ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Master Thesis Quantitative Finance

Improving machine learning ensemble models with statistical techniques in the forecasting of cryptocurrency prices

Author:
Jasper Hu 490177

Supervisor:
Prof. Dr. DJC van Dijk
Second assessor:
Dr. A Tetereva

July 13, 2021

Contents

1 Introduction

Experts have applied quantitative predictive methods to pursue profits since the emergence of financial markets in the seventeenth century. Existing quantitative statistical methods based on statistical models that have been proven to work form part of the strategies practised in these financial markets. The digital age gave rise to the cryptocurrency industry and vast computational power to process large amounts of data. The latter has allowed rapid development in machine learning (ML), which gave rise to more new predictive methods.

However, new complex methods may not always lead to better (unbiased) results when compared to simple existing methods (Makridakis et al., 2018a). For example, results may be biased to a specific (part of the) dataset, or the derived results may be disputable. Complex machine learning methods also struggle with unexplainability and incomprehensibility, which may cause a preference for existing explainable statistical models. Conversely, complex machine learning methods offer more flexibility and adaptability, possibly resulting in a better fit for the entire dataset.

In this paper, the primary goal is to provide insights on the effect of blending particular statistical techniques with different ensemble techniques and machine learning models in a highly volatile, and thus, challenging market environment. This goal leads to the research question: "*Can machine learning ensemble models be improved with statistical techniques in forecasting cryptocurrency prices?*". This research question is answered by investigating the effect of different ensemble techniques on a mixture of machine learning models, which is later extended by adding and combining the basis with statistical techniques and models. In particular, the model's objective is to forecast the cryptocurrency price of Bitcoin in the following period, and all other historical data features are used as independent variables. Furthermore, in the evaluation, a combined dataset is used with features to capture the macro-economic effects, as well as multiple other cryptocurrencies, described in Section 2. The forecasts are assessed based on recent historical market data using the arithmetic mean of the SMAPE and MASE, explained in Section 4.3, as using multiple accuracy metrics yields a lower bias in measuring the performance.

As mentioned previously, the field of cryptocurrencies offers another problem of high volatility (Yi et al., 2018). This phenomenon can be primarily explained by the low market capitalisation, resulting in the possibility for an investor or a collective of investors to influence the market price significantly. High volatility results in more unpredictability due to fatter tails and increases the difficulty of fitting a model to the data. This phenomenon conventionally requires a more flexible model, where the noise may taint conventional econometric models' performance (Bakar and Rosbi, 2017). Khedr et al. (2021) showed in their survey that individual statistical methods experience difficulties in forecasting cryptocurrency prices due to the requirement of many statistical assumptions.

Existing studies in different areas compare the power of the novel machine learning models against the sophisticated econometric models in forecasting. However, these studies are contradictory as they may argue in favour of either set of models (Ahmed et al., 2010; Makridakis et al., 2018b). Ensemble models, which combine multiple individual models, have recently been shown to outperform individual methods, for example, by averaging the forecasts of multiple individual models (Makridakis et al., 2018a). In the M5 forecasting competition, the top-performing models also consisted of ensemble (ML) models, outperforming models with a purely statistical basis (Makridakis et al., 2020). Although this competition's focal point was not forecasting stock prices, it still shows the potential of novel ensemble models with a basis in ML and thus provides another reason for the choice of machine learning ensemble models as a basis for this research. In the context of cryptocurrencies, Borges and Neves (2020); Livieris et al. (2020) have shown promise in forecasting the price movements using pure ensemble ML methods.

Although there exist studies on using ensemble methods in forecasting, they mainly utilise either machine learning methods or statistical methods, but not a combination of both. This phenomenon is not uncommon; existing literature has often drawn a clear boundary between statistical methods and machine learning methods, where the statistical side assumes that the data is generated by a stochastic model, while the AI side considers the data mechanisms as unknown (Breiman et al., 2001). Januschowski et al. (2020) explain in their research that although ensemble models exist and there should be no clear methodological distinction between these two fields, combinations of these two fields are still considerably unexplored, and particularly in the field of financial data. Therefore, this paper is likely to be a valuable contribution to both fields, as investigating the mixture of both AI and statistics could provide new insights in either field.

This research has achieved satisfactory performance. Using a mixture of statistical and machine learning techniques, and with the correct set of hyperparameters, given the data up until an interval t, and the goal of forecasting the BTC/USDT pair's opening price of a subsequent interval at time $t+1$, an increase in forecasting ability has been achieved of approximately 14.2%. Aside from the increase in forecasting ability, the robustness of the predictions increased by approximately 13.88%. This well-performing mixture model of both statistical and machine learning techniques underlines the statement that no rigid distinction between both areas should exist.

This report will continue with the description of the cryptocurrency and indexes' dataset in Section 2. Section 3 describes the techniques used in this research. The methodology is followed up with the experimental setup, including the evaluation method, the models used in the comparison, and the performance measures, in Section 4. Hereafter, in Section 5, the results of the experiment are described, and the best performing model is highlighted. Finally, in Sections 6 and 7, the conclusion of this research is drawn, and limitations and propositions for further research are noted.

2 Data

This section will first describe the dataset, followed up by a description of the key features of the dataset, and concludes with a brief overview of the transformations applied to the dataset.

2.1 Data description

The first dataset available for this research will consist of all hourly Open High Low Close Volume (OHLCV) data of the top 5 largest cryptocurrencies with Tether (USDT) as quote pair according to the market capitalisation provided by CoinMarketCap[1]. Tether is a stablecoin, which means that in theory, 1 USDT should represent the monetary value of 1 United States Dollar (USD), and thus it forms a tangible basis for the assets. The resulting cryptocurrency pairs with its market share relative to the total cryptocurrency market capacity at 04/05/2021 in brackets are:

1. BTC/USDT (46.662%)
2. ETH/USDT (17.1311%)
3. BNB/USDT (4.4743%)
4. XRP/USDT (2.9578%)
5. ADA/USDT (1.8644%)

These cryptocurrencies together dominate more than 73% of the market. The reason for the utilisation of multiple pairs is to allow the model to apply cross-sectional learning, which has been shown to allow for reducing forecast error (Armstrong, 2006).

As mentioned above, the OHLCV data consists of the Open, High, Low, Close, Volume data of a given interval in time. The sampling frequency of this dataset is hourly. The dataset starts on 14 August 2017, and this starting date only holds for Bitcoin and Ethereum. The other cryptocurrencies have been introduced later and thus differ in starting dates. The provided dataset ranges from 14/08/2017 until 01/05/2020, and the dataset from the beginning until 31/12/2020 will be used as the training and validation set. As a result, the dataset from 01/01/2021 until 01/05/2021 will be used as the hold-out or test set.

Other than the historical data for the cryptocurrencies, hourly OHLC data is also available through TradingView[2] for the S&P500 index (SPX), as well as for the Shanghai Composite Index (SHCOMP). The SPX could include more macro-economic effects, while a good reason to include SHCOMP would be to capture Chinese economic movements. These Chinese movements may be interesting to look at since more than 75% of the newly-mined Bitcoin is retrieved by Chinese miners (Ma et al., 2018; Kaiser et al., 2018). This starting date of the index dataset equals the starting date of the cryptocurrency dataset. Similarly to the cryptocurrency dataset, newer data may be fetched to be used in the test set. This additional dataset may be used to allow for cross-sectional learning

[1] `https://coinmarketcap.com`
[2] `https://tradingview.com`

and improve the model's forecasting ability. In contrast to the cryptocurrencies' dataset, this dataset does not contain the volume data per hourly interval. The resulting complete dataset with its summary statistics is shown in Appendix A.

2.2 Key features

The key features of the dataset are analysed and identified in this section. In particular, the trend, seasonality, non-linearity, and normality will be analysed. The emphasis in this section will be laid on the dependent variable, which is the opening price of the BTC/USDT pair since Table 5 in Appendix A shows that the correlation of the input features with the dependent variable is very high.

Figure 1: Open price of the BTC/USDT from 04-05-2018 until 31-12-2020

The opening price of the BTC/USDT pair or the dependent variable is shown in Figure 1. This figure indicates a trend within the data, but this assumption may be confirmed by regressing the price on a constant and $t = 1, \ldots, T$, where T is the length of the data. The estimate for the coefficient of t is 0.3673, with a standard error of 0.003, which means that it is significantly different from zero at the 1% significance level. Also, from the graph, it is already visible that there is no clear linear relationship within the data, especially near the end of the dataset, where it could even be classified as exponential growth. Also, the Augmented Dickey-Fuller test is executed to test for a unit root within the time series. The resulting test statistic is obtained with a value of 4.074 and with a p-value of 1 using 34 lags, which means that the null hypothesis of a unit root present in the time series is not rejected, and thus, the dataset is not stationary.

Since the dataset is not stationary, the first difference is taken, and its summary statistics are shown in Table 1. The variance is relatively high, and the distribution is

slightly skewed to the left. Also, the kurtosis of a normal distribution is 3, so a value higher than 37 indicates very fat tails and shows that the data does not follow a normal distribution. These statistics are also confirming the assumption that the data is highly volatile.

Table 1: Table giving an overview of the summary statistics of the first difference of the BTC/USDT pair's opening price from 04-05-2018 until 31-12-2020

Mean	Variance	Skewness	Kurtosis	Min	Max
0.8307	4682.0	-0.9139	37.2763	-1340.09	877.2

The following key feature that is analysed is seasonality. Existing literature has concluded that there does not reside a significant seasonal pattern in the data, except for negative returns on average in January (Kaiser, 2019). This result will be confirmed using three regressions to evaluate the January effect, a monthly seasonal pattern and a quarterly seasonal pattern.

The first regression regresses the difference in opening price on a dummy variable which equals one when the month equals January, and otherwise, it is zero. The estimate of the coefficient of this dummy variable is 1.3186 with a standard error of 1.774, and thus the null hypothesis of a January effect existing in this dataset is rejected. The second regression regresses the difference in opening price on twelve dummy variables, which equal one when the data point is in the respective month. The results of this regression yield only one significant variable with a p-value of 0.006. This variable is the dummy variable corresponding to the month December with the estimate of the coefficient equal to 4.0237 and a standard error of 1.450. The third regression is similar to the previous two, except that it uses four dummy variables for each quarter. For this regression, only the estimate of the coefficient of the dummy variable corresponding to the fourth quarter is significant with a positive coefficient of 2.1822 and a standard error of 0.842, resulting in a p-value of 0.010.

In contrast to the existing research where a significant seasonal pattern is found in January, from this regression result, the conclusion can be drawn that there is a positive significant seasonal pattern in the returns of the BTC/USDT pair during the month of December. Also, the significant effect of the fourth quarter is probably due to the strong seasonal pattern of the month December. However, considering the other months, there do not appear to be other significant seasonal patterns.

2.3 Data transformations

This section discusses the applied data transformation methods. The data features are transformed before applying the techniques mentioned in the methodology section, as a

crucial step in the processing pipeline is to preprocess the training data for the learning algorithm to digest. Some models already solve particular data-specific problems, such as differences in scale or seasonality. An example of such a model is the Exponential Smoothing (ES) model, discussed in Section 3.2.2. Therefore, the following data transformations are solely applied to the ensemble models that do not include the ES model, mainly ML models. Applying normalisation allows the models to obtain better results and reduce training time (Sola and Sevilla, 1997).

