an ensemble can also be formed by linearly combining predictions as follows:

$$\hat{y}_{t,2} = \begin{cases} 1 & \text{if} \quad \alpha \cdot \hat{y}_t^{\text{lstm}} + (1-\alpha) \cdot \hat{y}_t^{\text{rf}} > 0.5 \\ 0 & \text{if} \quad \alpha \cdot \hat{y}_t^{\text{lstm}} + (1-\alpha) \cdot \hat{y}_t^{\text{rf}} \leq 0.5 \end{cases}.$$

Where the value of α is set to maximize profit for $\tau = 0.08\%$, using the validation set.

A rolling window is used in which the LSTM network and random forest are repeatedly trained on a window that slides forward. Ideally, an approach as in Nelson et al. (2017) is adopted such that the network is trained daily. However, due to computational restrictions, the network and random forest are trained every 10 days here. The decision of how much historic data to include is more difficult. Hochreiter and Schmidhuber (1997) suggest that LSTM can pick up on long time lags of signals, LSTM might therefore be suited for the inclusion of more historic data. Since LSTM networks were designed with time dependencies in mind it seems reasonable to assume that RF might not be able to pick up on these long lags of signals as well. Therefore multiple training period lengths are considered, namely 30, 90 and, 180 days. The sliding window then consists out of respectively 30, 90, or 180 days of training followed by testing on 10 days of unseen data. After training and testing is completed, the sliding window moves ahead 10 days for the next iteration.

3.3 Evaluation

Evaluation of classifier performance will proceed by including accuracy, precision, recall, and F1 score. These are calculated as follows:

$$\begin{aligned}\text{precision} &= \frac{TP}{TP+FP},\\ \text{recall} &= \frac{TP}{TP+FN},\\ \text{accuracy} &= \frac{TP+TN}{n},\\ \text{F1} &= 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}},\end{aligned}$$

where n is the size of the test set, TP, FP, and FN denote the true positive count, false-positive count, and false-negative count respectively. Cumulative profits will be included and calculated as follows:

$$R_{\text{tot}} = \prod_{t=1}^{m} \left(1 + \hat{y}_t(r_t - \tau)\right),$$

with $\hat{y}_t$ the binary prediction at time t, r_t the realised return at time t, and τ the transaction fee. $\hat{y}_t$ takes on either the value 1 or 0, implying a buy or don't buy decision, respectively. The product of the prediction and the transaction fee subtracted from the realised return ensures that returns are positive only if a buy decision is made when returns exceed transaction costs. Besides the strategy based on the ensemble, investment strategies to compare against are included in a way similar to Nelson et al. (2017). To this end, a buy and hold strategy is utilized for which holds that:

$$R_{\text{tot}} = \frac{p_m^{\text{close}} - p_1^{\text{close}}}{p_1^{\text{close}}},$$

and a naive forecast is made for which predictions are given by

$$\hat{y}_t^{\text{naive}} = y_{t-1}.$$

The random forest and LSTM network trained using $\theta = 0$ are also used as a baseline.

Besides the comparisons described above we test the following hypotheses to determine whether the Sharpe ratio's (SR) of the strategies based on ensemble predictions are

significantly higher than the Sharpe ratio of the buy and hold strategy:

$$H_0 : SR_{\text{ens}} = SR_{\text{b\&h}}$$
$$H_a : SR_{\text{ens}} > SR_{\text{b\&h}},$$

where the (annualized) Sharpe ratio is given as in Sebastião and Godinho (2021):

$$SR = \frac{\mu}{\sigma}\sqrt{365}.$$

With μ the daily mean return and σ the standard deviation of said returns. These hypotheses are tested by making use of the following statistic and its asymptotic distribution under the null hypothesis as described by Bailey and Lopez de Prado (2012):

$$(\hat{SR}_{\text{ens}} - \hat{SR}_{\text{b\&h}}) \xrightarrow{\text{a}} N\left(0, \sigma^2_{\text{ens}} + \sigma^2_{\text{b\&h}}\right),$$

with

$$\sigma = \sqrt{\frac{1 + \frac{1}{2}SR^2 - \gamma_3 SR + \frac{\gamma_4 - 3}{4}SR_2}{m}},$$

where SR is the Sharpe ratio, m the size of the test set, γ_3 the skewness of daily returns, and γ_4 the kurtosis of daily returns.

By carrying out the research as specified in this section, the body of knowledge is expanded in a few ways. This research mainly extends upon the work of Sebastião and Godinho (2021) by applying an ensemble built from different machine learning methods to 15-minute data, instead of low-frequency (daily) data. Current research concerning cryptocurrency trading seems to focus on predicting price movements such that trading occurs in every period. This ensemble might be used in the construction of sparse trading strategies as a threshold is implemented, similar to Resta, 2006. Lastly, an attempt is made to provide handles to (institutional) investors looking to enter the cryptocurrency market by applying the constructed ensemble in a realistic and current setting. This way, a strategy that performs well during backtesting might be used for live trading.

