NOVA IMS Information Management School

MGI

Master Degree Program in
Information Management

Time series forecasting by combining LSTM RNN with ARIMA method

Akvilina Akstinaitė

Dissertation
presented as partial requirement for obtaining the Master Degree Program in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

TIME SERIES FORECASTING BY COMBINING LSTM RNN WITH ARIMA METHOD

By

Akvilina Akstinaitė

Master Thesis presented as partial requirement for obtaining the Master’s degree in Information Management, with a specialization in Knowledge Management and Business Intelligence

Supervisor: Roberto Henriques

November 2022

INDEX

LIST OF ABBREVIATIONS AND ACRONYMS

DLNN	Deep Learning Neural Network
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
SARIMA	Seasonal ARIMA
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
BPTT	Backpropagation Through Time
ADF	Augmented Dickey Fuller
ACF	Autocorrelation Function
PACF	Partial Autocorrelation Function
AIC	Akaike Information Criterion
TPA	Turning Point Accuracy
APA	Action Prediction Accuracy
AIC	Akaike Information Criterion

Abstract

This study aims to analyze the performance of the ensemble model – the combination of Long Short-Term Memory Recurring Neural Network model with the ARIMA model. We developed these models separately to perform the best on their own in predicting prices 1, 7, and 14 observations ahead while taking into account the last 30, 60 and 90 observations and checking if the combination of them outperforms the standalone models. We evaluated the models based on RMSE and their ability to predict the turning points. Models were developed and tested on two different types of securities – index S&P 500 and cryptocurrency Bitcoin (BTC). The combined methods demonstrated strong performance on BTC data set and gave at least 90% turning point prediction accuracy when predicting the price for one observation ahead. For the S&P 500 data set, the performance of the stacked model was poor – it outperformed the standalone models only in one test out of eighteen – while predicting prices one observation ahead, looking back at the past 30 observations.

Keywords Reoccurring Neural Networks, Deep Learning, Long Short-Term Memory, Financial Time Series, ARIMA.

1 Introduction

Analyzing financial time series has been an area of interest for the past few decades and has an essential role in the financial market. The researchers are interested in tackling the challenging task of predicting financial time series data due to the volatility and unknown changes in the economic situation. On the other hand, laypeople also started to gain interest in this field. They want a quality, wealthy life which requires more financial resources. Working more and saving up does not seem appealing anymore – it results in less free time, making people unproductive, less happy, and unhealthy (A. V. Whillans, 2019). Also, stress caused by issues concerning time affects happiness more negatively than unemployment (A. Whillans, 2020). In addition to the need for more financial resources, people are more willing to understand that money is constantly losing its buying power. This leads people to look for alternative ways to increase their income - one of them is investing.

There are two main investment strategies – passive and active. While passive investment is a long-term investment strategy, also called buy-and-hold strategy, that aims to benefit from the expected market price increase over time; active investment strategy seeks short-term profits from actively buying and selling assets. This investment strategy is relevant for financial time series analysis, more precisely – stock price predictions. While some laypeople try to make the predictions themselves and avoid giving the decision power to their counterparts, others tend to trust experts in their judgment on forecasts in the stock market(Huber et al., 2019). Hence financial institutions need to have a strategy and valid forecasting models that demonstrate financial institutions' ability to use and divide the investment portfolio to obtain the gains promised. Due to the fast growth of the financial market and higher accessibility to historical data, we can use time series forecasting, which utilizes historical time-stamped data to make future predictions. It is a helpful approach for the development of forecasting models. Hence, lately, this approach has been used extensively in developing models for stock price prediction (Brockwell & Davis, 2016; Shen & Shafiq, 2020).

In this work, we will focus on comparing performance of several time series forecasting methods - an ensemble model based on ARIMA and LSTM models and standalone models. Comparisons will be made for price predictions on two different time series – index S&P 500 and Bitcoin (BTC) cryptocurrency historical daily prices. For both methods, we will develop models that predict the stock open and close prices 1, 7, and 14 observations ahead based on the past 30, 60, and 90 observations. To make the ensemble model, we will use the stacking strategy. To compare the models, we will use statistical parameters RMSE and MAPE and their ability to predict turning points.

The structure of this paper is as follows: section 2 presents the review of relevant studies on stock market prediction methods; section 3 makes the description of the methodology; section 4 describes the data sets and models used in the methodology; section 5 presents the results of the experiments performed and; section 6 makes the conclusions and presents future work.

2 Related work

There are many methods to choose from when deciding which to use for financial time series forecasting. Some well-known ones are ARIMA family models, neural networks, and ensembles (Brockwell & Davis, 2016; Christiansen, 2018; Mauldin et al., 2021; Shen & Shafiq, 2020; Wolpert, 1992).

2.1 ARIMA

ARIMA method was introduced half a century ago by Box and Jenkins(Box & Jenkins, 1970). According to Stellwagen's and Tashman's (Stellwagen & Tashman, 2013) review, ARIMA models were not widely used in business mainly because of the difficulty of finding a suitable form of the model for the data set given. However, that changed with the discovery of methods for automatically finding the best ARIMA model. One of the most well-known methods is the Akaike information criterion (AIC) minimization method (Akaike, 1974, 1979),(Guthery et al., 2003). ARIMA is one of the traditional forecasting techniques, but it is more sensitive to variability and does not perform as well when there is no pattern in the data (Kolarik & Rudorfer, 1994). Also, this method can use only one factor for predictions – the same factor you want to predict.

2.2 Neural Networks

Another well-known and currently most used approach to forecasting is Artificial Neural Networks. It was the first time used in time series forecasting in 1964 (Hu & Root, 1964). More than two decades later, Rumelhart, Hinton, and Williams (Rumelhart et al., 1986) introduced a backpropagation algorithm called Recurrent Neural Network, which was further formulated by Werbos (Werbos, 1988). By then, the number of publications where ANNs were applied to forecasting increased substantially. Zhang, Patuvo and Hu (G. Zhang et al., 1998) presented a survey on all the research about ANN applications in forecasting. They mention research comparing ANNs (specifically MLP models) with traditional statistical methods, one of them being ARIMA. Most of them find ANNs to be superior to ARIMA, but one research finds that "ANNs are hardly better than ARIMA" for short-term predictions (1-step ahead) (Caire et al., 1992). ANNs are so popular in the forecasting field because of their wide range of applications and the low amount of information on the data needed for analysis. Kolarik and Rudorfer (Kolarik & Rudorfer, 1994) used artificial neural networks to analyze financial time series. They also mention one of the main advantages of neural networks over ARIMA – these models are able to describe nonlinear time series and are able to deal with increased variance. In their research, authors find neural networks to be "viable alternatives" to already widely used techniques, and they also point out that ANNs can be used alone or in ensemble models. Moreover, ANNs can be not only univariate, but also multivariate.

In 1997 Hochreiter & Schmidhuber introduced a kind of RNN called Long Short-Term Memory network(Hochreiter & Schmidhuber, 1997). It is one of currently the most attraction grabbing DLNN models. This kind of network was proposed to overcome the main problem of RNNs – carrying over "long-term dependencies". Bengio, Simard and Frasconi explored this problem in-depth (Bengio et al., 1994) and found that this problem occurs due to vanishing and exploding gradients. LSTM networks are designed to tackle this long-term dependency problem. Moreover, these networks are capable of learning how to connect time intervals that have more than 1000 steps even when the input sequence is noisy and incompressible, without losing its ability to remember short-term information. However, some research find that DL models are unreliable in cryptocurrency price predictions (Pintelas et al., 2020). Also, these models can be too complex for a smaller data sets that can lead to overfitting (Mauldin et al., 2021).

2.3 Ensemble

Another approach to the forecasting is ensemble method. Previously mentioned research on DNNs models application for cryptocurrency prices prediction suggest that ensemble methods should be investigated to obtain more reliability in the model (Pintelas et al., 2020). In the research that investigates the ensemble DL models for cryptocurrency price prediction (Livieris et al., 2020) - to make the model more reliable authors combine three deep learning models – LSTM, Bi-directional LSTM and Convolutional NN with an ensemble method (averaging, bagging and stacking strategies used) which in the end leads to development of '*strong, stable, and reliable forecasting models*'. Besides that, ensemble methods are known because of their ability to outperform individual models also in ensemble model of NN and traditional model - the combination of SARIMA and neural networks in seasonal time series forecasting has been tested (Tseng et al., 2002) and the results showed that the combined model is superior to stand alone models. Moreover, in one research it was found that the ensemble mean often has an error that is 30% smaller than the median error of the individual ensemble member (Christiansen, 2018) and it helps to deal with the overfitting problem in LSTM models composed on small datasets (Mauldin et al., 2021).