There are multiple ways to normalise the features, of which the primary methods are min-max and z-score normalisation. The first method is shown in Equation 1, which fits the values in an interval between 0 and 1. The caveat method is that if only a subset of the data is given, the global minimum or maximum is unknown. Also, this method does not handle outliers well.

$$x_i^* = \frac{x_i - max_i}{max_i - min_i} \tag{1}$$

The second method is that of z-score normalisation. This method subtracts the mean from the value x_i and divides it by the standard deviation. As the mean and standard deviation may change over time, it could introduce problems to use this method in the standard approach. Therefore, the choice has been made to use a sliding window of a to be determined length θ in combination with the conventional z-score normalisation formula to arrive at an adaptive z-score normalisation shown in Equation 2. This method aims to provide a solution to non-stationarity within the data using this sliding window approach. In this equation, $\mu_\theta(i)$ is the moving average with length θ, and $\sigma_\theta(i)$ is the standard deviation of $x_{i-\theta}, \ldots, x_i$. The conventional z-score normalisation is a special case of this formula with $\theta = i$.

$$x_i^* = \frac{x_i - \mu_\theta(i)}{\sigma_\theta(i)}, 1 \leq i \leq n \tag{2}$$

The last data transformation method solves the problem of missing data. The indexes' dataset contains gaps because the conventional exchange is not open 24 hours a day and seven days a week, resulting in missing data points when the exchange is closed. Therefore, the missing data in this dataset is fixed by backfilling using the latest point available. Using backfilling in an actual market scenario would define the latest value as the latest data available. If, in this case, linear interpolation were to be applied, it would introduce a look-ahead bias since now future information about market price movements would be available.

In summary, the dataset features are backfilled to fix the problem of missing data, after which adaptive normalisation is applied. Consequently, at time t, all transformed and normalised features, except for the closing price of the cryptocurrency with the highest market capitalisation, Bitcoin, are taken as the input features for the learning algorithms,

resulting in a total number of 20 input features. The normalised closing price of the BTC/USDT pair will be taken as the dependent variable, which also forms the price at time $t+1$. Finally, when the ensemble model has predicted the price at $t+1$, this can be transformed back to the actual price at time $t+1$ using the $\mu_\theta(t+1)$ and $\sigma_\theta(t+1)$ from Equation 2.

3 Methodology

The methodology is split up into four segments to evaluate the research question. The first two sections introduce the individual ML and statistical methods examined in this research. Afterwards, the third section will discuss the ensemble techniques which combine the individual ML or statistical methods. The last segment introduces the optimisation method for the models' hyperparameters and the performance metrics used in the evaluation.

3.1 Machine learning methods

The machine learning methods will be used as predictors and combined using the ensemble technique mentioned in the previous section. This combination is adopted as a basis to improve upon using the statistical techniques mentioned in Section 3.2. Since the dependent variable displays signs of high volatility and a non-linear relationship, flexible models may be preferred. Therefore, a combination of a model utilising deep learning and a tree-based ensemble model is chosen. The deep learning model consists of a Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) units, introduced by Hochreiter and Schmidhuber (1997), and the novel LightGBM represents the tree-based ensemble model, introduced by Ke et al. (2017). Additionally, these models also formed the top models in the M4 and M5 forecasting competitions (Makridakis et al., 2018a, 2020). Hereafter, these models will be individually described in more detail.

3.1.1 LSTM-layered Recurrent Neural Network

The first ML predictor is the LSTM-layered Recurrent Neural Network and differs from a conventional neural network in the sense that it allows for recognising temporal patterns, such as speech analysis, where preceding words partially define following words (Graves et al., 2013). Since it allows for temporal dependencies within the data, it is a good fit for sequential time series data. An alternative to RNNs is Convolutional Neural Networks (CNN), which is mainly used for image classification (Krizhevsky et al., 2012). In general, the CNNs may be used interchangeably with the RNNs for time series data. However, in contrast to the CNNs, the RNNs allow for some memory measures.

The memory measure added to the RNN may be found in the network's hidden layer and comprises LSTM units. In conventional RNNs, the gradient function decays exponentially over time and minimises the impact of non-recent data. However, these RNNs may suffer from problems in the learning process, such as exploding or vanishing gradients (Bengio et al., 2013). An example of such a situation is illustrated in the context of speech recognition when the goal of the model is to predict the last word of the sentence with the entire paragraph until the last word as input. The paragraph could be anything like,

"I am born in The Netherlands, . . . , I speak fluent *Dutch*", where the crucial information was given in the first sentence. A conventional RNN could find difficulties predicting the word Dutch due to the RNN's problem of handling long-term dependencies. Pascanu et al. (2013) provides a more in-depth explanation of this problem and state that the LSTM units provide a partial solution to this problem.

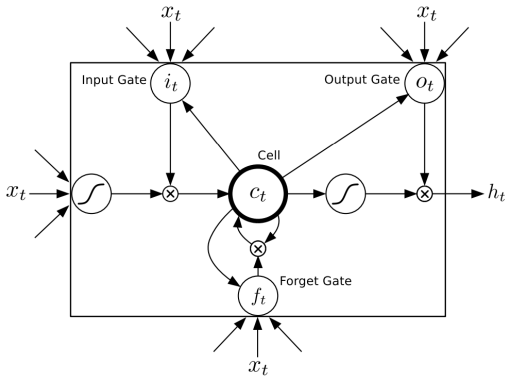


Figure 2: Figure of LSTM cell, from Graves et al. (2013)

$$
\begin{aligned}
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\
h_t &= o_t \tanh(c_t).
\end{aligned} \tag{3}
$$

An LSTM layer consists of stacked LSTM units, of which a single LSTM unit is depicted in Figure 2, and its equations are shown in Equation 3. This LSTM cell consists of input i_t, forget f_t, and output gates o_t, where the forget gate handles diminishing the impact of non-recent data. The conventional RNN implements a simplified implementation of such an LSTM cell, where it only contains an input gate with input x_t, output gate with output x_t and the hidden state with tanh activation function h_t. The x_t from the output gate and the hidden state h_t are then passed to the next cell in the layer. The hidden layer weight matrices are defined as W and the bias vectors as b. The extra forget gate f_t utilises the Sigmoid activation function σ to output a value between zero and one to determine how much of the cell's state c_t should be forgotten mathematically shown in the first equation. The two following equations determine how much of the new information fed through the input gate i_t should be retained in the cell's new state c_t, and which part of the old cell's state c_{t-1} should be kept. The last two lines of the equation handle the conversion to the required output by utilising the tanh activation function h_t

to squash the values between -1 and 1. It also multiplies this output o_t with the output of the Sigmoid gate, such that only the required output is returned. As the forget gate's activation function never returns a value greater than zero, the gradients do not vanish, nor do they explode.

In this research, the learning rate of the RNN α will be deemed as a hyperparameter, as well as the number of hidden LSTM cells used in the model. The number of hidden layers will be constrained in the optimisation between one and two due to issues in the runtime complexity of the algorithms. The RNN model will use the Symmetric Mean Absolute Percentage Error (SMAPE) loss function, shown in Equation 30, since it is not possible to use the COMB method, shown in Equation 32, which is also used in the final evaluation. The reason for not being able to use the COMB method is that the COMB method uses the arithmetic mean of multiple models, which is not known for a single model. Also, SMAPE yields higher discrepancies within the values, allowing it to be a better choice for hyperparameter fitting than the other performance metric used in the COMB measure.

3.1.2 LightGBM

The novel LightGBM method, introduced by Ke et al. (2017), is the second ML predictor and provides a highly efficient and scalable implementation of the gradient boosting decision tree algorithm, initially introduced by Friedman (2001). The model itself is already an ensemble model of decision trees trained sequentially by fitting the negative gradient of the previous iteration's loss function, explained in a more detailed manner in Section 3.3.2. Other than the superior performance it has shown in the M5 competition, where it formed the basis for the top-performing models during the forecasting competition, it has also shown robust performance has been shown in forecasting the price trend of cryptocurrencies (Sun et al., 2020).

The LightGBM method is used with a regression objective, implying that it uses regression decision trees. Internally it utilises the sum of squared residuals, also known as the L2 loss function, which penalises the errors made by the models. The regression objective averages the result of all weak learners to obtain the final weighted value compared to using the conventional classification objective and thus using majority voting to decide upon the final weighted value.

Subsequently, this novel method utilises two novel techniques to handle high feature dimensionality and large data size without significantly sacrificing the model's accuracy, which will be discussed in the following paragraphs. The first technique is the Gradient-based One-Side Sampling (GOSS) method and aims to achieve a good balance between reducing the number of data instances and maintaining the accuracy for learned decision trees. This algorithm is shown in Appendix B.1. It keeps the data points with large

gradients and performs random sampling on the data points with small gradients. Additionally, it multiplies the sampled data with small gradients with a factor $\frac{1-a}{b}$ when the information gain is calculated to emphasise under-trained data points. a is a constant for selecting the top a * 100% instances, where b corresponds to the constant for randomly sampling b * 100% data points with small gradients.

The second technique is Exclusive Feature Bundling (EFB) and is shown in Appendix B.2. It aims to reduce the number of features effectively. The idea is that high-dimensional data is usually very sparse, which allows bundling exclusive features into a single feature, or also called an exclusive feature bundle. Exclusive features imply that two features never take non-zero values simultaneously. This EFB algorithm reduces the number of features by bundling exclusive features and thus avoids unnecessary computation for zero feature values.

The last technique to increase optimisation and reduce training time is not stated in the research of (Sun et al., 2020) directly but implemented implicitly. This technique concerns the speed of convergence in the optimisation, and in contrast to solely using the gradient, it also uses the curvature information or Hessian. This method is also called the Newton method, and changes Equation 25 to Equation 4, allowing to reach the point of convergence faster.

$$y_i^* = -\left[\left(\frac{\partial^2 L(y_i, F(x_i))}{\partial F^2(x_i)}\right)^{-1} \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)-F_{m-1}(x)}, \quad \text{for } i = 1, \ldots, N \tag{4}$$

3.2 Statistical techniques

While novel machine learning methods may capture a dataset's critical features from a data science perspective, well-established statistical techniques may improve an ML model's performance by capturing other features. Januschowski et al. (2020) argue that a hard methodological distinction between these two areas does not exist. Techniques improving forecasting performance by modifying the dataset and adding an additional predictor are proposed and investigated. The dataset-modifying techniques encompass removing noise from the dataset using a Fourier transformation-based technique and adding data regarding the probability of being in a bull or a bear market. The additional predictor could allow for an increase in robustness in the forecasting ability of the ensemble model. These proposed statistical techniques are discussed in the following sections.

3.2.1 Discrete Wavelet Transform

Although machine learning models should provide an increase in flexibility, a high number of aberrant observations and thus a significant level of corresponding noise could negatively

affect the models' performance. A solution is proposed by Shensa et al. (1992) in the form of applying Discrete Wavelet Transform (DWT) to denoise the data and thus aid the feature extraction of the dataset by the models. Al Wadia and Ismail (2011) have shown the significance of utilising this technique in the statistical context, while Nobre and Neves (2019) have also illustrated the addition of this technique in the context of an ensemble machine learning model. In this research's context, de-noising the original dataset before feeding it to the models will be examined for significance in the models' performance. Hereafter, a comprehensive description of this technique will be given.