Results

All reported results are obtained using a sliding window containing 180 days of training data, unless stated otherwise. The methods using 90 or 30 days of historical data per-

formed similarly or inferior to methods using 180 days on the validation set, as can be observed in the appendix. Some further results using 90 or 30 days of historical data are included in the appendix. The random forest was generated on the validation set using 50, 100, 200, 500, and 1000 trees. Changing the amount of trees did not seem to affect performance much. Furthermore, as Vo and Yost-Bremm (2020) seemed to obtain the best results using 100 trees, all results here are obtained using 100 trees. Some results regarding the choice for the number of trees are reported in the appendix.

The ensemble constructed using a weighted average of the RF and LSTM predictions only outperformed the individual methods for $\theta = 0.2$. At the other thresholds, profit was highest when $\alpha = 1$, resulting in a pure LSTM. Therefore, "Weighted avg ensemble", refers to the ensemble trained using $\theta = 0.2$ and constructed using $\alpha = 0.58$. See Figure 8 through 10 for these results.

First statistics regarding the classification performance will be reviewed. Then, the profitability of the constructed strategies is considered. Lastly, some statistics are reported regarding financial performance.

Tables 1 through 3 provide classification reports for the ensemble, the individual methods, and the naive forecast for the three thresholds. Increasing the threshold leads to imbalanced datasets. Therefore, the accuracy is less informative for the methods trained using a non-zero threshold, of which the classification reports are shown in Table 2 and Table 3. To evaluate performance based on these classification results, the context of the problem needs to be considered. False negatives mean that no buy decision is made, even though it should have. The cost incurred as a consequence of this mistake is zero, not buying does not carry a cost. On the other hand, false positives are wrongly made buy decisions. These mistakes are very costly as a buy decision is made while returns might be negative, or insufficient to offset transaction costs. Therefore, the key metrics to consider are precision for the positive class and recall for the negative class. High precision for the positive class means that in periods for which a price increase is predicted, this price increase is likely to materialize. Similarly, a high recall for the negative class indicates a low number of wrongly predicted buy decisions, compared to the support of the negative class. Using the table, comparisons can be drawn about the ensemble trained using a zero threshold and the individual methods used in the construction of this ensemble. For all three thresholds the ensemble outperforms the individual methods, in terms of precision. The recall for the negative class is also higher for all ensembles. As expected the ensembles achieve lower recall for the positive class, indicative of more false negatives. Thus, the ensembles are able to predict a price increase more precisely, even though picking up on less of the price increases compared to the individual methods.

All methods outperform the naive forecast on the test set, taking into account both accuracy and positive class precision.

Table 1: Classification reports, $\theta = 0$

Method	class	precision	recall	f1-score	observation count	accuracy
Ensemble	0	0.51	0.73	0.60	8744	0.519
	1	0.55	0.32	0.40	9112	
LSTM	0	0.52	0.52	0.52	8744	0.534
	1	0.54	0.55	0.54	9112	
RF	0	0.51	0.57	0.54	8744	0.518
	1	0.53	0.47	0.50	9112	
Naive forecast	0	0.47	0.47	0.47	8744	0.479
	1	0.49	0.49	0.49	9112	

Table 2: Classification reports, $\theta = 0.1$

Method	class	precision	recall	f1-score	observation count	accuracy
Ensemble	0	0.71	0.99	0.83	12621	0.71
	1	0.57	0.05	0.08	5235	
LSTM	0	0.72	0.97	0.82	12621	0.709
	1	0.52	0.08	0.14	5235	
RF	0	0.72	0.89	0.80	12621	0.681
	1	0.40	0.17	0.24	5235	
Naive forecast	0	0.71	0.71	0.71	12621	0.597
	1	0.31	0.31	0.31	5235	

Table 3: Classification reports, $\theta = 0.2$

Method	class	precision	recall	f1-score	observation count	accuracy
Ensemble	0	0.84	1.00	0.91	14998	0.84
	1	0.50	0.02	0.04	2858	
Weighted avg ensemble	0	0.84	0.99	0.91	14998	0.84
	1	0.50	0.04	0.08	2858	
LSTM	0	0.84	0.99	0.91	14998	0.839
	1	0.47	0.04	0.07	2858	
RF	0	0.85	0.97	0.91	14998	0.831
	1	0.38	0.09	0.15	2858	
Naive forecast	0	0.85	0.72	0.78	14998	0.66
	1	0.19	0.35	0.25	2858	

From Table 4 it can be seen that as the threshold during training (θ) increases, the time in the market drastically reduces. Furthermore, the LSTM network takes a long position less often than the random forest for $\theta > 0$. A figure comparing the profitability of these individual methods is included in the appendix, see Figure 7.

The predictions of the ensembles (excluding the weighted average ensemble) are given by the intersection between the predictions of the LSTM network and the predictions of the RF. Naturally, the ensembles enter the market less than the individual methods for every value of θ.

Table 4: % of time in the market

θ	Ensemble	LSTM	RF
0	29.56%	48.58%	44.88%
0.1	2.48%	4.31%	12.59%
0.2	0.73%	1.19%	3.87%

In Table 5 the accumulated value on a hypothetical investment of 100$ is displayed. The buy and hold strategy outperforms all other methods, even when assuming transaction fees to be zero. Furthermore, if transaction fees are assumed to be zero, the ensembles perform worse than the individual methods. When transaction costs are positive the ensembles outperform the individual methods. Furthermore, the ensemble trained using $\theta = 0.1$ remains profitable for the highest level of transaction costs. The naive forecast performs worst, only exceeding profits of some ensembles when trading fees are assumed to be zero.