2.4 Research gap

The performance of ARIMA and LSTM methods on financial time series were compared multiple times, and it was found that LSTM performs better than ARIMA (Kolarik & Rudorfer, 1994; Liu, 2022; Ma, 2020; Mahadik et al., 2021; Siami-Namini et al., 2019; Siami-Namini & Namin, 2018). Moreover, one of these studies showed that the LSTM model outperformed the ARIMA model and reduced error rates by 84 – 87 percent (Siami-Namini et al., 2019). In these previous researches ARIMA and LSTM there mostly being compared against each other. Nevertheless, they could benefit from each other – ARIMA can be only univariate. Hence combining it with LSTM would allow us to introduce more factors in the analysis. Also, with LSTM, we get the ability to identify nonlinear relationships, and at the same time, the ARIMA method could help avoid overfitting in the LSTM method. Considering the previous research on using a combined model of SARIMA and LSTM(Tseng et al., 2002) there is ground to believe that the combined model of ARIMA and LSTM would outperform independent models.

Moreover, more recent research investigated the application of the ARIMA and LSTM ensemble method in medical and energy fields and found that the ensemble model outperformed the individual models (Deng et al., 2020; Tang et al., 2019; Wang et al., 2020). The application of the DLNN and ARIMA ensemble model has also been tested in the financial field, specifically for stock price or exchange rate predictions (Fathi, 2019; Verma et al., 2022; Xiao & Su, 2022; P. G. Zhang, 2003). Though, the approach used in these researches is different than the approach we are going to use - (P. G. Zhang, 2003) does not use LSTM as a DLNN model, and the other three use different ensemble approaches. In (Verma et al., 2022) the authors are making an ensemble method by choosing the prediction that falls between the set threshold (5% price limit is considered for the threshold value). (Fathi, 2019)uses a different ensemble approach - ARIMA to obtain trend function (seasonal component and noise are retained for later), and then LSTM to use the retained seasonal component and noise to make the forecast of the original time series and (Xiao & Su, 2022) uses mean errors of LSTM and ARIMA models and the weights calculated based on them to obtain the hybrid model prediction and does not consider turning point as comparison option. Nevertheless, all of them demonstrated good results where the ensemble method did outperform the standalone models and (Verma et al., 2022) suggest the usage of different ensemble methods for the future research. Hence, this study introduces a different ensemble of LSTM and ARIMA models approach and aims to compare its performance against the standalone models for short term stock price prediction.

3 Methodology

In this paper we propose the methodology with the following steps illustrated in the Figure 1.

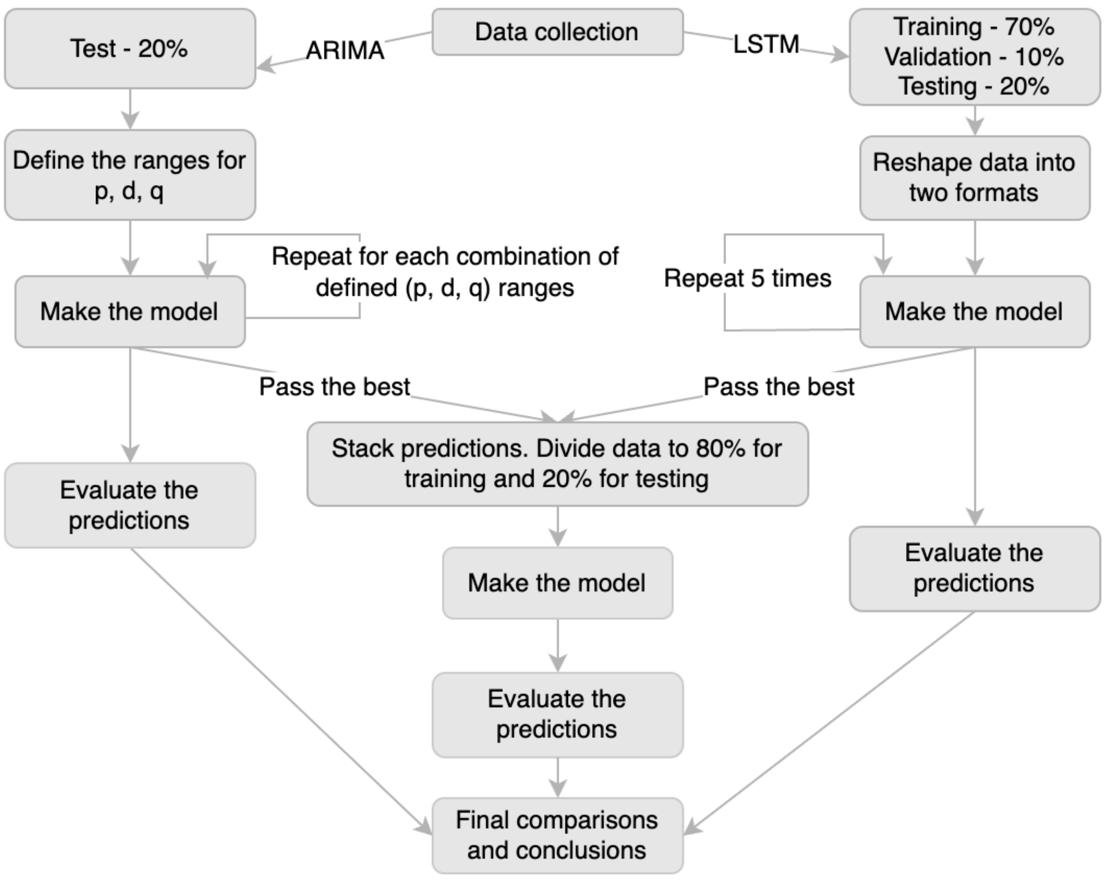


Figure 1 - Methodology

3.1 Data

We collected the data for the historical daily stock prices of the index S&P 500 and cryptocurrency Bitcoin (BTC) from YAHOO Finance for the period between Jan 2017 and May 2022. The dataset contains daily information about the date and the open, high, low and close prices. We will check if there are any missing values in the data sets, although we will not be removing any outliers, as financial time series grow in time. As LSTM uses sigmoid and tanh functions, which are sensitive to magnitude, we will standardize the prices using the Sciki-learn function StandardScaler. To make the comparison more accessible, we will develop all models on standardized data.

To make LSTM models we are going to divide the data set to train, validation and test datasets. For training we will use 70%, 10% for validation and for testing – 20% of the data. For LSTM model it is required to have each data set in two formats. The first format will contain the information to make the prediction and we will call it X, and the second format will have the information of what we want to predict – we will call it Y. The first format has to be in a shape of

$$\mathrm{n_{samples}} \times \mathrm{n_{past}} \times \mathrm{n_{features}}$$

where $\mathrm{n_{samples}}$ is the number of samples we are going to make the model on, n_{past} – timesteps we are going to look back at to make a prediction and $\mathrm{n_{features}}$ is the number of features we are going to be using. Each sample is going to have $\mathrm{n_{features}} \times \mathrm{n_{timesteps}}$ to predict the corresponding sample in the second format. Hence, the second format has to be in a shape of

$$\mathrm{n_{samples}} \times \mathrm{n_{features}}$$

where $\mathrm{n_{samples}}$ is the same as in the first format and $\mathrm{n_{features}}$ is the number of features we are going to be predicting. We are not going to use the resampling for this model, but we are going to train and validate each model 100 times (epochs) and we will save the best model based on lowest validation loss.

To fit the ARIMA models, we are going to use only one data set – test. This is because in our test case we want to make predictions based on the last 30, 60 or 90 observations, hence the model will be developed and the prediction will be made on them. Also, for ensemble model we will need to have the same size data set as for LSTM model. Hence this method does not require train and validation data sets to make the model and to make the predictions we will use only testing data set which will be the same size as for LSTM – 20% of the latest observations. Our chosen approach for ARIMA method also does not utilize resampling (where the train data set is divided into different samples and the best model is chosen), hence for training the model we are going to use the first 30, 60 or 90 ($\mathrm{n_{past}}$) observations from the test data set and modify this set with each iteration. After each prediction is made, we will append the data set with the next in line observation from test data set and delete the first observation from the used data set to make the predictions leaving us with the last 30, 60 and 90 observations to make accurate predictions in the following step. This procedure will be repeated until the predictions are made for the whole train data seta.