The DWT is a variation of the Fourier transformation at multiple scales and converts the signal to a set of multiple mutually orthogonal wavelets. The relation between the discrete wavelet and the mother wavelet is shown in Equation 5. In this equation, ψ is the mother wavelet, $m \in \mathbb{N}$ the parameter for controlling the wavelet width or also called dilation, and $n \in \mathbb{N}$ the parameter for controlling the translation of the wavelet over the x-axis. s_0 is a specified fixed dilation step parameter, for which $s_0 > 1$, and b_0 is the initial translation parameter, for which each $b_j > 0$ and $j \in N$ with N as the number of wavelets. These parameters, except for the ψ are estimated at every time interval of the data or signal to estimate the continuous wavelet signal, shown in Equation 6. In this equation, the wavelet transform $T_{m,k}$ transforms the continuous signal $x(t)$ with the discrete wavelets, which in this research will be used to denoise the data. Equation 7 shows that this can also be expressed as the inner product of the DWT, shown in Equation 5, and the continuous signal, shown in Equation 6. An example of this process is shown in Figure 3, where the wavelet with fixed dilation or width is fitted on the signal. The fitting of the wavelet at a single location, and the effect of the surrounding points on the integral of the continuous wavelet, is shown in Figure 3(a), where the regions with a '+' imply a positive effect on the integral, and the regions with a '-' denote a negative effect on the integral. Using multiple points, which in Figure 3(b) is shown as $b_1, \ldots, b_4$, the final continuous wavelet $T(a, b)$ is approximated.

$$\psi_{m,k}(t) = \frac{1}{\sqrt{s_0^m}} \psi\left(\frac{t - k\tau_0 s_0^m}{s_0^m}\right) \tag{5}$$

$$T_{m,k} = \int_{-\infty}^{\infty} x(t) \frac{1}{s_0^{m/2}} \psi(s_0^{-m} t - k\tau_0) dt \tag{6}$$

$$= \langle x, \psi_{m,k} \rangle \tag{7}$$

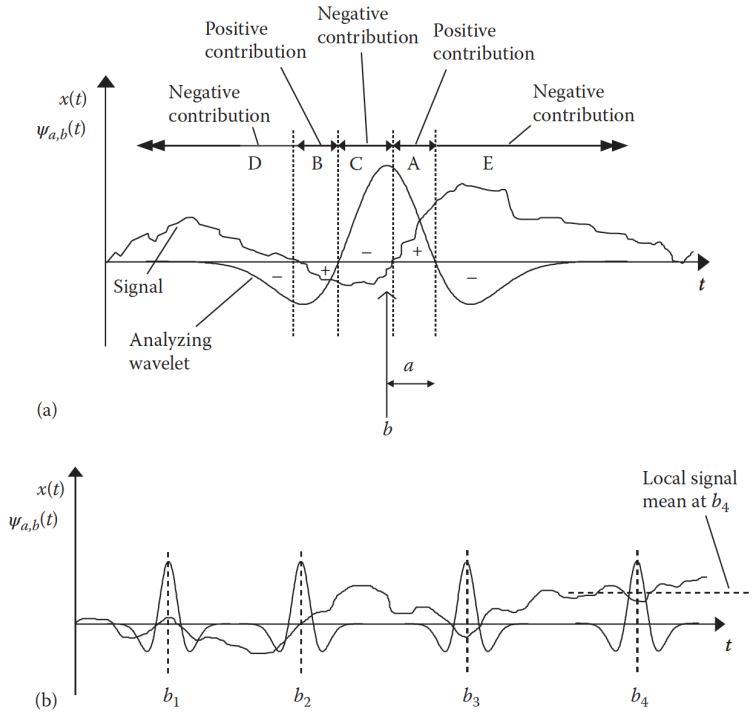


Figure 3: In figure (a), the wavelet of the specific dilation and the translation on the signal is shown. The regions containing the + or - denote the effect they have on the continuous wavelet integral. In figure (b), the wavelet with fixed dilation is fitted on the signal. At location $b_1, T(a,b)$ has a large positive value, at $b_2, T(a,b)$ is approximately zero, $b_3, T(a,b)$ carries a large negative value, and at $b_4, T(a,b)$ the local minimum corresponds with the high part of the wavelet and the surrounding higher parts correspond to the negative parts of the wavelet. Combined, this gives a large negative value for $T(a,b)$. This figure is retrieved from Addison (2017)

The complete in-depth derivation of the parameter estimation in each interval and the derivation of the wavelet transform procedure is shown by Addison (2017). A significant reduction in the number of parameters that have to be estimated is achieved using this estimation method. The mother wavelet ψ has to be chosen from a total of 77 wavelets originating from seven wavelet families[3], and will be chosen through the optimisation framework.

Subsequently, this wavelet can be extended to multiple high and low-frequency layers, where the low-frequency layers capture the general movement of the data, and the high-frequency layers capture the details. Consequently, a Low-Pass Filter (LPF) can be applied, which applies a threshold to the high-frequency layers to remove the details or noise. This filter sets the value to zero if it does not exceed the threshold. These filters can be extended to encompass multiple layers, as shown for a two-layer case in Figure 4. The original data $x[n]$ is split evenly into two parts, where the high-pass filter contains the first level details and its coefficients, and the low-pass filter contains the residuals. Afterwards,

[3] `http://wavelets.pybytes.com/`

the first-level low-pass filter residuals are split in two again to obtain the level two detail coefficients in the high-pass filters and the residuals together with their approximation coefficients. In the general case, the number of times that the data is split equals the level of depths in the DWT decomposition. This method allows the signal to be reconstructed using the inverse function with the obtained coefficients. This reconstructability allows the removal of specific filtered data from the signals, after which the original denoised data can be reconstructed with the inverse function and its parameters.

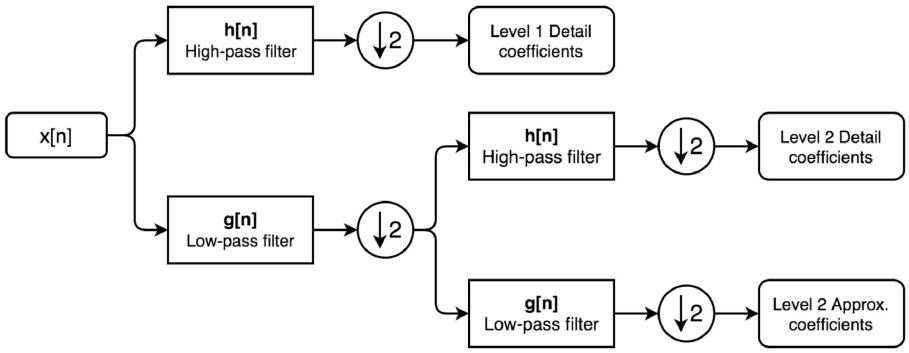


Figure 4: DWT decomposition of a signal for depth level 2, from Nobre and Neves (2019)

3.2.2 Exponential Smoothing

The following statistical technique is a well-known and widely-used technique in the context of time series analysis. This technique is the method of Exponential Smoothing (ES) (Gardner Jr, 2006) and carries the capability to apply deseasonalisation as well as adaptive normalisation. The choice has been made to implement a more extensive ES model to allow for seasonality and a trend, which is the Holt-Winters' multiplicative ES model (Hyndman et al., 2008). Similar to the RNN-LSTM method, the ES model attaches an exponentially decreasing weight to observations in the past. An important note is that this model does not require normalised data, and therefore the data is not normalised before feeding it to this ES predictor to digest. In this research, the ES model is added as an extra predictor which is then combined with the other ML methods using bagging to form a new larger ensemble model. Hereafter, the different types of possible ES models are described, and these different types will be seen as configurations of hyperparameters to optimise.

The non-seasonal ES model provides a smoothed value using Equation 8, where α is the smoothing coefficient, and l_t is the smoothed value or level at time t of the series.

$$l_t = \alpha y_t + (1 - \alpha) l_{t-1} \tag{8}$$

The equations can be updated to account for seasonality and trend components, which results in the Holt-Winters ES model's equations, shown in Equation 9, where β and γ

represents the smoothing coefficient for the trend and seasonality component, respectively. b_t and s_{t+K} represent the trend and seasonality component themselves, and K is the number of observations per seasonal period, which for a monthly series is, for example, 12. As can be seen in these equations, the trend component introduces a linear relationship in the local trend, an assumption made to simplify the equations and calculations.

$$\begin{aligned} l_t &= \alpha y_t / s_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \\ s_{t+K} &= \gamma y_t / (l_t + b_t) + (1-\gamma) s_t \\ \hat{y}_{t+1} &= (l_t + K b_t) s_t \end{aligned} \tag{9}$$

Additionally, the optimisation may conclude that it may provide a better option to include either seasonality or trend components. In this case, a simplified version of the Holt-Winters' model is used, and is shown in Equations 10 and 11 for the exclusive seasonality and trend components, respectively.

$$\begin{aligned} l_t &= \alpha y_t / s_t + (1-\alpha) l_{t-1} \\ s_{t+K} &= \gamma y_t / l_t + (1-\gamma) s_t \\ \hat{y}_{t+1} &= l_t * s_t \end{aligned} \tag{10}$$

$$\begin{aligned} l_t &= \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \\ \hat{y}_{t+1} &= l_t + K b_t \end{aligned} \tag{11}$$

3.2.3 Markov Switching

The cryptocurrency market has been known for extensive periods corresponding with high volatility upwards, increasing the price, and the opposite. These two periods can also be seen as two states, where it is possible to switch from one state to the next state in the following period with a specific probability. Another example of such two states is the phenomenon of Bull and Bear markets, also called expansion and recession periods. In such an expansion period, the asset price rises continuously for a more extended period, and in more statistical terms, this phenomenon is defined as having a continuous positive trend for a substantially more extended period. The recession period is the opposite, where the trend is continuously negative for a considerable time. The ability to identify such regimes is supplied using the Markov Switching (MS) technique, and this technique could supply the predictors with the probabilities of situating in a specific regime and the transition probability to the other regime. Kole and Van Dijk (2017) have shown in their research that the usage of Markov Switching models leads to superior forecasts and out-of-sample performance in equity markets. In this research, the MS technique will be added to the ensemble model by extending the input dataset with the two sets of

probabilities as features.

The input dataset consists of price data at an hourly interval, which the MS model aims to extend by adding the probability of being in state 0 as a feature. Two states will be considered in the MS model to follow the idea of the bull and bear market state or regime. State 0 would then correspond to either the bull market or the bear market, as the model's specification does not distinguish the order of the states. As new data arrives at an hourly interval, the MS model would update its probabilities each hour to improve its estimates in the ideal situation. However, due to deficiencies in processing power and constraints in calculation time, it is impossible to update the probabilities each hour, and thus the update frequency of the probabilities has been decreased to one update every three hours. Henceforth, the model and the process of retrieving the smoothed probabilities will be described in more detail.