Table 5: Profit under transaction costs

Method	tc=0	tc=0.05%	tc=0.1%
Ensemble, $\theta = 0$	213.20	15.22	1.08
Ensemble, $\theta = 0.1$	169.29	135.68	108.73
Ensemble, $\theta = 0.2$	110.33	103.33	96.78
Weighted avg ensemble	127.75	112.52	99.11
LSTM, $\theta = 0$	252.25	3.30	0.04
RF, $\theta = 0$	257.57	4.68	0.08
Naive forecast	157.40	1.65	0.02
Buy and hold	289.68	-	-

Below, in Figure 4, the accumulated value over the test set of a 100$ investment is plotted against transaction costs (τ). Besides the ensembles, the LSTM network without threshold is included as a baseline. The horizontal dotted line indicates the starting value, 100 dollars. Once again, it is apparent that the ensemble trained using $\theta = 0.1$ remains profitable for the highest level of transaction costs. The methods trained using a zero threshold quickly return a loss as transaction costs exceed 0.01%. Note that realistic roundtrip transaction fees are 0.2% on the Kraken exchange and at least 0.08% on Binance. At transaction fees of 0.08% only the ensemble trained using $\theta = 0.1$ is profitable, none of the methods are profitable for transaction fees of 0.2%.

Figure 4: Value accumulated versus transaction fees

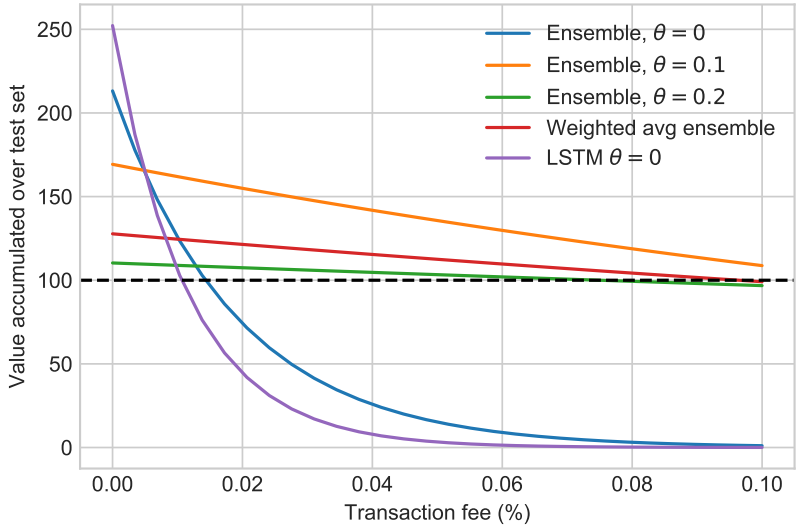


Table 6 contains performance statistics calculated under the assumption of no trans-

action fees. The maximum drawdown is the largest difference between a peak and subsequent through, conveying a sense of tail risk. The ensembles have a smaller maximum drawdown than the other methods. Furthermore, the ensembles attain a higher mean profit per trade compared to the alternatives. The annualized Sharpe ratio relates the return of a strategy to its volatility. In this regard, the ensemble using $\theta = 0.1$ performs best, achieving a Sharpe ratio of 7.5. Volatility is included as the standard deviation of daily percentage returns.

The hypotheses as stated under the Research Design section are tested:

$$H_0 : SR = SR_{\text{b\&h}}$$
$$H_a : SR > SR_{\text{b\&h}},$$

The buy and hold strategy is taken as baseline. The Sharpe ratio's of the ensembles trained using $\theta = 0$ or $\theta = 0.1$ exceed that of the buy and hold strategy at a 1% confidence level, just like the LSTM based strategy using $\theta = 0.1$. Furthermore, the Sharpe ratio of the random forest trading strategy using $\theta = 0$, exceeds that of the buy and hold at a 5% confidence level.