The data set for the ensemble model will consist of the predictions from both ARIMA and LSTM models – predictions made on the test data set which will form a new data set for ensemble. We will divide the new data set into 80% training and 20% testing. We will also use similar way of resampling as used for ARIMA method – after the prediction is made, we will append the train data set with the first observation from the test data set, but we will not be deleting any observations from train data set as the requirement to look at the specific amount of data to make the prediction is already satisfied while making the data set for ensemble itself.

3.2 ARIMA

According to Box and Jenkins (Box & Jenkins, 1970), time series $z_t, t = 0,1,2,\ldots$ is generated by $ARIMA(p,d,q)$ process, if

$$\phi(B)(1-B)^d z_t = \theta(B)a_t$$

where:

- $\phi(B) = \left(1 - \phi_1 B - \cdots - \phi_p B^p\right)$ and $\theta\,(B) = (1 + \theta_1 B + \cdots + \theta_q B^q$ are polynomial operators in B of degree p and q respectively. $\phi(B)$ is called the *autoregressive operator*

$AR(p)$ and $\theta(B)$ is called the *moving average operator* $MA(q)$. Also, these operators are assumed to be stationary, that is the roots of $\phi(B) = 0$ and $\theta(B) = 0$ lie outside the unit circle.

- $(1 - B)$ is differencing operator with d being the number of regular differences.
- z_t is the observed value at the time t.
- a_t is called Gaussian white noise with mean 0 and variance δ_a^2.
- B is backwards shift operator.

To define hyperparameters p, d, q we will use multiple methods. First, we will define the differencing operator d. Then we will use the *pmdarima* function *auto_arima*. This function discovers the optimal order of ARIMA model by first conducting differencing tests to determine d (This step will be skipped, as we will specify d to be used) and then fitting different models while minimizing AIC value. However, with this method we are not able to specify how many days ahead we want the model to predict, so additionally to this we will use the following method:

- Define the range for possible p values by using Partial autocorrelation function.
- Define the range for possible q values by using Autocorrelation function.
- Perform a grid search with different combinations of p, q and d values to find the best hyperparameters for each 30, 60 and 90 observations train data set predicting 1, 7 and 14 observations ahead. We will choose the best parameters based on the lowest RMSE value.

As mentioned before, we will be redeveloping the model in each iteration by introducing new values to the train data set:

1. Make the train data set - based on the first 30, 60 or 90 observations from test data set.
2. Train the model and make the prediction 14 time steps ahead
3. Save the 1st, 7th and 14th prediction (as it represents the observations ahead predictions we are aiming for).
4. Append training dataset with the actual value of the next observation and delete the first value in the training data set
5. Repeat steps 2-4 till predictions for the whole test data set are made.

This methodology is represented in Figure 2.

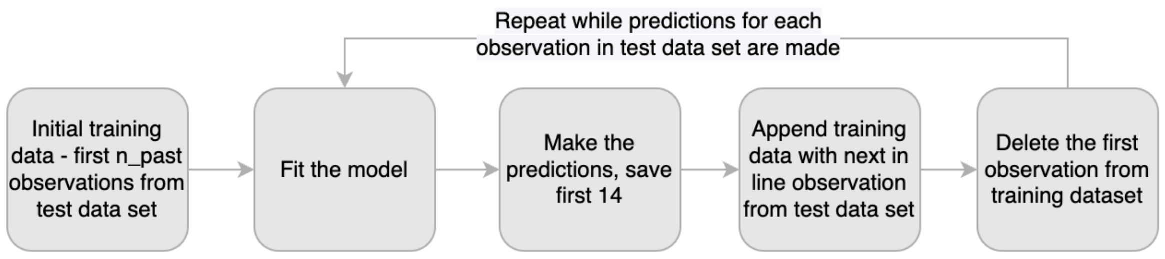

Repeat while predictions for each observation in test data set are made
Initial training data - first n_past observations from test data set
Fit the model
Make the predictions, save first 14
Append training data with next in line observation from test data set
Delete the first observation from training dataset

Figure 2 - ARIMA model methodology

3.3 LSTM RNN

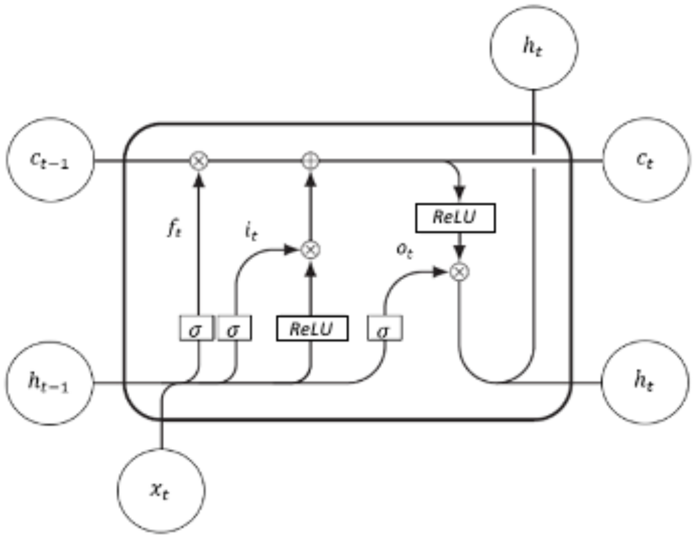


Figure 3 – LSTM architecture

The architecture of LSTM model can be seen above in Figure 3 and is defined by the following equations:

$$f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}x_{t-1} + b_f\right)$$

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_{t-1} + b_i)$$

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_{t-1} + b_o)$$

$$c_t = f_t * C_{t-1} + i_t * tanh(W_{hc}h_{t-1} + W_{xc}x_{t-1} + b_c)$$

$$h_t = o_t * \tanh{(c_t)}$$

where f_t, i_t, o_t, c_t and h_t represent the forget gate, input gate, output gate, memory cell and hidden state respectively. W and b are the weight matrix and bias vector, $\otimes$ and $\oplus$ denotes the element-wise product.

The main component of LSTMs is the memory cell c_t – the horizontal line in the top of the diagram. To control the cell state we have three gates – forget, input and output – that decide on what information is taken and what is left out by using sigmoid function σ. However, for the memory cell itself and the hidden state we will use rectifier linear activation function or ReLu, which helps with unstable gradients problem and has shown a better performance (Brownlee, 2016).

To predict the 'Open' and 'Close' prices with LSTM we are going to use all four features – "Open", "High", "Low" and "Close".

To choose the best model we are going to try out 5 different combinations of the number of hidden layers and neurons as follows:

- One layer with 32 neurons (code: SP_32 or BTC_32)
- One layer with 48 neurons (code: SP_48 or BTC_48)
- One layer with 64 neurons (code: SP_64 or BTC_64)
- Two layers where 1st layer with 32 neurons 2nd layer with 16 neurons (code: SP_32_16 or BTC_32_16)
- Two layers where 1st layer with 64 neurons 2nd layer with 32 neurons (code: SP_64_32 or BTC_64_32)

We are going to train models with different number of hidden layers and neurons 100 times each and will save the best out of these 100 tries. The

To select the best model for each prediction type we will use RMSE – the model with the lowest RMSE will be selected for stacking.

For each combination of hidden layers the methodology for making LSTM model is presented in Figure 4.

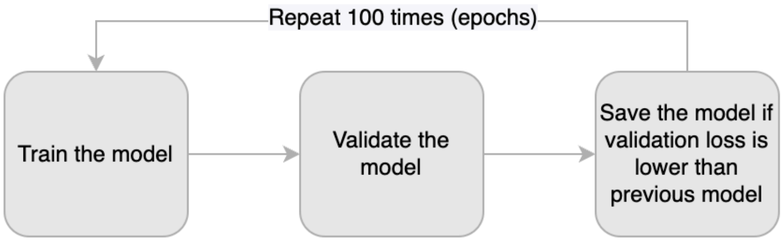

Repeat 100 times (epochs)
Train the model
Validate the model
Save the model if validation loss is lower than previous model

Figure 4 - LSTM model methodology

3.4 Ensemble Approach

We chose to use stacking as an ensemble method in this work. Stacking or stacked generalization is a technique which aims to achieve highest generalization accuracy possible (Wolpert, 1992). In this technique we use the first-level learners (base models) to generate a new data set called meta-dataset (made out of results of first-level learners) for second-level learner (meta learner). For regression tasks linear regression is usually used as meta learner. To avoid bias it is recommended to train first-level learners on a different training set than the one used to generate meta-dataset, therefore we are having 20% of the data set for testing

For stacking method we will use predictions from base learners. Depending on the number of observations to future we will be predicting, ARIMA predictions data set might be longer than from LSTM (, hence we will choose only applicable predictions from ARIMA model based on LSTM predictions interval. To make the stacking model we used similar approach as with ARIMA model:

1. Divide the dataset in train and test sets.
2. Fit the model on the training dataset (used Logistic Regression as meta learner) aiming for the corresponding value from the original test data set
3. Make predictions on new test data set and save the first prediction.
4. Append train data set with the corresponding value from original test dataset.
5. Repeat steps 1-4 for the whole test data set.