In the Markov Switching (MS) model, the assumption is made that the returns process X_t follows different distributions in different states. In case of a bull and bear market, the process S_t could be identified as $S_t = 0$ when it is a bear market and $S_t = 1$, when it is a bull market. Combining this process with the assumption that the returns Y_t follow a normal distribution in both states, this results in the following equation:

$$Y_t \sim \begin{cases} \mathcal{N}(\mu_1, \sigma_1^2) & \text{when } S_t = 0 \\ \mathcal{N}(\mu_2, \sigma_2^2) & \text{when } S_t = 1 \end{cases}$$

The latent process S_t follows a first order ergodic Markov chain, which means that at time t, the process S_t only depends at the process value of previous time S_{t-1}. This can be translated into a transition probability matrix $\mathbf{P}$, which is given in Equation 12, where p_{ij} indicates the probability to go to state $S_t = i$ from state $S_{t-1} = j$.

$$\boldsymbol{P} = \begin{pmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{pmatrix} = \begin{pmatrix} p_{00} & 1 - p_{11} \\ 1 - p_{00} & p_{11} \end{pmatrix} \tag{12}$$

Since process S_t is a latent process, the actual and the initial value S_1 are unobserved. Therefore an additional variable is introduced, which equals the probability that $S_1 = 0$, which is shown more formally in Equation 13. Since the initial probabilities sum up to 1, the probability that $S_1 = 1$ equals $1 - \zeta$. This newly-defined parameter will be estimated as well.

$$\zeta = P(S_1 = 0) \tag{13}$$

As previously mentioned, the probability values of the latent process S_t have to be estimated. For the process at time t, all information is used up until time t, which should be contained in the returns $y_1, \ldots, y_t$, which can be extended to forecast the latent process

at time $t+1$ using all information up until time t, which is formally defined in Equation 14 and 15.

$$\boldsymbol{\xi}_{t|t} = \begin{pmatrix} P(S_t = 0|y_1, \ldots, y_t) \\ P(S_t = 1|y_1, \ldots, y_t) \end{pmatrix} \tag{14}$$

$$\boldsymbol{\xi}_{t+1|t} = \begin{pmatrix} P(S_{t+1} = 0|y_1, \ldots, y_t) \\ P(S_{t+1} = 1|y_1, \ldots, y_t) \end{pmatrix} \tag{15}$$

Furthermore, both states' (normal) densities are inserted in a vector as shown in Equation 16.

$$\boldsymbol{f}_t = \begin{pmatrix} f(y_t, \mu_1, \sigma_1^2) \\ f(y_t, \mu_1, \sigma_1^2) \end{pmatrix} \tag{16}$$

Combining these definitions and applying recursion to estimate the probabilities is also called the Hamilton filter, and applies the Equations 17 and 18 recursively, with the starting point $\xi_{1|0} = (\zeta, 1-\zeta)'$, where $\odot$ denotes the element-wise multiplication (Hamilton, 1989).

$$\boldsymbol{\xi}_{t|t} = \frac{1}{\boldsymbol{\xi}'_{t|t-1}\boldsymbol{f}_t}\boldsymbol{\xi}_{t|t-1} \odot \boldsymbol{f}_t \tag{17}$$

$$\boldsymbol{\xi}_{t+1|t} = \boldsymbol{P}\boldsymbol{\xi}_{t|t} \tag{18}$$

Consequently, the Kim smoother is used to reduce the noise in the estimates (Kim, 1994). This smoother uses all information known at time T to estimate the probabilities of the regimes S_t at times $t = 1, \ldots, T$. Since these estimates contain less noise in their estimates, these smoothed probabilities are added as a feature to the input dataset at every time t. The formal definition is shown in Equation 19, where $\div$ denotes the element-wise division operator.

$$\boldsymbol{\xi}_{t|T} = \begin{pmatrix} P(S_t = 0|y_1, \ldots, y_t) \\ P(S_t = 1|y_1, \ldots, y_t) \end{pmatrix} = \boldsymbol{\xi}_{t|t} \odot (\boldsymbol{P}'(\boldsymbol{\xi}_{t+1|T} \div \boldsymbol{\xi}_{t+1|t})) \tag{19}$$

The estimation is done using the Expectation-Maximisation (EM) algorithm, shown in this context by Hamilton (1990). In this algorithm, S_t follows a Bernoulli distribution, after which the log-likelihood, which is shown in Equation 20, is maximised concerning each parameter to get the new iteration of the parameter estimate.

$$
\begin{aligned}
\log\big(\mathcal{L}(Y_T, S_T, \theta)\big) = &\sum_{t=1}^{T} \Big((1-s_t)\log(f(y_t, \mu_1, \sigma_1^2)) + s_t \log(f(y_t, \mu_2, \sigma_2^2)) \Big) \\
&+ \sum_{t=2}^{T} \Big((1-s_t)(1-s_{t-1})\log(p_{00}) + (1-s_t)s_{t-1}\log(1-p_{11}) \\
&+ s_t(1-s_{t-1})\log(1-p_{00}) + s_t s_{t-1}\log(p_{11}) \Big) \\
&+ (1-s_1)\log(\zeta) + s_1 \log(1-\zeta)
\end{aligned} \tag{20}
$$

3.3 Ensemble techniques

Techniques to combine multiple individual models or methods are called ensemble techniques, which focus on enhancing a model's predictive performance by using a combination of multiple model-fitting methods instead of using a single fit of the method (Bühlmann, 2012). Forecast errors originate mostly due to noise, model bias, and model variance. As mentioned earlier, the problem of model bias and variance can be reduced by using ensemble techniques to combine multiple different models, where the only requirement is that the difference between models is significant (Kuncheva and Whitaker, 2003). Combining multiple different models is done using a model-fitting method, which may be defined as estimating the real-valued function, shown in Equation 21. In this equation, k is the length of the input vector for the predictors, given input $X_i, \dots X_n$ and output $Y_i, \dots, Y_n$. In this report, the two most widely-used ensemble techniques will be evaluated using the predictors: bagging and boosting. The description and the implementation of these techniques will be described in the following sections.

$$
g : \mathbb{R}^k \mapsto \mathbb{R} \tag{21}
$$

3.3.1 Bagging

The first technique is called Bootstrap AGGregatING or bagging and combines multiple individual models' forecasts using a linear combination of multiple individual forecasts to combine them into one forecast. This technique is fairly easy to implement, as all models can individually be developed and used when forecasting. Another essential feature of this method is its potential in handling imbalanced classes (Galar et al., 2011), which in this paper's context could translate to attaching a higher weight to the individual models which focus on predicting in different states as classified by the MS model. Its version of the ensemble method is given in Equation 22, where $f_j(\cdot), j = 1, \dots, d$ is the individual predictor, and $w(\cdot)$ is the weighting function for all individual predictors. A simple version or implementation, used in the top model of the M5 competition, is

evenly weighting the individual predictors, such that the equation simplifies to Equation 23. In this research, the multiple predictor models will be used to forecast data points individually and combined using this simplified equation.

$$g_{\text{bagging}}(\cdot) = w(f_1(\cdot), \ldots, f_d(\cdot)) \tag{22}$$

$$g^*_{\text{bagging}}(\cdot) = \frac{1}{d}\sum_{i}^{d} f_i(\cdot) \tag{23}$$

3.3.2 Boosting

The second technique is boosting, which in contrast to bagging, uses a basis of multiple weak learners as a basis to be combined into a strong learner. It uses an iterative process to sequentially pass the first predictor's residuals in a loss function and then passed on to the following learner. In a simple illustration, it could be expressed in econometric terms as fitting subsequent models on the residuals of the prior ones. In a linear regression case, the loss function would be defined as the sum of squared residuals, which then can be passed to the subsequent regression model to process or minimise.

In this research, boosting is used as the basis of a more sophisticated technique in one of the predictor models, namely, the gradient boosting decision tree method (Friedman, 2001). This gradient boosting decision tree method is implicitly implemented in the LightGBM method, which is mentioned earlier in Section 3.1.2, and utilises the L2 loss function, also called the sum of squared residuals. Subsequently, this method follows a similar iterative sequential approach and is thus fitted on a loss function utilising the predicted values and the actual values as input. The gradient or derivative with respect to the first learner can be derived using this loss function. Like a conventional gradient descent algorithm, the next learner is optimised in the negative gradient direction. More formally, the procedure is defined as follows:

The goal is to find a $\hat{F}(x)$, which provides the best approximation for the output variable y given the input variable x. Given the loss function $L(y, F(x))$, the expression may be formalised as shown in Equation 24.

$$\hat{F}(x) = \arg\min_{F} \mathbb{E}[L(y, F(x))] \tag{24}$$

The gradient boosting method estimates $\hat{F}(x)$ with a constant and the sum of weighted functions $h_m(x)$ from a set with a length M, consisting of weak learners $\mathcal{H}$. The first learner is then defined as the constant, which is estimated as:

$$F_0(x) = \arg\min_{F} \sum_{i=1}^{N} L(y_i, F(x_i))$$

Then for each of the number of iterations $m \in \{1, \ldots, M\}$, the negative gradient of the residuals y_i^* is calculated, as shown in Equation 25.

$$y_i^* = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)-F_{m-1}(x)}, \quad \text{for } i = 1, \ldots, N \tag{25}$$

Consequently, the weak learner is fitted on the new training set (x_i, y_i^*) instead of (x_i, y_i), for $i = 1, \ldots, N$. After this new weak learner has been fitted on the new training set, the weight β_m is determined for this new weak learner by solving the equation shown in Equation 26.

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h_m(x_i)) \tag{26}$$

Finally, the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \beta_m h_m(x) \tag{27}$$

3.4 Optimisation

As each model carries its own set of hyperparameters, these hyperparameters' values have to be set. Trying every possible combination of hyperparameters would be very tedious. Therefore, when choosing the optimal set of hyperparameters, a loss function is specified with the goal to minimise it. An example of such an objective function is given in Equation 28, where the parameter vector w that minimises $F(w)$ has to be determined or optimised, and $F_i(w)$ corresponds to the objective function evaluated at observation i. In linear regression, this loss function would be the sum of squared residuals, and its goal would be to minimise the sum of squared residuals to find the optimal $\boldsymbol{\beta}$, which is the optimal set of hyperparameters.

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} F_i(w) \tag{28}$$

In the context of this research paper, it would make sense to choose a loss function similar to the performance measures used to evaluate the models. Therefore, the loss function will be chosen to be equal to the performance measure stated in Equation 32.

As mentioned above, finding the optimal set of hyperparameters is very tedious, and therefore an optimiser will be used to speed up the process. The hyperparameter space is vast due to the usage of multiple combined individual models in the form of an ensemble model. Also, the individually-applied statistical techniques will require the optimisation of a large number of hyperparameters. Therefore, the choice has been made to use a different approach from the conventional gradient-utilising approaches, such as Stochastic Gradient

Descent or Adam, introduced by Kingma and Ba (2014). Bergstra et al. (2011) have introduced the Tree-structured Parzen Estimator (TPE) optimiser, which allows a faster optimisation in high-dimensional parameter space and hence uses fewer resources. It uses a Bayesian approach, and models $p(x|y)$ and $p(y)$, where x is the vector of hyperparameters and y represents the hyperparameters' fitness score. $p(x|y)$ is approximated by replacing the probability density of the hyperparameters with non-parametric densities, which is shown in Equation 29.

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{else} \end{cases} \tag{29}$$

In this equation, $l(x)$ is the density formed using the observations x, such that the corresponding loss function $F(x)$ was less than y^*, and $g(x)$ is formed using the remaining observations. y^* is chosen to be some quantile γ of the y values. Since the algorithm maintains a sorted list of observed variables in a set that we define as $\mathcal{H}$, the runtime of each iteration of the algorithm scales linearly in the length of this set, $|\mathcal{H}|$, and also linearly in the number of dimensions being optimised. A more detailed explanation of this algorithm can be found in Bergstra et al. (2013).

4 Experimental setup

Following the previous sections, which gave descriptions of the dataset and the methods applied in this research, this section compares the enhanced optimised models. Adding a new technique to the ensemble model will be evaluated against all other possible combinations for its significance. The models will be used to make point forecasts for the BTC/USDT closing price at time $t+1$, given the data at time t, which are then evaluated using multiple performance measures. A schematic overview of the entire testing process may be found in Figure 13 of Appendix C. The following sections will describe the models used for comparison, the testing procedure, and the used performance measures.

4.1 Models used for comparison

The new technologies have been identified and have to be compared against a base model. The base ML ensemble model will act as the benchmark model to compare against, as this research focuses on improving an ensemble of machine learning models. It consists of an ensemble of the LightGBM model and the RNN-LSTM model, with the ensemble technique mentioned in Section 3.3 and is also shown on the first line of Table 2. Additionally, every technique will be tested and compared against every other combination of any statistical technique, as shown in the table. Exactly how the ensemble model is extended with each technique is detailed in Section 3, which also contains descriptions for each individual technique. The evaluation will be done according to the performance measures, which are described in Section 4.3.