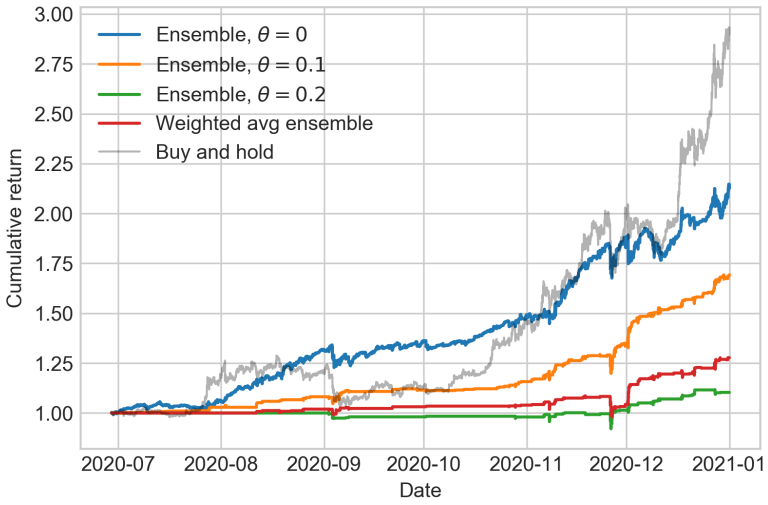
Table 6: Performance statistics

Method	Max. Drawdown	Mean profit per trade	Sharpe Ratio	Std. deviation (%)
Ensemble, $\theta = 0$	-9.0%	0.0148%	5.5***	1.45
Ensemble, $\theta = 0.1$	-7.0%	0.1209%	7.5***	0.73
Ensemble, $\theta = 0.2$	-8.0%	0.078%	2.5	0.40
Weighted avg ensemble	-10.0%	0.0994%	2.9	0.88
LSTM, $\theta = 0$	-16.0%	0.0112%	5.3***	1.87
RF, $\theta = 0$	-12.0%	0.0122%	5.4**	1.88
Naive forecast	-22.0%	0.0054%	2.4	2.10
Buy and hold	-20.0%	0.0064%	4.2 (baseline)	2.80

(*) p=0.1, (**) p=0.05, (***) p=0.01

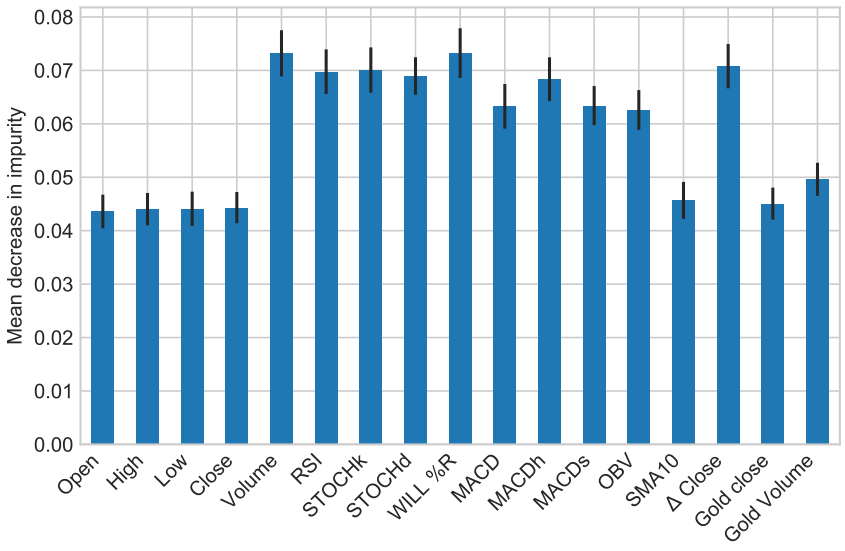
Figure 5 shows the cumulative returns of the ensembles as well as the buy and hold strategy, assuming transaction fees to be 0. The ensembles trained using positive values of θ seem largely unaffected by price fluctuations exhibited in the buy and hold strategy. Their eventual return is lower, however.

Figure 5: Cumulative returns using various thresholds



In Figure 6 feature importances are given, calculated by the mean impurity decrease over all trees of the random forest. The standard deviation of the impurity decrease over all trees is illustrated by the error bars. Most indicators attain a higher feature importance compared to the price data of the current period. Trade volume in the current period also seems more important than the candlestick data. Gold closing price and trade volume provide a significant, positive, mean decrease in impurity, although being less influential than most indicators.

Figure 6: Feature importance calculated by mean decrease in impurity



4 Conclusion and Discussion

A random forest and an LSTM network were combined in an ensemble which took a long position only if both methods agreed to do so. Such ensembles yielded worse returns under the assumption of no trading fees, upon inclusion of said trading fees the ensembles outperformed the individual methods. Furthermore, the ensembles exhibited lower volatility than the individual methods. Besides combining methods to form an ensemble, reducing the number of trades made, a threshold was introduced. Using this threshold the methods were trained to make a buy decision only if the price increase was sufficiently large, further reducing the amount of trading done. This proved to be a valuable tool for increasing the average profit per trade. The far reduced number of trades in the strategies which combined the ensemble with such a threshold did result in lower returns. Under realistic transaction costs only the ensemble trained using $\theta = 0.1$ yields a slight profit. However, this ensemble is still outperformed by the buy and hold strategy.

These results are largely in line with the findings of Sebastião and Godinho (2021), who found that most ensembles generated a loss when used for bitcoin trading. On the other hand, Vo and Yost-Bremm (2020) constructed an extremely profitable strategy using just a random forest. Their classifier achieved an F1 score of 0.901 on historic data. These results could not be reproduced while using recent data. This could be indicative of a now more developed, more efficient bitcoin market.

Some limitations of the current research should be noted. All strategies were tested from 29/6/2020 until 1/1/2021, in this period the bitcoin price increased rather rapidly with only a few dips. As strategies were only able to take a long position this growth might have positively inflated results, as seen from the naive forecast generating a positive return. However, the high mean return and low maximum drawdown of the ensembles show that the predictions are better than random. Moreover, the results show that even in periods of market downturn the ensembles do not incur large losses.

Another limitation is the mere use of trial and error for setting the LSTM network parameters, instead of a grid search over all parameters. The ensembles may therefore still perform sub-optimal. Since training an LSTM network using 180 days of historical data already took several hours, a grid search was deemed infeasible. More extensive parameter optimization could result in performance gains.