This methodology is represented in Figure 5.

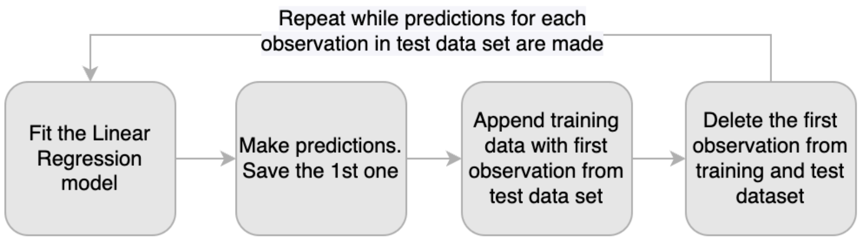

Repeat while predictions for each observation in test data set are made
Fit the Linear Regression model
Make predictions. Save the 1st one
Append training data with first observation from test data set
Delete the first observation from training and test dataset

Figure 5 - Stacking model methodology

3.5 Error Assessment

To evaluate our forecasting, we will be using two main statistical criteria:

- Root Mean Square Error (RMSE)
- Mean Absolute Percentage Error (MAPE)

RMSE and MAPE can be described by the following equations respectively:

$$RMSE = \sqrt{\sum_{t=1}^{T} \frac{(P_t - Z_t)^2}{T}}$$

$$MAPE = \frac{1}{T}\sum_{t=1}^{T} \frac{P_t - Z_t}{Z_t}$$

where P_t is the predicted value at time t, Z_t is the actual value at time t and T is the number of predictions.

In addition to the statistical evaluation methods, in stock markets, it is also essential to understand how good the model is in predicting the turning points or if the model can make the right decision based on predicted price changes. To predict the turning points correctly the following has to be met:

$$sign\left(P_t - P_{t-n_{future}}\right) = sign(Z_t - Z_{t-n_{future}})$$

Hence we are going to calculate the Turning Point Accuracy (TPA) with the following formula:

$$TPA = \frac{1}{T}\sum_{t=1}^{T} A_t \Leftrightarrow t$$

where

$$\begin{cases} A_t = 1\ if\ \Delta Z_t > 0 \\ A_t = 0\ if\ \Delta Z_t \leq 0 \\ F_t = 1\ if\ \Delta P_t > 0 \\ F_t = 0\ if\ \Delta P_t \leq 0 \end{cases}$$

Moreover, we are going to implement a method proposed by (Chen et al., 2021) which evaluates the turning point prediction based on calculated “Buy” or “Sell” signal. We will adapt the method to our use case and define the signal for the actual prices and the forecasted value using the following method:

1. Calculate the difference of the actual (A_t) and forecasted (F_t) stock price:

$$F_t = P_{t+n_{future}} - Z_t$$

$$A_t = Z_{t+n_{future}} - Z_t$$

2. Calculate the change rates:

$$Fr_t = \frac{F_t}{Z_T}$$

$$Ar_t = \frac{A_t}{Z_T}$$

3. If the change rate is greater than θ the time point $t + n_{future}$ is signed as “Buy” signal. If the change rate is less than $-\theta$ the time point T is signed as “Sell” signal. For the change rates between θ and $-\theta$ we are going to assign “Hold” signal.
4. If the signal for Ar_t is equal to the signal for Fr_t, we are going to count it as correct classification and sum all of them.
5. Calculate the Action Prediction Accuracy (APA) of the correct classifications by dividing the sum of correct classification by T.

To test the APA we chose to use the 5% threshold.

4 Experimental settings

4.1 Data sets

S&P 500

The shape of the data before processing is (1357, 5), which means that we have 1357 observations and five features in each observation. No missing values were detected in the data set. We removed the ‘Date’ feature from the further analysis, as we will be using only Open High, Low and Close prices for the predictions. Main statistics for the data (rounded till 2 digits after comma):

	Open	High	Low	Close
Mean	3211.64	3228.71	3192.62	3211.76
Standard deviation	719.67	724.38	714.27	719.45
Min	2251.57	2263.88	2191.86	2237.4
Max	4804.51	4818.62	4780.04	4796.56

Table 1 – S&P 500 main statistics before processing

After the standardization, the mean and standard deviation became 0 and 1, respectively, and min and max values as follows (rounded till two digits after the comma):

	Open	High	Low	Close
Min	-1.33	-1.33	-1.4	-1.35
Max	2.21	2.19	2.22	2.2

Table 2 – S&P 500 min and max values after processing

After splitting the data set for ARIMA model the test data set contains 1085 observations and test data set contains 272 observations.

The shapes of data for LSTM models after splitting and reshaping the data as well as the data set sizes for stacking model can be seen in Table 8 in Annex 2.

BTC

The shape of the data before processing is (1969, 7), which means that we have 1969 observations and 7 features in each observation. Two additional features in this data set are ‘Adj. Close’ and ‘Volume’. No missing values were detected in the data set. We removed the ‘Date’, ‘Adj. Close’ and

‘Volume’ features from further analysis, as we will be using only Open High, Low and Close prices for the predictions. Main statistics for the data (rounded till 2 digits after comma):

	Open	High	Low	Close
Mean	17261.92	17716.21	16757.33	17273.93
Standard deviation	17586.23	18040.13	17057.70	17580.01
Min	775.18	67549.73	755.76	777.76
Max	67549.73	68789.62	66382.06	67566.83

Table 3 – BTC main statistics before processing

After the standardization the mean and standard deviation became 0 and 1 respectively, and min and max values as follows (rounded till 2 digits after comma):

	Open	High	Low	Close
Min	-0.94	-0.94	-0.94	-0.94
Max	2.83	2.83	2.91	2.86

Table 4 – BTC min and max values after processing

After splitting the data set for ARIMA model the test data set contains 1575 observations and test data set contains 394 observations.
The shapes of data for LSTM models after splitting and reshaping the data as well as the data set sizes for stacking model can be seen in Table 9 in Annex 2.

4.2 ARIMA

First we are going to define d – the number of differences. For this we will first check the stationary of the time series with Augmented Dickey Fuller test.
For S&P 500 Open price data series $p-value = 0.713$, which is way above the significance level 0.05 (5%). After differencing the time series 1 time $p\ -value = 0.00$, hence we defined $d = 1$. For Close price data series $p-value = 0.708$, which is also way above the significance level 0.05 (5%). After differencing the time series 1 time $p\ -value = 0.00$, hence we defined $d = 1$. For BTC Open price data series $p-value = 0.575$, which is above the significance level. After differencing the time series 1 time $p\ -value = 0.00$, hence we defined $d = 1$. For BTC Close price data series $p-value = 0.585$, which is also above the significance level. After differencing the time series 1 time $p\ -value = 0.00$, hence we defined $d = 1$.
For *auto_arima* function we specified these parameters:

$$auto\_arima\ =\ (\ train\_data, test = "\ adf", d = 1, trace = True)$$

Models chosen as the best ones by this function are as follows:

Obs. back	**BTC**		**S&P 500**	
	Open	Close	Open	Close
30	(0, 1, 0)	(0, 1, 0)	(0, 1, 0)	(0, 1, 0)
60	(0, 1, 0)	(0, 1, 0)	(0, 1, 0)	(2, 1, 2)
90	(0, 1, 0)	(0, 1, 0)	(0, 1, 0)	(2, 1, 2)

Table 5 - Best models from *auto_arima* function

From partial autocorrelation and autocorrelation graphs for S&P 500 Open and BTC Open and Close price series in Annexes

Annex 1 we see that after 0, the first point to stand above the significance line in all graphs is 9. Hence for these time series the range for $p = [0,9]$ and the range for $q = [0,9]$. From S&P 500 Close price data series partial autocorrelation graphs in Annexes

Annex 1 we see that after 0, the first point to stand above the significance line is 2, hence the range for $p = [0,2]$. Based on autocorrelation plot we can see that the first point to stand above the significance line is 9, hence the range for $q = [0,9]$.