Table 2: Table giving an overview of the models used for comparison and evaluation

Ensemble model	Additional statistical techniques
LightGBM & RNN-LSTM	
LightGBM & RNN-LSTM	DWT
LightGBM & RNN-LSTM	MS
LightGBM & RNN-LSTM	DWT & MS
LightGBM & RNN-LSTM & ES	
LightGBM & RNN-LSTM & ES	DWT
LightGBM & RNN-LSTM & ES	MS
LightGBM & RNN-LSTM & ES	DWT & MS

4.2 Time series k-fold cross-validation

Conventional cross-validation allows a model's performance to be assessed on a part that the model has not seen before, such that it would indicate the model's performance with limited bias. However, using the last part of the dataset as the test set to validate

the model's performance could lead to reliability or bias problems. Therefore, k-fold cross-validation is introduced, which splits the dataset into k folds, for which each fold a specific training (80%) and hold-out (20%) set is defined. Using multiple folds and multiple different kinds of data as the hold-out set should yield a less biased evaluation. The exact k will be determined using the elbow plot, which aims to select the k such that the majority of the variation is retained. This method is also used in Principal Components Analysis to select the k majority principal components.

However, conventional k-fold cross-validation cannot be used due to the dataset's temporal relations considering the given data. The conventional method does not differentiate in time, and thus the model may have seen and trained on future data when evaluating its performance. Therefore, a slightly modified version is used: the Time-Series k-fold cross-validation to assess the model's accuracy. This modified version splits the dataset into k folds, which given an input matrix X results in $\{X_i\}, i = 1, \ldots, k$. Then within a fold X_i, the model is trained on samples $X_{i,1}, \ldots, X_{i,t-1}$, after which it predicts the next point $\hat{y}_{i,t}$, and trains the model on observations $X_{i,1}, \ldots, X_{i,t}$ to repeat the loop until observation $X_{i,M}$, where M is the length of the fold, and t is the first point in fold X_i to start evaluating from. With this procedure, the temporal relationships are kept in place.

An overview of this modified version within a single fold is shown in Figure 5. During the optimisation process, described in Section 3.4, the optimiser aims to find the optimal set of hyperparameters based on the lowest loss score value using this procedure. However, as this procedure may be repeated multiple times using different configurations, it could lead to the concept of data mining and a look-ahead bias, where the model is overfitted to minimise the error in every hold-out data set iteratively, and would hence provide knowledge to the subsequent iterations of the model updates. This issue occurs due to the fact that the data used in the final validation of the models has already been used in the hyperparameter-fitting process, and therefore the models could already be optimised on data that it should not have seen before.

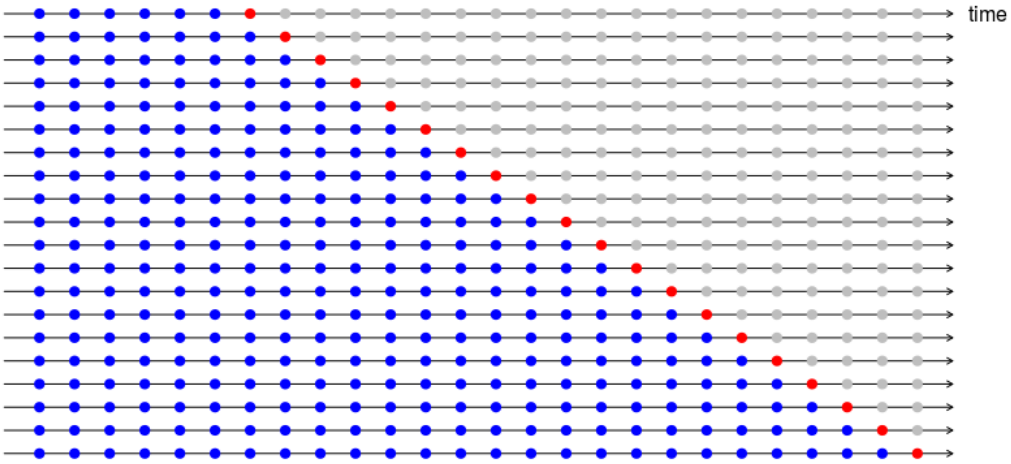


Figure 5: An illustration of time series 1-fold cross-validation.

Therefore, as mentioned in Section 2, where the used datasets are described, a newer chunk of the data source is used. The dates of the training and validation datasets range from 14/07/2017 to 31/12/2020. The resulting total number of training observations is 23,270 and is split over k folds. These training observations are used in the model's optimisation. Therefore, the newer chunk may be chosen as the data with dates 01/01/2021 to 01/05/2021, resulting in a validation set of 2,871 extra observations and thus a complete dataset of 26,141 observations. The newer chunk of the dataset is split such that the model's performance is undoubtedly evaluated with a dataset it has not seen before, which increases the level of unbiasedness and reliability of the results.

4.3 Performance measures

Many different performance measures exist to evaluate the model's performance, and the performance of the 'best' forecast depends on the error of the accuracy measure (Kolassa, 2020). Therefore, two individual performance measures and one combined performance measure are used, where the combined measure will be used as a general guideline. The first performance metric is the Symmetric Mean Absolute Percent Error (sMAPE). sMAPE is an accuracy measure based on percentage errors, and in contrast to the conventional Mean Absolute Percent Error (MAPE), sMAPe is symmetric and offers upper and lower bounds (Makridakis, 1993). The formula for sMAPE is given in Equation 30, where Y_t is the actual value, $\hat{Y}_t$ is the forecasted value, and n the length of the total sample.

$$SMAPE = \frac{2}{n} \sum_{t=1}^{n} \frac{|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} \tag{30}$$

The second metric is the Mean Absolute Scaled Error (MASE) and is the Mean Absolute Error (MAE) of the forecasted values divided by the MAE of the training data

one-step naive forecast, which is shown in Equation 31 (Hyndman and Koehler, 2006).

$$MASE = \frac{1}{n} \frac{\sum_{t=1}^{n} |Y_t - \hat{Y}_t|}{\frac{1}{n-1} \sum_{t=2}^{n} |Y_t - Y_{t-m}|} \tag{31}$$

Finally, the combined metric utilised both metrics mentioned previously and is called COMB. As the two metrics differ in the denominator, they cannot be evenly weighted, and thus, the arithmetic mean is taken instead. Given the performance metrics of the N models, we have the individual performance metric values $\{MASE_i\}$ and $\{SMAPE_i\}$, $i = 1, \ldots, N$. The COMB measure is defined as shown in Equation 32 and utilises these individual performance metric values. As aforementioned, this metric will form the primary guideline to evaluate a model's performance.

$$COMB_i = \frac{1}{2} \left(\frac{MASE_i}{\sum_{k=1}^{N} MASE_k} + \frac{SMAPE_i}{\sum_{k=1}^{N} SMAPE_k} \right) \tag{32}$$

5 Experimental evaluation

The dataset and individual techniques have been introduced in the previous sections, and the ensemble techniques to combine them. In this section, the evaluation of the combination of the proposed techniques will be described. A three-step approach will be used for this evaluation. First, the details of the testing framework will be described. Secondly, the optimisation of the techniques' parameters will be discussed. Finally, the results of the different configurations will be discussed.

5.1 Time series k-fold cross validation

The performance of the ensemble model in the training process is assessed using Time series k-fold cross-validation. As mentioned in Section 4.2, k, or the exact number of folds used in this cross-validation procedure, is treated as a hyperparameter to optimise. A larger k may lead to a bias reduction but a higher variance, while a small k may lead to a high bias but lower variance. Also, a considerable k may lead to an increase in computational time. At the same time, the number of test samples decreases, which also decreases the reliability of the final score. Since there is no single well-established method for determining the correct k, the more empirical Elbow method is used. In the Elbow method, the number of folds is displayed on the x-axis, and the average COMB score utilising the k-folds is shown on the y-axis. Finally, the number of k is chosen according to where a kink in the graph may be found, similarly to an elbow.

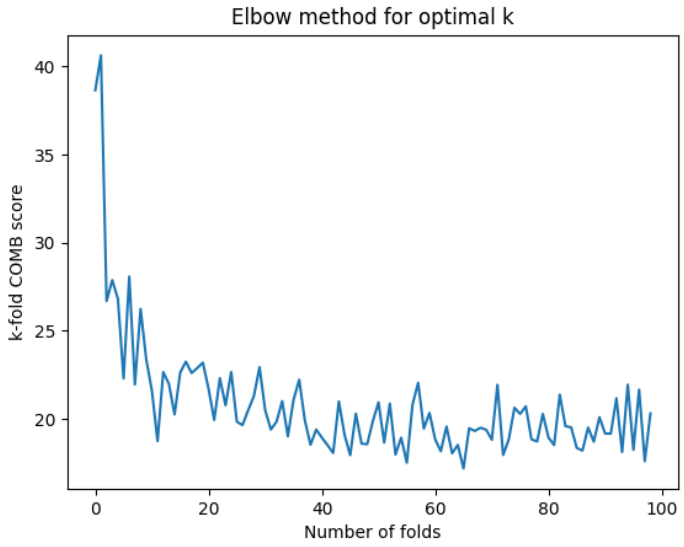


Figure 6: Elbow plot showing the number of folds plotted against the COMB score

Consequently, in Figure 6, the corresponding Elbow plot is depicted. As can be seen in this figure, a kink occurs in the performance around approximately five folds. Therefore, the choice has been made to use five folds in the model-fitting process.

5.2 Parameter optimisation

Many techniques have been described in Section 3, and for these techniques to be able to perform optimally, the hyperparameters of these techniques have to be tuned. The addition of each hyperparameter to the global search space increases the number of possibilities exponentially. Therefore, the TPE optimiser algorithm was chosen to mitigate this exponential effect on training times, as described in Section 3.4. The results of the optimisation are shown and described in the following sections.

5.2.1 Adaptive z-score normalisation

In Section 2, the dataset and the technique to normalise the data have been discussed. The adaptive z-score normalisation is shown in Equation 2, and the hyperparameter of this method is θ. For this parameter, the same holds as for the h parameter in the MS model; a higher θ results in an increase in smoothing of the parameter, similar to the θ in the moving average model, and a lower θ will result in higher volatility but will reflect newer movements earlier. The optimisation framework has set $\theta = 20$, corresponding to almost a full day, which is not a very high θ but should smooth the data sufficiently to remove some noise. An example of adaptive normalisation being applied is given in Figure 7. Although the normalisation does not perfectly retain the entire shape of the original time series, it accomplishes its task of refitting the data to a time-dependent standardised scale while retaining the key features.

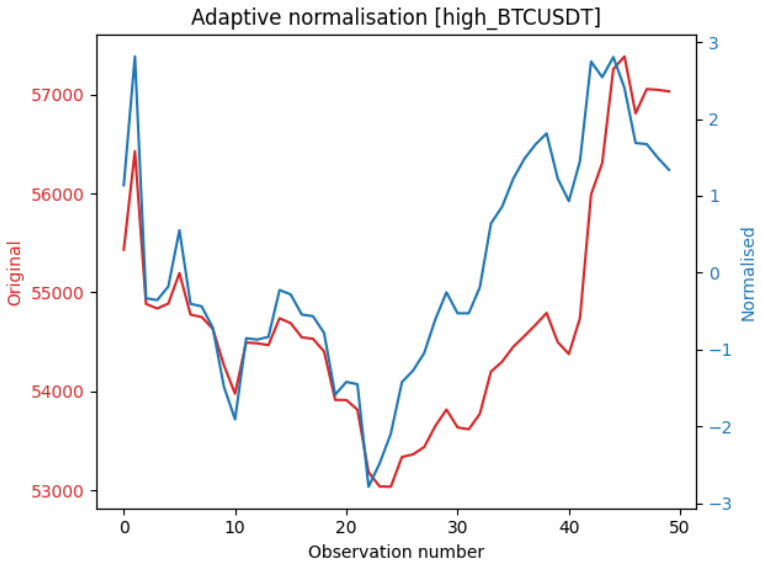


Figure 7: Adaptive normalisation applied on the last 50 observations of the BTC/USDT pair's high price, retrieved from the training dataset. The red axis and line show the original data, while the blue axis and line show the adaptive-normalised data

5.2.2 Discrete Wavelet Transform

As mentioned earlier, the DWT decomposition requires the fitting of all parameters in Equation 5, as well as the optimal choice of the mother wavelet. The optimisation framework has concluded that the optimal threshold value is 0.63, together with the choice of the Symlet as the mother wavelet with a depth level of five. The Symlet mother wavelet is shown in Figure 8, which is almost symmetric but still allows for high volatility on both sides, resulting in a logical fit for the highly volatile cryptocurrency prices.