The results allow for future research in various directions. A natural extension could be the inclusion of more individual methods in the construction of ensembles. Other neural networks come to mind, like the LSTM fully convolutional network (LSTM-FCN), shown to perform outstandingly on time series classification tasks (Karim et al., 2019). Another

angle might be extending the classification task by allowing short selling, extending the problem to a 3-class classification. Furthermore, if more computational power can be acquired, one could consider including more historical data and performing a grid search for the LSTM parameters. Lastly, as the main issue of the strategies posed here seems to be the accumulation of transaction costs, it might be interesting to lower the frequency of trading. As trading is done at lower frequencies, transaction costs pose less of an issue.

The results have some practical implications. Bitcoin trading, using ensembles of random forests and LSTM networks, seems barely profitable for individuals, as the transaction fees often exceed the average profit per trade. Such an ensemble might still be viable for institutional investors, as trading in large volumes tends to lower transaction fees, due to tiered pricing (Binance, 2021). The average profit per trade may therefore exceed transaction fees for corporate investors.

For risk-averse corporations, an approach as employed here might be useful. The ensembles seemed to perform consistently even when the market fluctuated wildly, as indicated by the maximum drawdown of 7% for one of these ensembles.

Regarding the research question: "How can random forests and LSTM networks be used and combined to construct a profitable sparse trading strategy for cryptocurrency when facing transaction costs". The findings in this paper suggest that it is possible to create a slightly profitable trading strategy while accounting for a realistic, although low level of transaction fees. In practice however, trading at this low level of transaction fees is only possible for corporations that are able to trade large volumes. Improvements are needed to make automated trading using LSTM-RF ensembles viable for individuals. Such improvements can possibly be made by including more historical data, using a larger variety of machine learning methods in the construction of the ensembles, or further tweaking the LSTM and RF used here.

5 Appendix

Table 7: LSTM classification reports for different rolling window sizes

Method	class	precision	recall	f1-score	observation count	accuracy
180 days, $\theta = 0$	0	0.52	0.54	0.53	8744	0.529
	1	0.54	0.51	0.53	9112	
90 days, $\theta = 0$	0	0.51	0.53	0.52	8744	0.522
	1	0.53	0.52	0.53	9112	
30 days, $\theta = 0$	0	0.50	0.54	0.52	8744	0.514
	1	0.53	0.49	0.50	9112	
180 days, $\theta = 0.1$	0	0.72	0.97	0.82	12621	0.709
	1	0.52	0.08	0.13	5235	
90 days, $\theta = 0.1$	0	0.72	0.95	0.82	12621	0.698
	1	0.43	0.09	0.14	5235	
30 days, $\theta = 0.1$	0	0.71	0.96	0.82	12621	0.696
	1	0.40	0.07	0.12	5235	
180 days, $\theta = 0.2$	0	0.84	0.99	0.91	14998	0.839
	1	0.47	0.04	0.07	2858	
90 days, $\theta = 0.2$	0	0.84	0.99	0.91	14998	0.837
	1	0.41	0.05	0.08	2858	
30 days, $\theta = 0.2$	0	0.84	0.99	0.91	14998	0.836
	1	0.32	0.02	0.04	2858	

Table 8: RF classification reports for different rolling window sizes

Method	class	precision	recall	f1-score	observation count	accuracy
180 days, $\theta = 0$	0	0.51	0.57	0.54	8744	0.518
	1	0.53	0.47	0.50	9112	
90 days, $\theta = 0$	0	0.51	0.55	0.53	8744	0.522
	1	0.53	0.50	0.51	9112	
30 days, $\theta = 0$	0	0.50	0.58	0.54	8744	0.511
	1	0.52	0.44	0.48	9112	
180 days, $\theta = 0.1$	0	0.72	0.89	0.80	12621	0.681
	1	0.40	0.17	0.24	5235	
90 days, $\theta = 0.1$	0	0.72	0.86	0.79	12621	0.668
	1	0.38	0.20	0.26	5235	
30 days, $\theta = 0.1$	0	0.72	0.83	0.77	12621	0.655
	1	0.36	0.23	0.28	5235	
180 days, $\theta = 0.2$	0	0.85	0.97	0.91	14998	0.831
	1	0.38	0.09	0.15	2858	
90 days, $\theta = 0.2$	0	0.85	0.96	0.90	14998	0.822
	1	0.34	0.11	0.17	2858	
30 days, $\theta = 0.2$	0	0.85	0.93	0.89	14998	0.803
	1	0.29	0.16	0.21	2858	

Table 9: Profit of ensembles by rolling window size

Method	tc=0	tc=0.05%	tc=0.1%
180 days, $\theta = 0$	213.20	15.22	1.08
90 days, $\theta = 0$	226.19	14.64	0.95
30 days, $\theta = 0$	181.81	14.73	1.19
180 days, $\theta = 0.1$	169.29	135.68	108.73
90 days, $\theta = 0.1$	136.68	102.12	76.30
30 days, $\theta = 0.1$	117.16	93.74	75.00
180 days, $\theta = 0.2$	110.33	103.33	96.78
90 days, $\theta = 0.2$	117.85	107.50	98.05
30 days, $\theta = 0.2$	98.74	93.13	87.83