After performing the grid search with specified parameter ranges for applicable models we define that the parameters for all models are (0, 1, 0) except for one - for BTC Close time series model looking back 60 observations and predicting 14 observations forward ARIMA parameters (3,1,3) demonstrated the best performance than comparing RMSE.

We are going to use the parameters defined by the grid search.

4.3 LSTM

The models we selected can be seen bellow:

Obs. ahead	**Data set**	**Observations to look back**		
		30	60	90
1	S&P 500 Open	SP_48	SP_48	SP_64
	BTC Open	BTC_48	BTC_64	BTC_64
	S&P 500 Close	SP_64_32	SP_64	SP_64
	BTC Close	BTC_64	BTC_48	BTC_64
7	S&P 500 Open	SP_48	SP_32_16	SP_48
	BTC Open	BTC_ 32	BTC_48	BTC_48
	S&P 500 Close	SP_64	SP_32_16	SP_64
	BTC Close	BTC_64	BTC_ 64	BTC_32
14	S&P 500 Open	SP_48	SP_ 32	SP_32_16
	BTC Open	BTC_64	BTC_ 64	BTC_64_32
	S&P 500 Close	SP_64_32	SP_32_16	SP_32
	BTC Close	BTC_48	BTC_48	BTC_32

Table 6 – Best LSTM models for S&P 500 and BTC data sets

5 Results

Table 7 presents the results for standalone and stacked models on Open and Close price time-series predictions.

Obs. back/ahead, Model			**Open**				**Model**	**Close**			
			RMSE	MAPE	TPA	APA		RMSE	MAPE	TPA	APA
30/1	S&P 500	ARIMA	0.064	0.029	0.491	0.819		0.090	**0.043**	0.509	0.697
		SP_48	0.035	0.178	0.814	**0.880**	SP_64_32	**0.085**	0.182	**0.517**	**0.727**
		Stacked	**0.033**	**0.020**	**0.898**	0.857		0.104	0.062	0.510	0.571
	BTC	ARIMA	0.039	0.179	0.509	0.880		0.027	0.117	0.509	0.868
		BTC_48	0.032	1.150	0.462	0.904	BTC_64	0.017	1.263	0.692	0.937

		Stacked	**0.007**	**0.007**	**0.685**	**1.000**		**0.002**	**0.002**	**0.781**	**1.000**
60/1	S&P 500	ARIMA	**0.025**	**0.012**	0.867	**0.893**		**0.067**	**0.032**	0.483	**0.804**
		SP_48	0.030	0.175	0.868	0.892	SP_64	0.085	0.184	**0.538**	0.783
		Stacked	0.032	0.020	**0.953**	0.814		0.104	0.060	0.488	0.581
	BTC	ARIMA	0.039	0.179	**0.519**	0.870		0.026	0.116	0.537	0.865
		BTC_64	0.014	0.173	0.455	0.961	BTC_48	0.006	0.174	**0.793**	0.976
		Stacked	**0.003**	**0.002**	0.448	**1.000**		**0.001**	**0.001**	0.791	**1.000**
90/1	S&P 500	ARIMA	0.025	**0.011**	0.860	**0.904**		**0.067**	**0.032**	0.498	**0.804**
		SP_64	0.034	0.191	0.835	0.879	SP_64	0.152	0.215	**0.500**	0.291
		Stacked	0.034	0.021	**0.946**	0.757		0.110	0.065	0.459	0.568
	BTC	ARIMA	0.039	0.179	**0.517**	0.870		0.026	0.116	0.483	0.863
		BTC_64	0.013	0.115	0.378	0.993	BTC_64	0.006	0.117	**0.674**	0.993
		Stacked	**0.002**	**0.002**	0.475	**1.000**		**0.002**	**0.002**	0.361	**1.000**
30/7	S&P 500	ARIMA	**0.150**	**0.072**	**0.587**	**0.483**		**0.159**	**0.077**	0.475	**0.392**
		SP_48	0.452	0.255	0.441	0.281	SP_64	0.205	0.167	0.483	0.277
		Stacked	0.219	0.137	0.583	0.352		0.237	0.152	**0.521**	0.278
	BTC	ARIMA	0.073	0.278	0.501	0.788		0.070	0.282	0.549	0.742
		BTC_32	0.071	1.041	0.358	0.775	BTC_64	0.068	1.143	0.394	0.835
		Stacked	**0.006**	**0.005**	**0.528**	**0.923**		**0.007**	**0.005**	**0.569**	**0.923**
60/7	S&P 500	ARIMA	**0.149**	**0.072**	**0.587**	**0.487**		**0.159**	**0.077**	0.471	**0.392**
		SP_32_16	0.320	0.204	0.330	0.297	SP_32_16	0.585	0.292	**0.495**	0.335
		Stacked	0.221	0.139	0.500	0.354		0.240	0.156	0.381	0.188
	BTC	ARIMA	0.073	0.276	0.504	0.780		0.069	0.282	**0.567**	0.736
		BTC_48	0.048	0.159	0.369	0.793	BTC_64	0.047	0.161	0.345	0.841
		Stacked	**0.005**	**0.005**	**0.515**	**0.917**		**0.005**	**0.005**	0.561	**0.917**
90/7	S&P 500	ARIMA	**0.149**	**0.072**	**0.587**	**0.494**		**0.159**	**0.077**	0.510	0.392
		SP_48	0.258	0.170	0.511	0.379	SP_64	1.100	0.604	**0.557**	**0.396**
		Stacked	0.228	0.148	0.500	0.310		0.243	0.161	0.250	0.190
	BTC	ARIMA	0.073	0.276	**0.491**	0.780		0.069	0.282	0.522	0.731
		BTC_48	0.035	0.112	0.352	0.901	BTC_32	0.036	0.116	0.339	0.885
		Stacked	**0.006**	**0.005**	0.467	**0.909**		**0.005**	**0.005**	**0.817**	**0.909**
30/14	S&P 500	ARIMA	**0.217**	**0.106**	0.584	0.310		**0.227**	**0.112**	0.461	0.256
		SP_48	2.411	1.326	0.546	**0.368**	SP_64_32	4.188	1.888	0.410	**0.277**
		Stacked	0.342	0.223	**0.630**	0.271		0.363	0.243	**0.500**	0.237
	BTC	ARIMA	0.111	0.405	0.510	0.697		0.107	0.432	**0.553**	0.653
		BTC_64	0.098	0.818	0.305	0.124	BTC_48	0.100	0.948	0.288	0.124
		Stacked	**0.007**	**0.006**	**0.775**	**0.845**		**0.009**	**0.009**	0.268	**0.845**
60/14	S&P 500	ARIMA	**0.216**	**0.106**	0.576	0.326		**0.225**	**0.111**	0.473	0.256
		SP_32	0.434	0.208	**0.523**	**0.340**	SP_32_16	3.120	1.374	0.442	**0.325**
		Stacked	0.383	0.263	0.375	0.189		0.387	0.264	**0.575**	0.208
	BTC	ARIMA	0.114	0.402	0.501	0.700		0.111	0.430	**0.471**	0.663
		BTC_64	0.073	0.155	0.280	0.060	BTC_48	0.079	0.158	0.259	0.048
		Stacked	**0.007**	**0.006**	**0.692**	**0.833**		**0.008**	**0.007**	0.215	**0.833**
90/14	S&P 500	ARIMA	**0.216**	**0.106**	**0.584**	0.326		**0.225**	**0.111**	0.510	0.256
		SP_32_16	0.835	0.438	0.568	**0.412**	SP_32	1.079	0.586	**0.550**	**0.434**
		Stacked	0.372	0.262	0.529	0.170		0.418	0.304	0.206	0.064
	BTC	ARIMA	0.110	0.403	**0.488**	0.697		0.107	0.431	**0.559**	0.642
		BTC_64_32	0.051	0.125	0.447	0.714	BTC_32	0.042	0.111	0.254	0.783
		Stacked	**0.007**	**0.007**	0.271	**0.819**		**0.007**	**0.006**	0.525	**0.819**

Table 7 - Model comparison for Open and Close price predictions (best values in bold)

First of all we can notice, that considering statistical model evaluation criteria, RMSE and MAPE, and turning point prediction evaluation criteria APA all stacked models on both, Open and Close, time series of BTC prices performed better than standalone models. Considering TPA – 8 out of 18 times it was also better for the stacked model than standalone. 8 out of 10 times then it underperformed the standalone model comparing TPA it was ARIMA model that had the best performance. Although overall the performance in predicting the turning point was low for BTC time series – it went above 80% accuracy only once while predicting BTC Close prices 7 observations ahead with 90 previous observations. It is worth mentioning, that in this case standalone models predicted the turning point correctly only 52% (ARIMA) and 34% (BTC_48). Also, the best model chosen for BTC LSTM models was always with 1 hidden layer, except when making the model looking back 90 observations and predicting 14 observations ahead the deepest and widest LSTM was chosen – 2 hidden layers with 64 and 32 neurons.