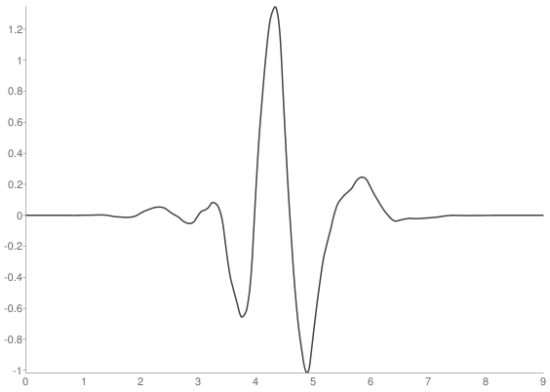
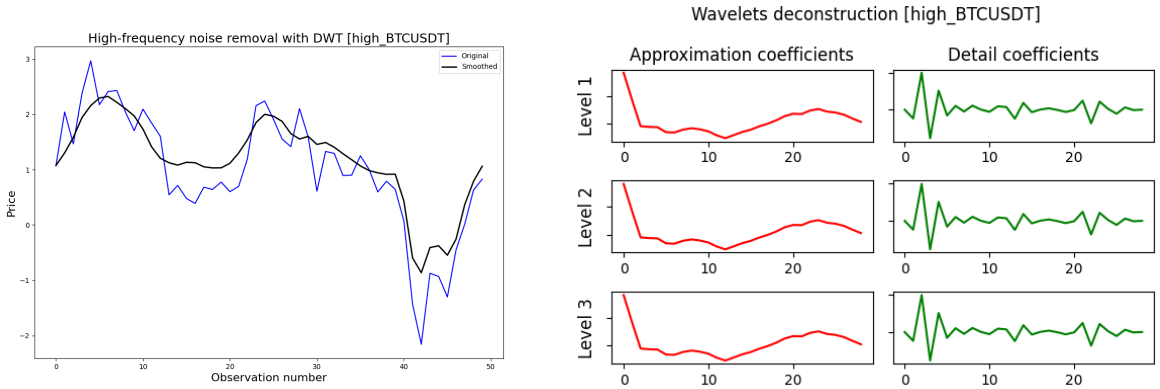


Figure 8: The symlet wavelet with a depth of five.

An example of the results of the DWT smoothing using these parameters is shown in Figure 9a, and the first three levels of the wavelet decomposition are shown in Figure 9b. These figures show that the DWT algorithm can smooth out a feature and thus remove the noise. The figure depicting the wavelet deconstruction also shows that the extracted noise in the first three levels is similar, implying that a single-layered filter already captures almost all the noise. However, it is uncertain if solely the noise is removed or the time series' essential features.



(a) The illustration of the DWT smoothing (b) The first three levels of the deconstruction

Figure 9: Discrete Wavelet Transform and its deconstruction applied on the last 50 observations of the BTC/USDT pair's high price, retrieved from the training dataset

5.2.3 Markov Switching

In Section 3.2.3, a detailed description of the Markov Switching technique was provided. As the (smoothed) probabilities are updated every $h = 3$ hours, showing the probabilities for more than 20,000 data points is too disordered. Therefore, the last period of the training dataset and the periods identified as State 0 are shown in Figure 10. It is not clear what kind of states or regimes the MS model extracts. As mentioned earlier, possible solutions could be the states reflecting expansion and recession periods or high and low volatility periods. If we look at the figure, the opening price of the BTC/USDT pair seems in an uptrend during the three months. In case State 0, which is shown in grey in the figure, would be interpreted as either a recession or expansion period, then the MS model would perform relatively poorly in the states. However, if we would take the other possible solution, which is classifying State 0 as a low volatility regime, and the white periods or State 1 as the high volatility regime, it would make much more sense. The figure also shows that, in general, when the price makes a significant move either up or down, it classifies the period as State 1, which would correspond to a high volatility state. However, between 15-12-202 and 01-01-2021, the MS technique classifies a significant part of the period as a high volatility regime, even though it looks like the price does not move sufficiently to be labelled as a high volatility state.

Consequently, incorrectly identifying a state could be very hurtful for the model's prediction. A reason for the errors could be the violation of the assumptions of the MS model, such as the normally distributed returns, which has been shown not to hold in Section 2.2. The violation of the assumptions could lead to the MS model performing poorly.

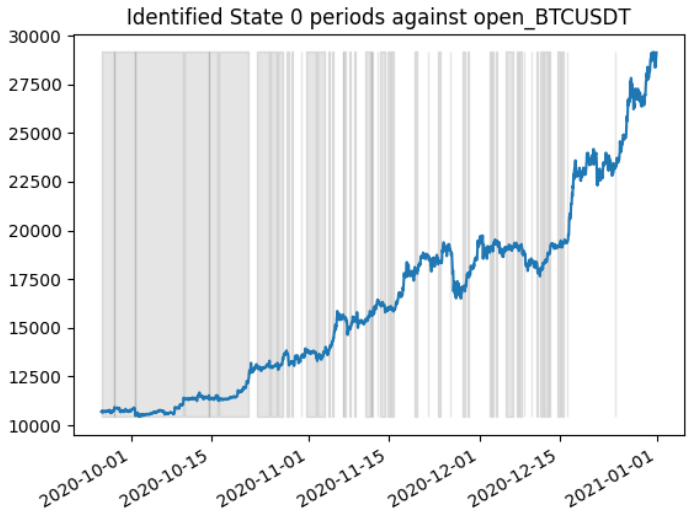
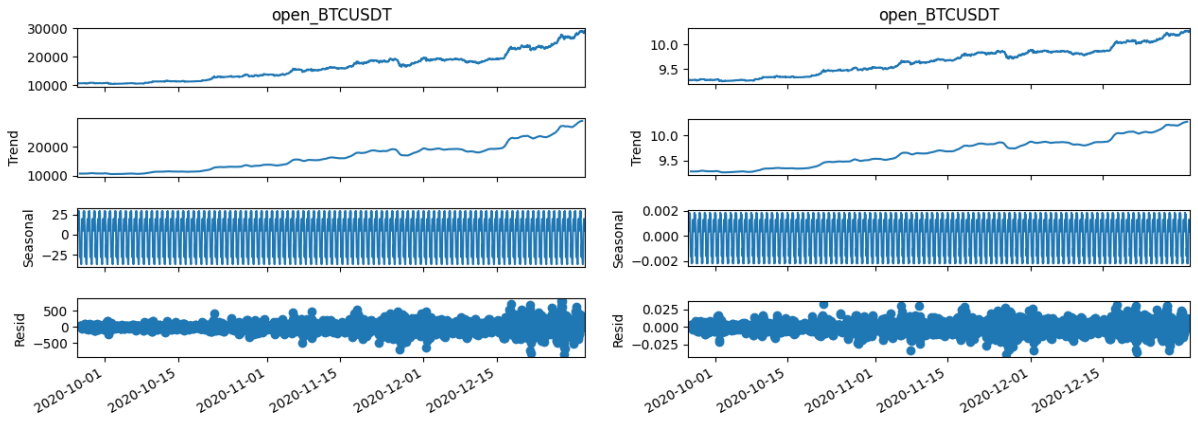


Figure 10: Periods corresponding to State 0 as identified by the Markov Switching technique, shown in grey, against the original price of the BTC/USDT pair from 25-09-2020 until 31-12-2020, shown in blue

5.2.4 Exponential Smoothing

In Section 3.2.2, the different configurations are shown in Equations 8, 9, 10, and 11. These configurations were all inserted in the optimisation framework to analyse and compare. The resulting final optimal configuration corresponds to Equation 11's ES configuration, which does not allow for seasonality.

$$Y(t) = T(t) + S(t) + e(t) \tag{33}$$



(a) Using the open price

(b) Using the log of the open price

Figure 11: The simple seasonal decomposition from Equation 33 applied on the open price and the natural logarithm of the open price of the BTC/USDT pair from 25-09-2020 until 31-12-2020

To be able to explain the reasoning behind this result, Equation 33 is used. First, the trend is estimated using a convolutional filter, and then the trend is removed to allow the seasonal pattern to be extracted from the de-trended series. The result is shown in Figure 11a, and the extracted seasonal pattern shown in the third row is very noisy. The residuals seem to increase in size, which could be explained due to the increasing trend. One solution may be to take the natural logarithm of the price and use that as the independent variable. The seasonal decomposition of this transformed variable is shown in Figure 11b. However, this transformation does not improve the result either, so the optimal model remains the configuration without the seasonal component. If the whole dataset is taken to be inserted in this method, then the seasonal subgraph is entirely filled, indicating that no general seasonal pattern could be found in the dependent variable by the simple algorithm.

Additionally, this phenomenon seems in contrast to the result found in Section 2.2, where a seasonal pattern is found in the month December. However, preferring a non-seasonal configuration may not contradict this finding since perhaps the ES Model could not extract this seasonal pattern. Also, it could be that the seasonal pattern found manually may not compensate for the model's shortcomings, which resulted in extracting erroneous seasonal patterns as shown in Figure 11.

5.2.5 Ensemble machine learning model optimisation

Two machine learning models have been implemented as an ensemble model, and following the optimisation engine, the optimal configurations have identified. These configurations are shown in Table 3. The RNN-LSTM model contains a single hidden layer with 4 LSTM units, processes batches of data with a size of 19, and iterates an approximate 466 epochs per sample. For the LightGBM model, the Gradient Boosting Decision Tree (GBDT) type is used, which is described in a more detailed fashion in Section 3.3.2. Also, the maximum number of leaves is set to 31, and the number of iterations is kept at 100. Due to the larger number of leaves, the LightGBM model seems more complex and should hence allow for more flexibility than the neural network. On the other side, a more complex configuration can illustrate signs of overfitting.