Table 10: Performance statistics for ensembles by rolling window size

Method	Max. Drawdown	Mean profit per trade	Sharpe Ratio	Std. deviation (%)
180 days, $\theta = 0$	-9.0%	0.0148%	5.5***	1.45
90 days, $\theta = 0$	-11.0%	0.0154%	6.5***	1.33
30 days, $\theta = 0$	-14.0%	0.0123%	4.3	1.49
180 days, $\theta = 0.1$	-7.0%	0.1209%	7.5***	0.73
90 days, $\theta = 0.1$	-8.0%	0.0552%	4.3	0.76
30 days, $\theta = 0.1$	-9.0%	0.037%	2.5	0.68
180 days, $\theta = 0.2$	-8.0%	0.078%	2.5	0.40
90 days, $\theta = 0.2$	-8.0%	0.0925%	2.4	0.72
30 days, $\theta = 0.2$	-10.0%	-0.0085%	-0.2	0.59

Table 11: Classification results of RF with different # of trees, on validation set

Method	class	precision	recall	f1-score	observation count	accuracy
50 trees	0	0.53	0.50	0.51	9023	0.518
	1	0.51	0.53	0.52	8737	
100 trees	0	0.53	0.48	0.50	9023	0.52
	1	0.51	0.56	0.53	8737	
200 trees	0	0.53	0.47	0.50	9023	0.522
	1	0.51	0.58	0.54	8737	
500 trees	0	0.53	0.45	0.49	9023	0.52
	1	0.51	0.59	0.55	8737	
1000 trees	0	0.54	0.45	0.49	9023	0.524
	1	0.51	0.60	0.55	8737	

Table 12: Profitability of RF with different # of trees, on validation set

Method	tc=0	tc=0.01%	tc=0.05%
50 trees	101.43	40.61	1.04
100 trees	107.67	41.26	0.89
200 trees	109.54	40.76	0.78
500 trees	94.30	34.12	0.58
1000 trees	116.43	41.78	0.69

Figure 7: Value accumulated versus transaction fees for RF and LSTM, 180 days of training

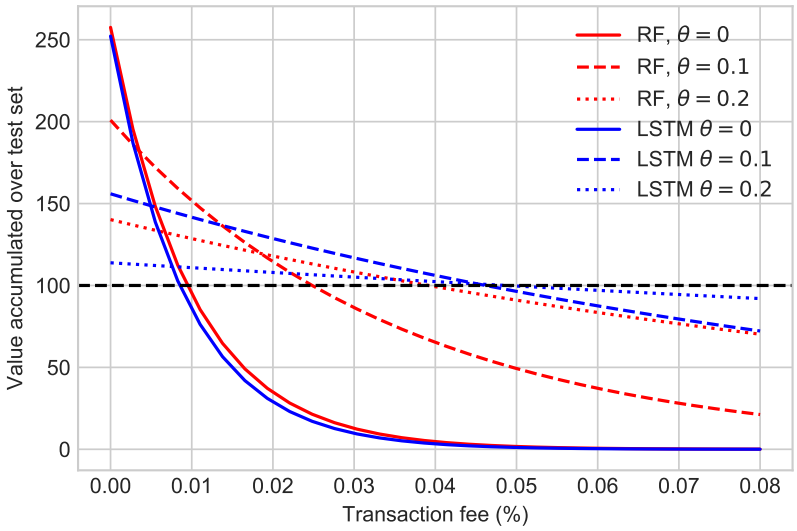


Figure 8: Accumulated value of 100 dollars, invested over training set, $\theta = 0$, $\tau = 0.08\%$

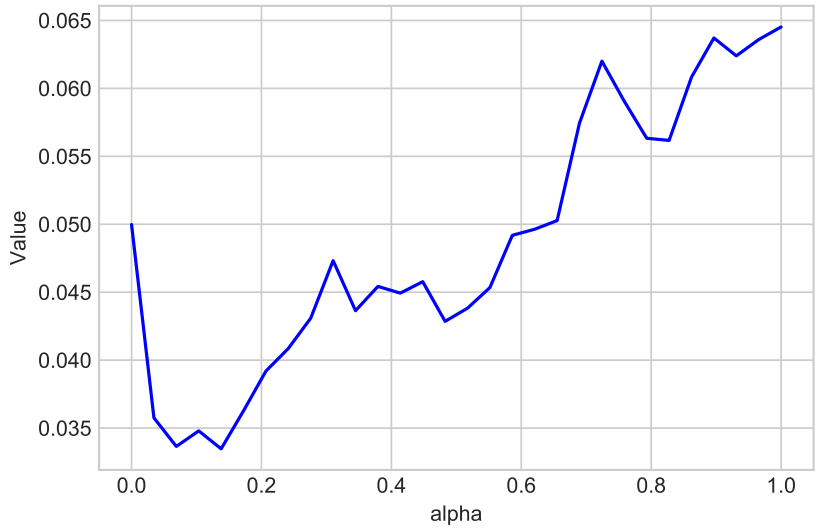


Figure 9: Accumulated value of 100 dollars, invested over training set, $\theta = 0.1$, $\tau = 0.08\%$