Considering the stacked models' behavior on S&P 500 data set – 11 out 18 times it underperformed in all criteria compared to standalone models. The worst performance of stacked models was observed while predicting Close prices. In fact, in these predictions it outperformed standalone models only 3 times out of 9 and only in one evaluation criteria – Turning Point Accuracy, and it was not more than 58%. 6 times out of 18 it underperformed standalone models in 3 criteria and the one measure that it performed better at was TPA. Though worthy to mention, that 2 times the turning point prediction accuracy was at least 90% – while predicting the Open prices 1 observation ahead looking back 30, 60 and 90 observations. For all of the other models TPA was mainly around 50%. Overall the performance of LSTM networks decreased in Close price predictions than we gave them more observations to look back at in case on 1 and 7 observations ahead prediction. In the prediction of 14 observations ahead LSTM models performance improved with the bigger amount of history data. While it was not the case with Open price predictions. In both, Open and Close price predictions, ARIMA was the dominant best performing model.

6 Conclusions

In this study we test the usage of stacking ensemble learning method with statistical model ARIMA and RNN LSTM in forecasting the prices for the index S&P500 and cryptocurrency Bitcoin. The ensemble models are composed of the best performing ARIMA and LSTM models and their performance is check against the standalone models. We evaluated classification and regression performance of standalone and stacked models.

It was found that combining ARIMA method with LSTM by stacking (using linear regression as meta learner) does show better performance in predicting price for 1, 7 and 14 observations ahead on Bitcoin crypto currency data set. Although for turning point prediction accuracy it shares the same success rate as ARIMA models. LSTM models on BTC dataset outperformed the Stacked and ARIMA only 2 times out of 18 in turning point prediction accuracy. Although the highest reached accuracy for BTC data set was 79% (by LSTM model). On the other hand, when we look at the models' performance on S&P 600 prices data set, stacked model most of the times underperformed the individual models. Despite that, the best performance of the stacked model was observed on Open price predictions 1 day ahead and the turning point accuracy achieved by this model was at least 90%.

For the future work it would be worthy to study the importance of other factors than Open, High, Low and Close prices of the stock – such as Volume (as it identifies the total number of shares bought/sold), sentiment analysis and level of the earnings base (e. g. Price-to-Earnings ratio). Another area worthy of interest is the study of the benefit of these forecast models with metrics that test the performance based on earnings. Lastly, the ability of the models to identify the outliers in the stock price time series could be tested in order to identify the market instability and adapt decisions based on that.

References

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, *19*(6). https://doi.org/10.1109/TAC.1974.1100705

Akaike, H. (1979). A Bayesian extension of the minimum aic procedure of autoregressive model fitting. *Biometrika*, *66*(2). https://doi.org/10.1093/biomet/66.2.237

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, *5*(2). https://doi.org/10.1109/72.279181

Box, G., & Jenkins, G. (1970). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Brockwell, P. J., & Davis, R. A. (2016). *Introduction to Time Series and Forecasting*. Springer International Publishing. https://doi.org/10.1007/978-3-319-29854-2

Brownlee, J. (2016). *Deep Learning With Python Develop Deep Learning Models On Theano And TensorFlow Using Keras i Deep Learning With Python*. Machine Learning Mastery.

Caire, P., Hatabian, G., & Muller, C. (1992). Progress in Forecasting by Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, *2*. https://doi.org/10.1109/IJCNN.1992.226932

Chen, J., Yang, S., Zhang, D., & Nanehkaran, Y. A. (2021). A turning point prediction method of stock price based on RVFL-GMDH and chaotic time series analysis. *Knowledge and Information Systems*, *63*(10). https://doi.org/10.1007/s10115-021-01602-3

Christiansen, B. (2018). Ensemble averaging and the curse of dimensionality. *Journal of Climate*, *31*(4). https://doi.org/10.1175/JCLI-D-17-0197.1

Deng, Y., Fan, H., & Wu, S. (2020). A hybrid ARIMA-LSTM model optimized by BP in the forecast of outpatient visits. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-020-02602-x

Fathi, O. (2019). Time series forecasting using a hybrid ARIMA and LSTM model. *Velvet Consulting*.

Guthery, F. S., Burnham, K. P., & Anderson, D. R. (2003). Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach. *The Journal of Wildlife Management*, *67*(3). https://doi.org/10.2307/3802723

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8). https://doi.org/10.1162/neco.1997.9.8.1735

Hu, M. J. C., & Root, H. E. (1964). An Adaptive Data Processing System for Weather Forecasting. *Journal of Applied Meteorology*, *3*(5). https://doi.org/10.1175/1520-0450(1964)003<0513:aadpsf>2.0.co;2

Huber, C., Huber, J., & Hueber, L. (2019). The effect of experts' and laypeople's forecasts on others' stock market forecasts. *Journal of Banking and Finance*, *109*. https://doi.org/10.1016/j.jbankfin.2019.105662

Kolarik, T., & Rudorfer, G. (1994). Time series forecasting using neural networks. *Proceedings of the International Conference on APL : The Language and Its Applications, APL 1994*. https://doi.org/10.1145/190271.190290

Liu, P. (2022). Time Series Forecasting Based on ARIMA and LSTM. *Proceedings of the 2022 2nd International Conference on Enterprise Management and Economic Development (ICEMED 2022)*, 1203–1208. https://doi.org/10.2991/aebmr.k.220603.195

Livieris, I. E., Pintelas, E., Stavroyiannis, S., & Pintelas, P. (2020). Ensemble Deep learning models for forecasting cryptocurrency time-series. *Algorithms*, *13*(5). https://doi.org/10.3390/A13050121

Ma, Q. (2020). Comparison of ARIMA, ANN and LSTM for Stock Price Prediction. *E3S Web of Conferences*, *218*. https://doi.org/10.1051/e3sconf/202021801026

Mahadik, A., Vaghela, D., & Mhaisgawali, A. (2021). Stock Price Prediction using LSTM and ARIMA. *Proceedings of the 2nd International Conference on Electronics and Sustainable Communication Systems, ICESC 2021*. https://doi.org/10.1109/ICESC51422.2021.9532655

Mauldin, T., Ngu, A. H., Metsis, V., & Canby, M. E. (2021). Ensemble Deep Learning on Wearables Using Small Datasets. *ACM Transactions on Computing for Healthcare*, *2*(1). https://doi.org/10.1145/3428666

Pintelas, E., Livieris, I., Stavroyiannis, S., Kotsilieris, T., & Pintelas, P. (2020). *Fundamental Research Questions and Proposals on Predicting Cryptocurrency Prices using DNNs*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In: Rumelhart D E, McClelland J L et al. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition. *MIT Press, Cambridge, MA*, *1*(V).

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, *7*(1), 66. https://doi.org/10.1186/s40537-020-00333-6

Siami-Namini, S., & Namin, A. S. (2018). *Forecasting Economics and Financial Time Series: ARIMA vs. LSTM*.

Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2019). A Comparison of ARIMA and LSTM in Forecasting Time Series. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*. https://doi.org/10.1109/ICMLA.2018.00227

Stellwagen, E., & Tashman, L. (2013). ARIMA: The Models of Box and Jenkins. *Foresight: The International Journal of Applied Forecasting*, *30*.