Technique	Learning rate	LSTM units	Batch size	Epochs
RNN-LSTM	0.01	4	19	466
Technique	Learning rate	Boosting type	Number of trees	Number of leaves
LightGBM	0.0682	GBDT	100	31

Table 3: The optimal parameters for the individual ML models, found by the optimisation framework

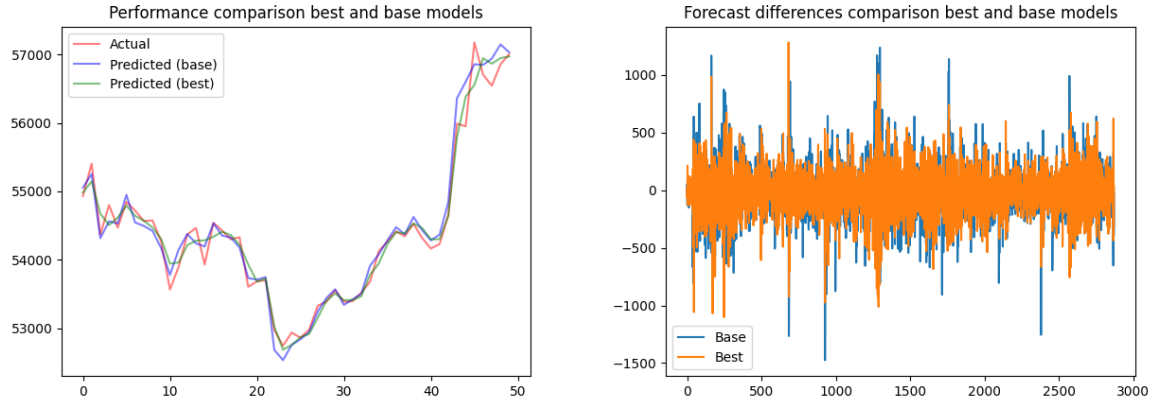
5.3 Model comparison

Finally, after all individual methods, techniques, and optimal parameters have been identified, the ensemble models combined with the new techniques may be compared. The configurations are compared using the performance metrics defined in Section 4.3. The COMB measure, shown in Equation 32, serves as the primary guideline. The results are displayed in Table 4, and the best model configuration is shown in bold.

Table 4: The overview of the model configurations used for comparison and evaluation and their corresponding performance metrics

Ensemble model	Stat. techniques	MASE	SMAPE	COMB
LightGBM & RNN-LSTM		4.8918	0.3905	1.6212
LightGBM & RNN-LSTM	DWT	5.0688	0.4047	1.6800
LightGBM & RNN-LSTM	MS	7.3991	0.5733	2.4159
LightGBM & RNN-LSTM	DWT & MS	8.982	0.7039	2.9493
LightGBM & RNN-LSTM & ES		**4.1969**	**0.3361**	**1.3931**
LightGBM & RNN-LSTM & ES	DWT	4.2793	0.3422	1.4195
LightGBM & RNN-LSTM & ES	MS	6.1762	0.4812	2.0221
LightGBM & RNN-LSTM & ES	DWT & MS	7.5917	0.5979	2.4989

This table shows that the best model configuration is the base ML ensemble model, adding an extra ES predictor and no other additional statistical techniques. Compared to the original model, the best model improved the forecasting performance with an approximate 14.1% in the COMB measure. In every configuration, adding the ES predictor to the ensemble model increased the performance, where the largest increase was approximately 15.5% for the configuration, including the MS technique. The predictions of this best model are plotted against the predictions of the base model, as well as the actual values, in Figure 12a.



(a) The forecast of the last 50 observations as well as the actual values

(b) The forecast differences over the entire test dataset

Figure 12: The comparison between the best and the base ensemble models over the test set

The figure shows that although both models are similar, the ensemble model, including the ES predictor, has an increased level of robustness compared to the base model. Furthermore, when the standard deviations of the differences between the predicted values and the actual prices are taken over the whole test set, the base model has a standard deviation of approximately 239.815, while the best model has a standard deviation of approximately 206.520. Ergo, the best model has a reduction in the standard deviation of 13.88%. The increase in robustness is depicted in Figure 12b, where the forecast differences of the base model exceed the values of the best model on average.

On the other hand, the configurations with the other statistical techniques that modify the dataset perform worse. As mentioned earlier, a likely reason for the DWT technique performing worse could be that the essential features in the time series are also smoothed away, resulting in a higher difficulty for the models to predict the actual data instead of the smoothed data. Especially since the ML models have a complex structural basis, it could also allow them to have increased flexibility. Consequently, applying the DWT technique that removes essential features other than the noise would limit the ML models' ability to flexibly model the original time series and thus also deteriorate their forecasting ability.

Subsequently, adding the MS technique allows the model to identify different regimes, and these could provide the models with a beneficial tool to enhance their forecasting ability. However, there is neither a definitive label for each period nor a central governmental institution that officially declares a period to classify as a recession. Therefore, these periods have to be estimated, and estimations are never flawless. If the errors are insignificant, applying the MS technique can still be deemed beneficial to the ML ensemble model. Following the results shown in the table, unfortunately, the MS algorithm also

does not allow itself to be a valuable addition to the base ML ensemble model.

Additionally, including both the DWT and the MS technique in the configuration yielded the worst performances, where the configuration without the ES predictor yielded the worst performance of the entire experiment. Instead of complementing each other's weaknesses, they did the exact opposite and strengthened them instead, worsening the result in the process.

6 Conclusion

This research report has laid its emphasis on whether machine learning ensemble models can be improved with statistical techniques in the context of forecasting cryptocurrency prices. This question has been answered by analysing a mixture of statistical preprocessing techniques and an ensemble predictor model with an origin in machine learning on an extensive dataset with the ability to capture macroeconomic effects.

The extensive dataset is the result of a combination of two different datasets. The first dataset is the cryptocurrency dataset, which allows the usage of the Open, High, Low, Close (OHLC) price of the top five largest cryptocurrencies on the market. The second dataset contains the historical OHLC data of the SPX and the SHCOMP indices to allow for macroeconomic effects. The resulting combined dataset is normalised for the ML techniques, using a newly proposed method called adaptive z-score normalisation. This method utilises a sliding window with a variable parameter to allow for variable means and standard deviations to normalise the original data with and is fitted per feature.

Consequently, using the adaptive normalised dataset, a flexible ML ensemble model is used as a basis to improve upon. This basis comprises of an RNN with a hidden layer consisting of LSTM units and a highly efficient gradient boosting machine, LightGBM. Two ensemble methods are used, which are Bagging and Boosting. The first ensemble technique is used to combine multiple predictors, and the second method is used implicitly in the LightGBM method, which utilises boosting in its gradient boosting framework.

Subsequently, the ML ensemble model is extended with three statistical techniques in an individual and a combined fashion. A single statistical technique was added as an additional predictor, and the remaining two techniques focused on modifying the dataset. The first statistical technique is the ES model, which is added as the extra predictor. This well-known technique carries the ability to deseasonalise, adaptively denormalise, and de-trend the data in its model-fitting process. The second technique aims to remove noise from the data by applying DWT. This technique is a Fourier transformation at multiple scales and converts the data to a mother wavelet, extended with multiple high and low-frequency layers. Low-pass filters are applied on these high and low-frequency layers, which filters the data through a threshold, removing the data's noise. The last statistical technique that is evaluated intends to identify different market regimes within the data and provide the model with valuable data to improve its forecasting ability. The technique assumes that the data follows a normal distribution in either market regime and that the transition to a new state follows a first-order ergodic Markov chain. The transition probabilities and the distribution parameters are estimated using the Expectation-Maximisation algorithm, and the transition probabilities are smoothed using the Kim smoother.

Afterwards, the three statistical techniques are evaluated using time series k-fold cross-validation. This form of cross-validation splits the data into $k = 5$ folds and maintains

the temporal dependency in each fold. In the cross-validation, a combination of two performance metrics is used. This combined performance metric uses an evenly weighted combination of the SMAPE and the MASE to decrease the error in the performance metric itself and offers a more reliable measure of accuracy. This time series k-fold cross-validation is used to optimise the parameters of each data preprocessing technique and the hyperparameters of each model. The optimisation framework utilises the TPE optimiser and hence allows a faster optimisation in high-dimensional parameter space and fewer constraints.

Thereupon, all combinations of the statistical techniques are evaluated, and the resulting best-performing model is the sole addition of the ES predictor to the basis of the ML ensemble model. It manages to reduce the error in forecasting by approximately 14.1% compared to the base model, and in the best-case scenario, it manages to increase forecasting performance by approximately 15.5%. Also, it can accomplish a lower standard deviation of the forecasting error in terms of robustness, where it manages to decrease it by approximately 13.88%. On the other hand, the dataset-modifying techniques did not perform as well. Adding the DWT technique or the MS technique worsened the forecasting ability of the resulting model significantly. The combination of the DWT and MS technique, without the additional ES predictor, finishes with an average performance of less than 50% compared to the best model. A Logical reason for the lousy performance of the DWT technique is that it smoothens the original feature more than solely removing its noise. For the MS technique, all regimes are estimated every three hours, which due to the high granularity and other problems in assumptions, may lead to significant errors in forecasting the different market regimes. These significant errors logically lead to a deterioration in forecasting ability.

Finally, a conclusion can be provided to the research question of whether machine learning ensemble models can be improved with statistical techniques in the context of forecasting cryptocurrency prices. Following the statement by Januschowski et al. (2020) that a hard methodological distinction between the machine learning and the statistical fields does not exist, the conclusion in this research may be drawn that improving the ML ensemble model basis is possible, but not with all statistical techniques. The exclusive addition of the ES predictor to the ML ensemble model, resulting in a model utilising a mixture of ML and statistics, yields a significant increase in forecasting ability performance and its robustness.

7 Limitations and Further Research

Although the research yielded a fruitful result, some difficulties and limitations were also encountered during the research. These limitations primarily concern the complexity of specific models and their combinations. As a result, the run times of some experiments would explode. Not to mention that all models were running in multi-threaded environments with GPU support enabled, if possible. Although these speedups could range in the tens or hundreds, running the optimisation and experiments on three servers still took two to three weeks. Therefore, the following suggestions to improve upon this research, assuming that there are no computational power constraints, are given in the following paragraphs.

The first suggestion is to consider more data transformations and modifications on the features or the dependent variable, such as taking the natural logarithm of the price data to improve its normality or include dummy variables corresponding to exogenous events to improve forecasting performance.

The second suggestion is to decrease the granularity when the MS model is refitted. Less variance in the data fed into the MS model should lead to a better fit of the MS model, thus decreasing error in classifications of recessions. Also, a normal distribution is currently assumed for the process, which may be changed to another distribution that fits the data better.

Another suggestion is that the MS probability of a recession is not added as a feature in the data. Instead, the time series could be split up in time series adhering to the two regimes, and two mixture ensemble models would be fitted for either of the periods. When a forecast is made, the probability of the first regime is determined by the MS technique, called p. In parallel, predictions are made by both mixture ensemble models, $\hat{y}_1$ and $\hat{y}_2$ for the model corresponding to the first and second state, respectively. These forecasts are then combined into a single forecast using the probability of the MS period, as shown in Equation 34.

$$\hat{y}_{12} = p * \hat{y}_1 + (1 - p) * \hat{y}_2 \tag{34}$$

The final suggestion is to change the forecast weighting of the ensemble model. Currently, the weights function, which combines the individual forecasts, takes the average of the forecasts. However, this simplification of the general function shown in Equation 22 is done to ease the process and run time but does not have to be the optimal solution for the predictors.