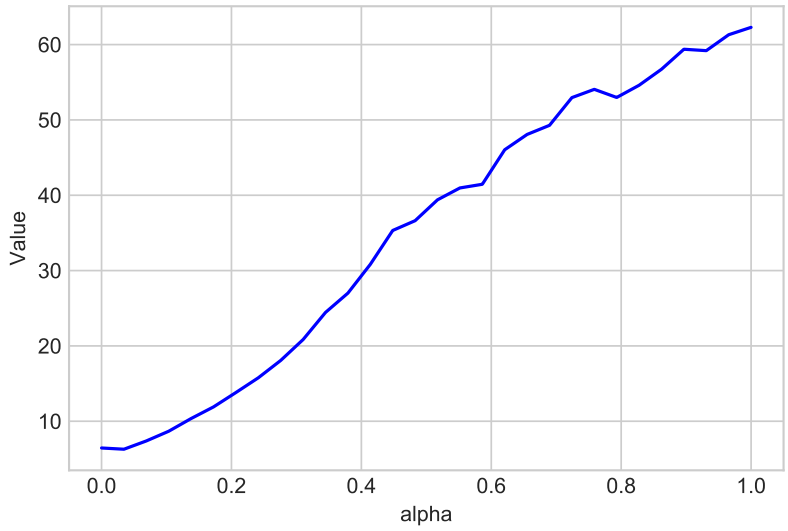


Figure 10: Accumulated value of 100 dollars, invested training set, $\theta = 0.2$, $\tau = 0.08\%$