Tang, L., Yi, Y., & Peng, Y. (2019). An ensemble deep learning model for short-term load forecasting based on ARIMA and LSTM. *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2019*. https://doi.org/10.1109/SmartGridComm.2019.8909756

Tseng, F. M., Yu, H. C., & Tzeng, G. H. (2002). Combining neural network model with seasonal time series ARIMA model. *Technological Forecasting and Social Change*, *69*(1). https://doi.org/10.1016/S0040-1625(00)00113-X

Verma, S., Prakash Sahu, S., & Prasad Sahu, T. (2022). *Ensemble Approach for Stock Market Forecasting Using ARIMA and LSTM Model*. https://doi.org/10.1007/978-981-16-7330-6_6

Wang, Z., Qu, J., Fang, X., Li, H., Zhong, T., & Ren, H. (2020). Prediction of early stabilization time of electrolytic capacitor based on ARIMA-Bi_LSTM hybrid model. *Neurocomputing*, *403*. https://doi.org/10.1016/j.neucom.2020.03.054

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1*(4). https://doi.org/10.1016/0893-6080(88)90007-X

Whillans, A. (2020). *Time Smart: How to Reclaim Your Time and Live a Happier Life*. Harvard Business Review Press.

Whillans, A. V. (2019). Time for Happiness: Why the Pursuit of Money Isn't Bringing You Joy—and What Will. *Harvard Business Review (Website) , Special Issue on HBR Big Idea: Time Poor and Unhappy*.

Wolpert, D. H. (1992). Original Contribution: Stacked Generalization. *Neural Netw.*, *5*(2).

Xiao, D., & Su, J. (2022). Research on Stock Price Time Series Prediction Based on Deep Learning and Autoregressive Integrated Moving Average. *Scientific Programming*, *2022*, 1–12. https://doi.org/10.1155/2022/4758698

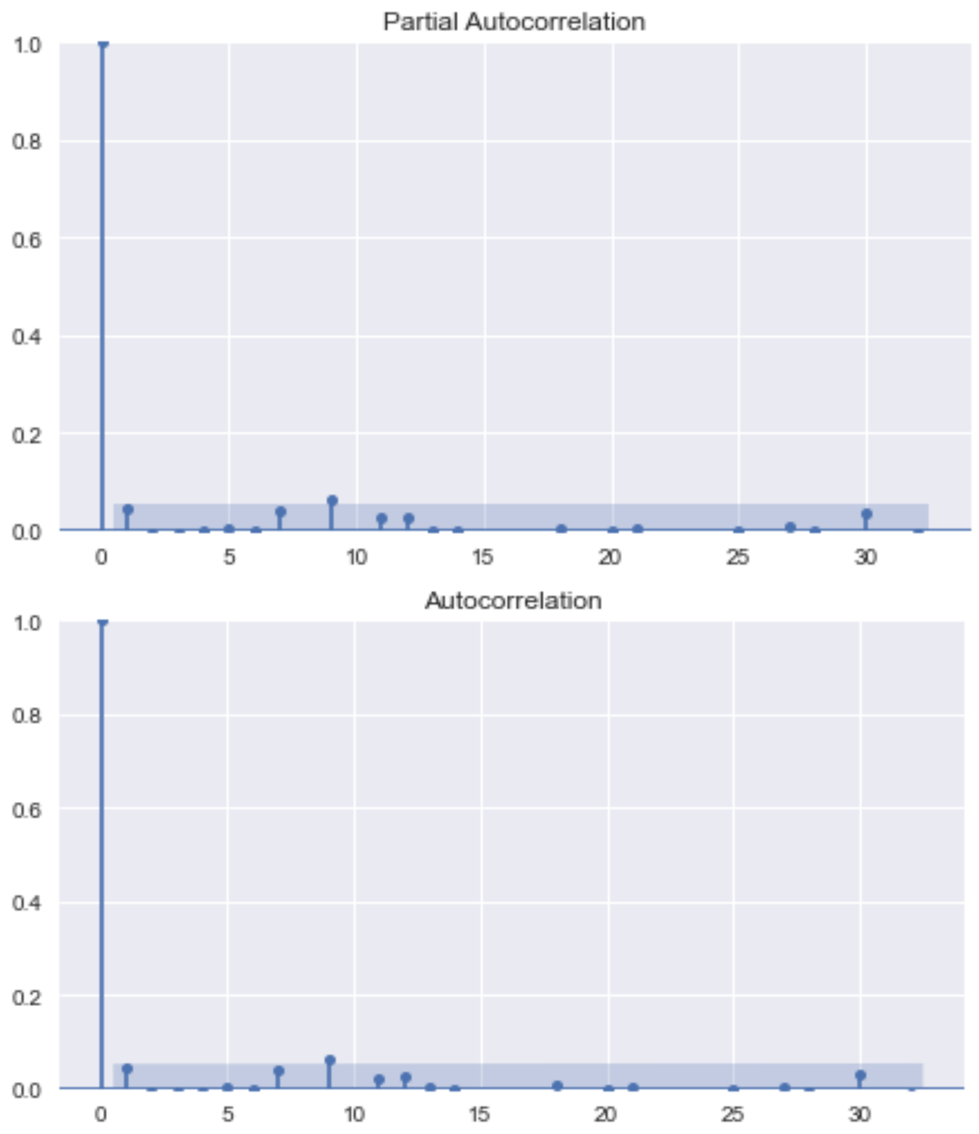
Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, *14*(1). https://doi.org/10.1016/S0169-2070(97)00044-7

Zhang, P. G. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*. https://doi.org/10.1016/S0925-2312(01)00702-0