Appendices

A Summary statistics dataset

Table 5: Table giving an overview of the summary statistics of the entire dataset. The dependent variable is shown in bold, and Corr indicates the correlation with the dependent variable

Feature	Mean	Variance	Skewness	Kurtosis	Min	Max	Corr
Open SHCOMP	2957.6293	56346.0	0.2201	-0.6492	2443.4974	3469.5924	0.6723
High SHCOMP	2962.9558	56714.0	0.2237	-0.6524	2464.3628	3474.9182	0.6703
Low SHCOMP	2952.0937	56211.0	0.2186	-0.65	2440.9066	3467.8217	0.6753
Open BNBUSDT	18.1718	62.0	0.4084	-0.648	4.1599	39.5193	0.7395
High BNBUSDT	18.2863	63.0	0.4109	-0.646	4.2047	39.9999	0.7389
Low BNBUSDT	18.0442	61.0	0.4058	-0.6502	4.12	39.0307	0.74
Open SPX	2989.2225	84283.0	0.5677	-0.1442	2208.87	3750.01	0.7952
High SPX	2995.4072	83656.0	0.5917	-0.1553	2245.88	3760.2	0.798
Low SPX	2983.0199	84876.0	0.5402	-0.1163	2191.86	3745.02	0.7929
Open ETH/USDT	265.6003	21734.0	1.3559	1.2273	82.16	839.5	0.6255
High ETH/USDT	267.2193	22025.0	1.3597	1.2412	82.95	839.99	0.625
Low ETH/USDT	263.8279	21406.0	1.3524	1.2149	81.79	830.6	0.6253
Open BTC/USDT	8508.5956	13930475.0	1.7866	5.4663	3172.62	29155.24	1.0
High BTC/USDT	8547.7182	14095618.0	1.7962	5.5179	3184.75	29300.0	0.9999
Low BTC/USDT	8465.8355	13749932.0	1.777	5.4153	3156.26	28960.17	0.9998
Open ADA/USDT	0.082	0.0	1.817	4.709	0.0201	0.3729	0.3899
High ADA/USDT	0.0827	0.0	1.8188	4.7013	0.0224	0.3743	0.3898
Low ADA/USDT	0.0814	0.0	1.8181	4.7315	0.0176	0.3667	0.3892
Open XRP/USDT	0.327	0.0	1.3276	1.838	0.1194	0.9321	0.0492
High XRP/USDT	0.3293	0.0	1.3318	1.8333	0.1336	0.935	0.0537
Low XRP/USDT	0.3246	0.0	1.3272	1.8658	0.1013	0.915	0.0441

B Pseudocode LightGBM

B.1 Pseudocode GOSS

Algorithm 1: Pseudo code of the Gradient-based One-Side Sampling (GOSS) algorithm, retrieved from Ke et al. (2017)

input: $I \leftarrow$ training data, $d \leftarrow$ iterations, $a \leftarrow$ the sampling ratio of large gradient data, $b \leftarrow$ the sampling ratio of small gradient data, $loss \leftarrow$ loss function, $L \leftarrow$ weak learner

models $\leftarrow$ { }, fact $\leftarrow \frac{1-a}{b}$

topN $\leftarrow a \times$ length(I), randN $\leftarrow b \times$ length(I)

for $i \leftarrow 0$ **to** d **do**

- preds $\leftarrow$ models.predict(I)
- g $\leftarrow loss(I, \text{preds}), w \leftarrow \{1, 1, \dots\}$
- sorted $\leftarrow$ GetSortedIndices(abs(g))
- topSet $\leftarrow$ sorted[1:topN]
- randSet $\leftarrow$ RandomPick(sorted[topN:length(I)], randN)
- usedSet $\leftarrow$ topSet + randSet
- w[randSet] = w[randSet] $\times$ fact $\triangleright$ Assign weight $fact$ to the small gradient data
- newModel $\leftarrow$ L(I[usedSet], - g[usedSet], w[usedSet])
- models.append(newModel)

end

B.2 Pseudocode EFB

Algorithm 2: Pseudo code of the Exclusive Feature Bundling (EFB) algorithm, retrieved from Ke et al. (2017)

```
input  : numData ← number of data, F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
Output: newBin, binRanges
for f in F do
    totalBin = totalBin + f.numBin
    binRanges.append(totalBin)
end
newBin ← new Bin(data)
for i ← 0 to numData do
    newBin[i] ← 0
    for j ← 0 to length(F) do
        if F[j].bin[i] ≠ 0 then
            newBin[i] ← F[j].bin[i] + binRanges[j]
    end
end
```

C Schematic flow testing procedure

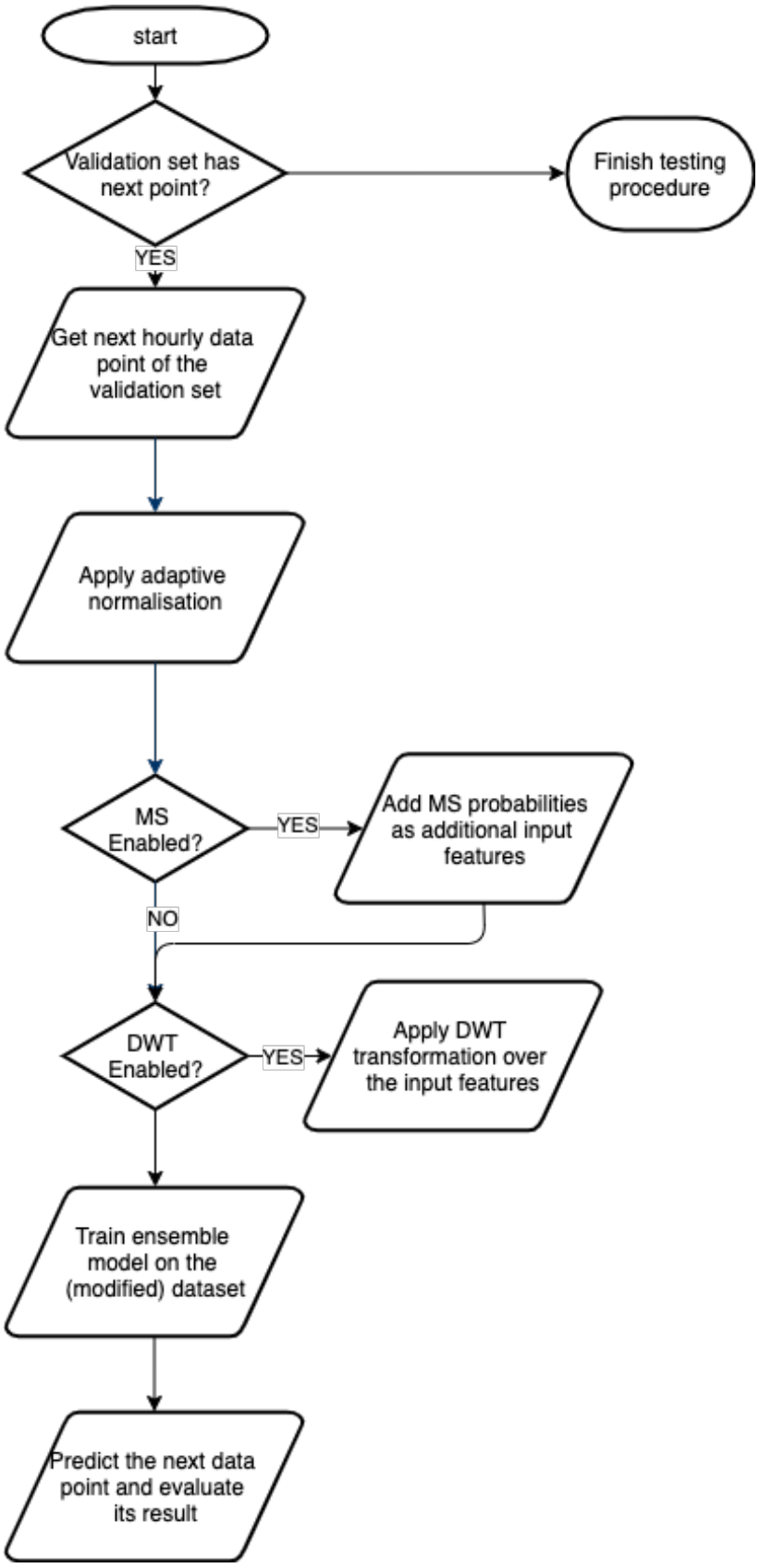


Figure 13: Schematic flow diagram illustrating a simplified testing procedure

References

P. S. Addison, *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2017.

N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

M. Al Wadia and M. T. Ismail, "Selecting wavelet transforms model in forecasting financial time series data based on arima model," *Applied Mathematical Sciences*, vol. 5, no. 7, pp. 315–326, 2011.

J. S. Armstrong, "Findings from evidence-based forecasting: Methods for reducing forecast error," *International Journal of Forecasting*, vol. 22, no. 3, pp. 583–598, 2006.

N. A. Bakar and S. Rosbi, "Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction," *International Journal of Advanced Engineering Research and Science*, vol. 4, no. 11, p. 237311, 2017.

Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *25th annual conference on neural information processing systems (NIPS 2011)*, vol. 24. Neural Information Processing Systems Foundation, 2011.

J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.

T. A. Borges and R. F. Neves, "Ensemble of machine learning algorithms for cryptocurrency investment with different data resampling methods," *Applied Soft Computing*, vol. 90, p. 106187, 2020.

L. Breiman *et al.*, "Statistical modeling: The two cultures (with comments and a rejoinder by the author)," *Statistical science*, vol. 16, no. 3, pp. 199–231, 2001.

P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of computational statistics*. Springer, 2012, pp. 985–1022.

J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

E. S. Gardner Jr, "Exponential smoothing: The state of the art—part ii," *International journal of forecasting*, vol. 22, no. 4, pp. 637–666, 2006.

A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.

J. D. Hamilton, "A new approach to the economic analysis of nonstationary time series and the business cycle," *Econometrica: Journal of the econometric society*, pp. 357–384, 1989.

——, "Analysis of time series subject to changes in regime," *Journal of econometrics*, vol. 45, no. 1-2, pp. 39–70, 1990.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M. Bohlke-Schneider, and L. Callot, "Criteria for classifying forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 167–177, 2020.

B. Kaiser, M. Jurado, and A. Ledger, "The looming threat of china: An analysis of chinese influence on bitcoin," *arXiv preprint arXiv:1810.02466*, 2018.

L. Kaiser, "Seasonality in cryptocurrencies," *Finance Research Letters*, vol. 31, 2019.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

A. M. Khedr, I. Arif, P. R. P V, M. El-Bannany, S. M. Alhashmi, and M. Sreedharan, "Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey," *Intelligent Systems in Accounting, Finance and Management*, 2021.

C.-J. Kim, "Dynamic linear models with markov-switching," *Journal of Econometrics*, vol. 60, no. 1-2, pp. 1–22, 1994.

D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

S. Kolassa, "Why the "best" point forecast depends on the error or accuracy measure," *International Journal of Forecasting*, vol. 36, no. 1, pp. 208–211, 2020.

E. Kole and D. Van Dijk, "How to identify and forecast bull and bear markets?" *Journal of Applied Econometrics*, vol. 32, no. 1, pp. 120–139, 2017.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

I. E. Livieris, E. Pintelas, S. Stavroyiannis, and P. Pintelas, "Ensemble deep learning models for forecasting cryptocurrency time-series," *Algorithms*, vol. 13, no. 5, p. 121, 2020.

J. Ma, J. S. Gans, and R. Tourky, "Market structure in bitcoin mining," National Bureau of Economic Research, Tech. Rep., 2018.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m5 accuracy competition: Results, findings and conclusions," *Int J Forecast*, 2020.

S. Makridakis, "Accuracy measures: theoretical and practical concerns," *International journal of forecasting*, vol. 9, no. 4, pp. 527–529, 1993.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

——, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

J. Nobre and R. F. Neves, "Combining principal component analysis, discrete wavelet transform and xgboost to trade in the financial markets," *Expert Systems with Applications*, vol. 125, pp. 181–194, 2019.

R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

M. J. Shensa *et al.*, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on nuclear science*, vol. 44, no. 3, pp. 1464–1468, 1997.

X. Sun, M. Liu, and Z. Sima, "A novel cryptocurrency price trend forecasting model based on lightgbm," *Finance Research Letters*, vol. 32, p. 101084, 2020.

S. Yi, Z. Xu, and G.-J. Wang, "Volatility connectedness in the cryptocurrency market: Is bitcoin a dominant cryptocurrency?" *International Review of Financial Analysis*, vol. 60, pp. 98–114, 2018.