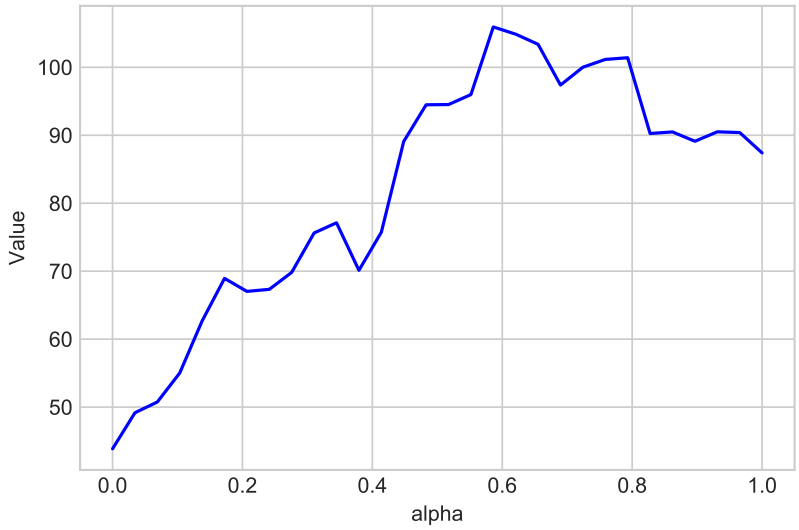


Figure 11: Loss and validation loss versus epoch, $\theta = 0$

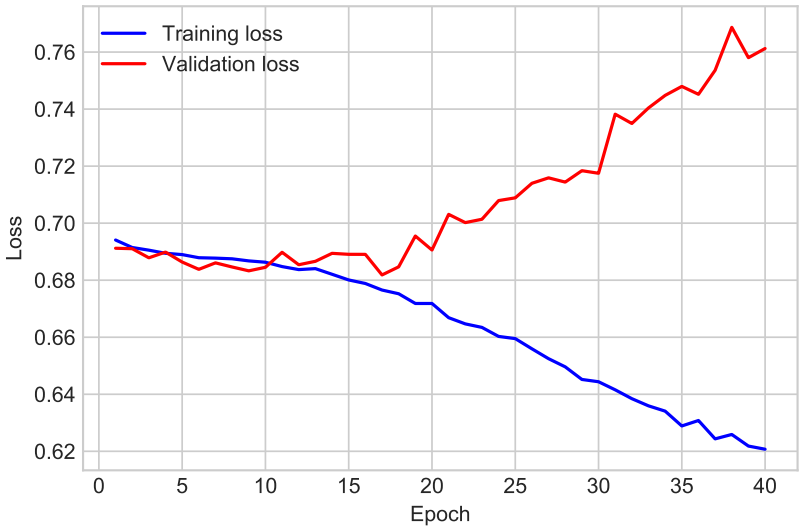


Figure 12: Loss and validation loss versus epoch, $\theta = 0.1$

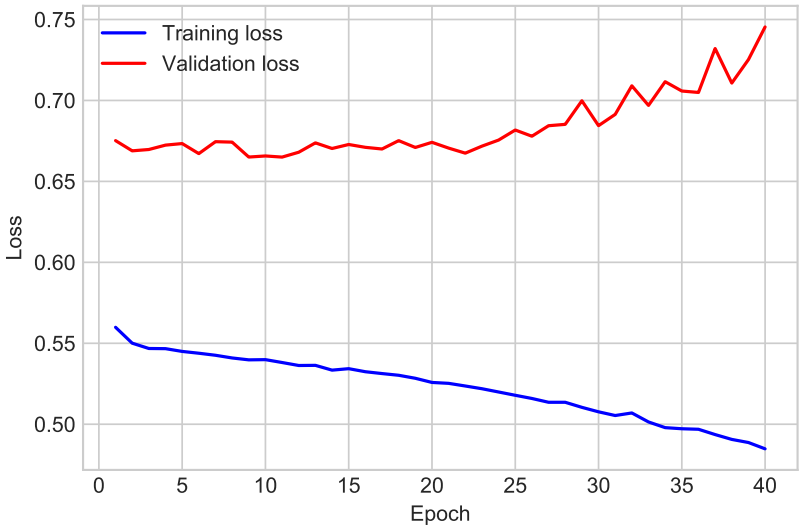
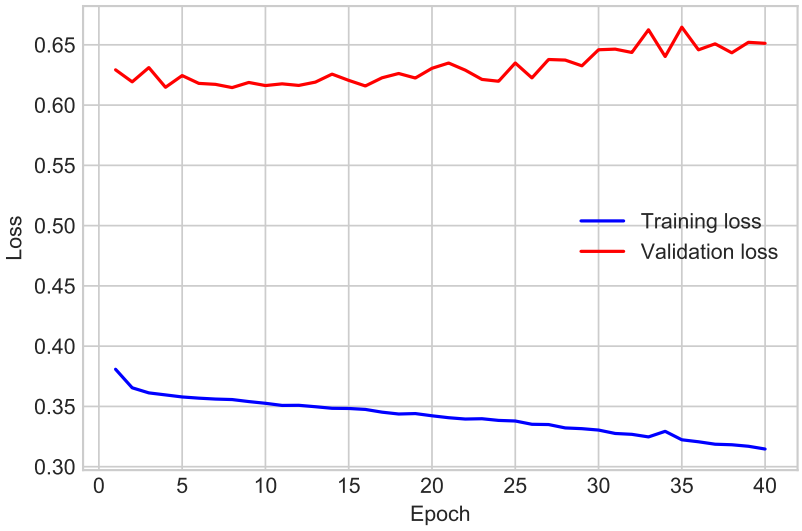


Figure 13: Loss and validation loss versus epoch, $\theta = 0.2$



References

Bailey, D. H., & Lopez de Prado, M. (2012). The sharpe ratio efficient frontier. *Journal of Risk*, *15*(2), 13.

Beaufays, F. (2015). *The neural networks behind google voice transcription*. https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html

Binance. (2021). *Trading fees*. https://www.binance.com/en/fee/schedule

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Browne, R. (2020). *Bitcoin investors are bracing for a key technical event — here's what you need to know*. https://www.cnbc.com/2020/05/11/bitcoin-halving-heres-what-you-need-to-know.html

Chen, Z., Li, C., & Sun, W. (2020). Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied Mathematics*, *365*, 112395.

Fang, F., Ventre, C., Basios, M., Kong, H., Kanthan, L., Li, L., Martinez-Regoband, D., & Wu, F. (2020). Cryptocurrency trading: A comprehensive survey. *arXiv preprint arXiv:2003.11352*.

Google. (2021). *Google trends*. https://trends.google.nl/trends/explore?date=all&q=Bitcoin

Hochreiter, S., & Schmidhuber, J. (1997). Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 473–479.

Karim, F., Majumdar, S., Darabi, H., & Harford, S. (2019). Multivariate lstm-fcns for time series classification. *Neural Networks*, *116*, 237–245.

Kristoufek, L. (2013). Bitcoin meets google trends and wikipedia: Quantifying the relationship between phenomena of the internet era. *Scientific reports*, *3*(1), 1–7.

Kristoufek, L. (2015). What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis. *PloS one*, *10*(4), e0123923.

Kursa, M. B., Rudnicki, W. R. et al. (2010). Feature selection with the boruta package. *J Stat Softw*, *36*(11), 1–13.

Madan, I., Saluja, S., & Zhao, A. (2015). Automated bitcoin trading via machine learning algorithms. *URL: http://cs229.stanford.edu/proj2014/Isaac%20Madan*, *20*.

McNally, S., Roche, J., & Caton, S. (2018). Predicting the price of bitcoin using machine learning. *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)*, 339–343.

Nakamoto, S. (2009). Bitcoin. *bitcoin.org*.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with lstm neural networks. *2017 International joint conference on neural networks (IJCNN)*, 1419–1426.

Olah, C. (2015). Understanding lstm networks.

Resta, M. (2006). On the profitability of scalping strategies based on neural networks. *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, 641–646.

Sebastião, H., & Godinho, P. (2021). Forecasting and trading cryptocurrencies with machine learning under changing market conditions. *Financial Innovation*, *7*(1), 1–30.

Tesla. (2021). *Q1 quarterly disclosure*. https://tesla-cdn.thron.com/static/R3GJMT_TSLA_Q1_2021_Update_5KJWZA.pdf?xseo=&response-content-disposition=inline%5C%3Bfilename%5C%3D%5C%22TSLA-Q1-2021-Update.pdf%5C%22

Vo, A., & Yost-Bremm, C. (2020). A high-frequency algorithmic trading strategy for cryptocurrency. *Journal of Computer Information Systems*, *60*(6), 555–568.