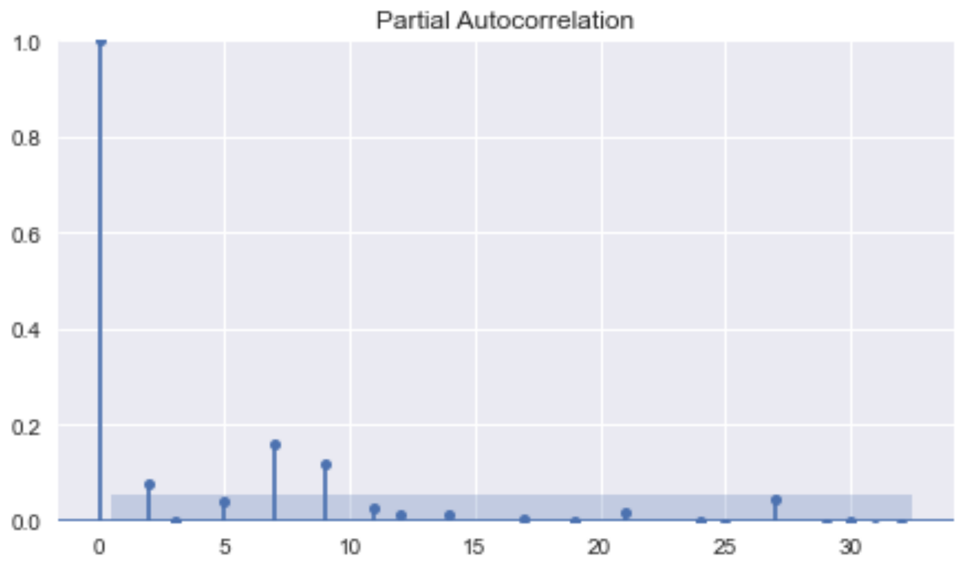
Annexes

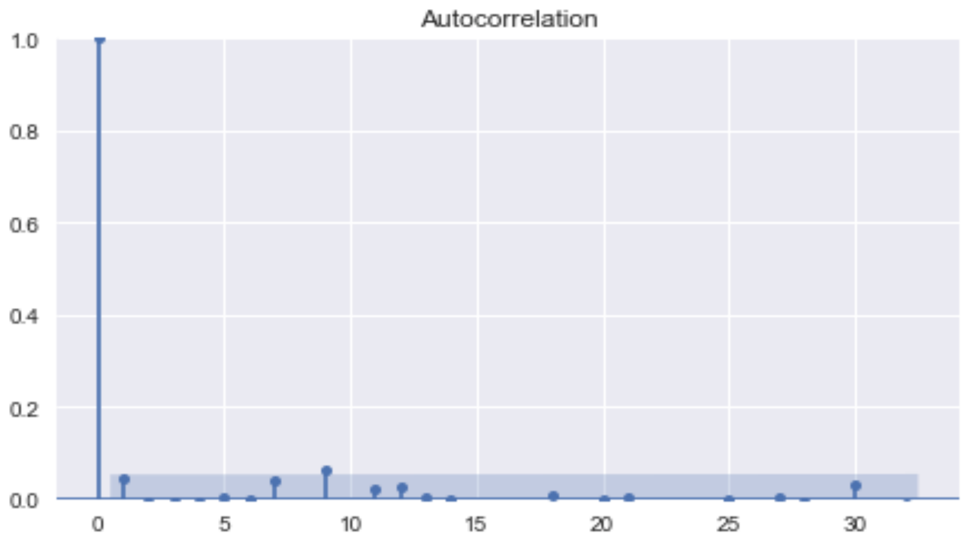
Annex 1

S&P 500 Open price time series

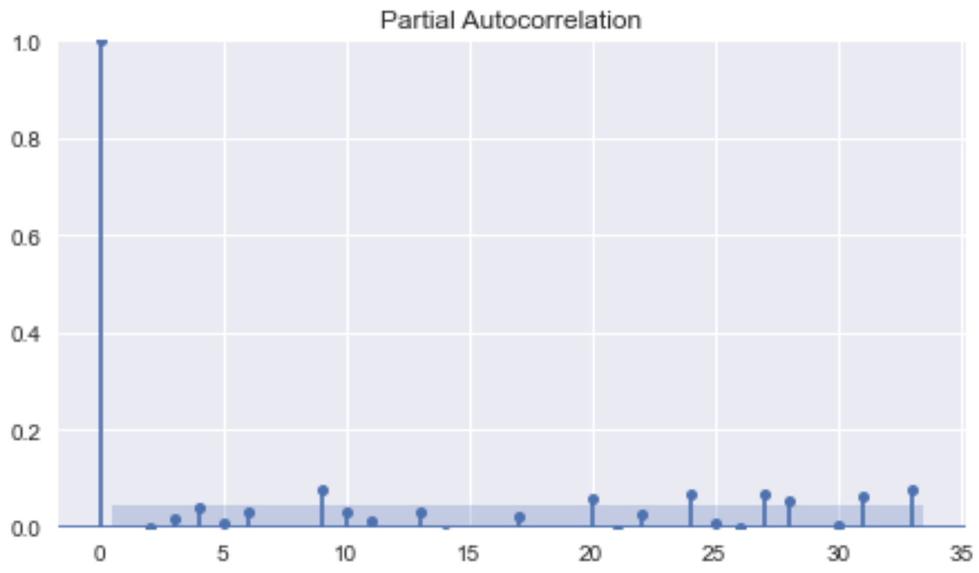


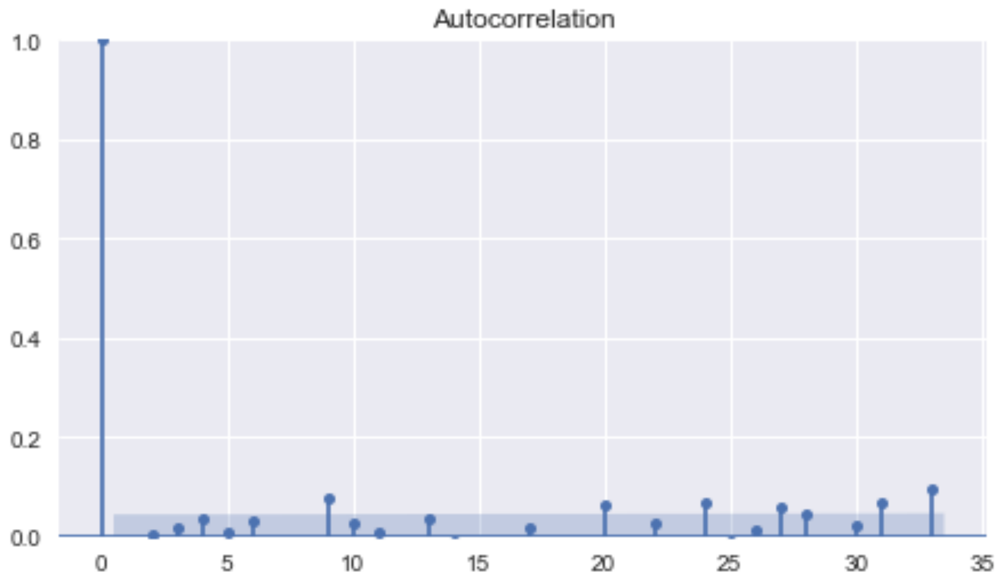
S&P 500 Close price time series



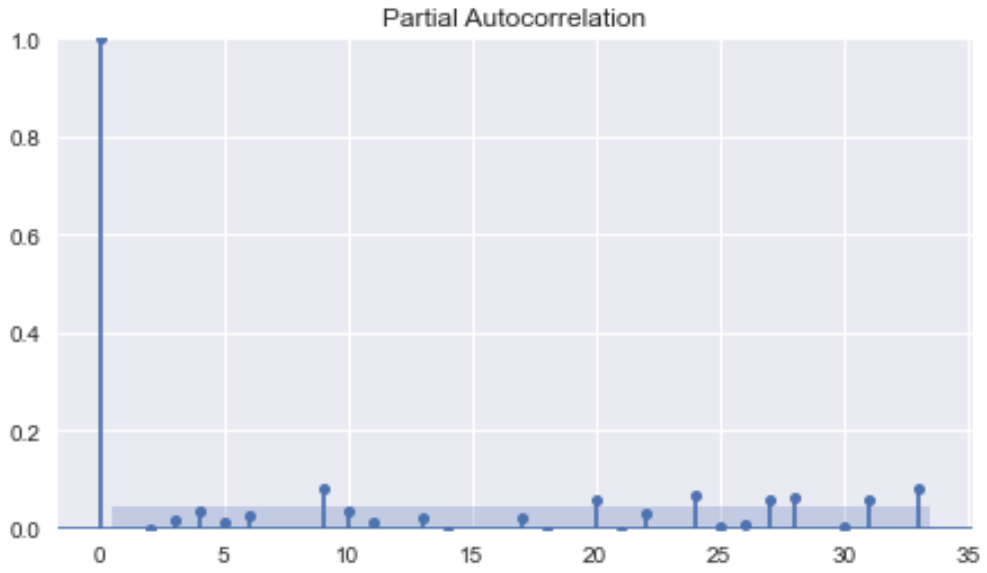


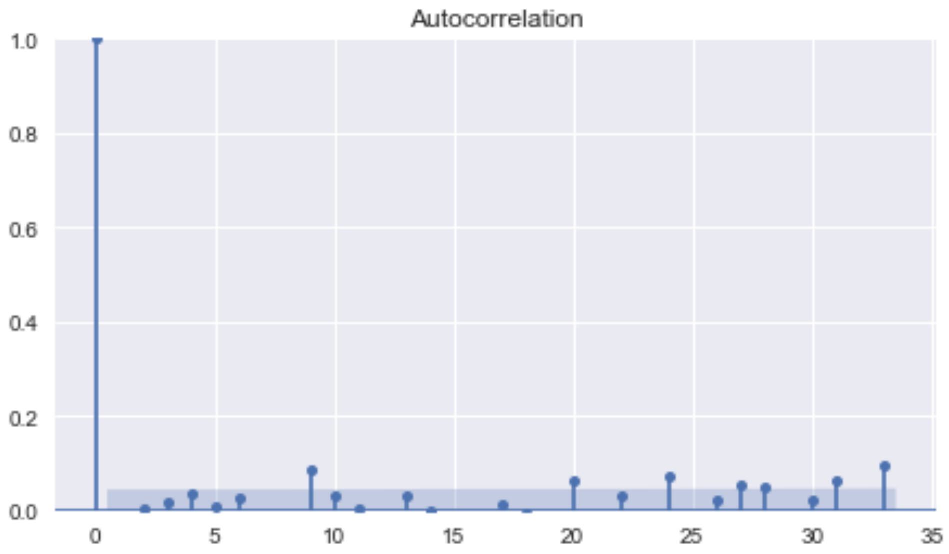
BTC Open price time series





BTC 500 Close price time series





Annex 2

Obs. ahead	Observations to look back to the future		
	30	60	90
1	X for train: (919, 30, 4), Y for train: (919, 1), X for validation: (106, 30, 4), Y for validation: (106, 1), X for test: (242, 30, 4), Y for test: (242, 1). Stacking data set size: 242, Train size: 194, Test size: 48	X for train: (889, 60, 4), Y for train: (889, 1), X for validation: (76, 60, 4), Y for validation: (76, 1), X for test: (212, 60, 4), Y for test: (212, 1). Stacking data set size: 212, Train size: 170, Test size: 42	X for train: (859, 90, 4), Y for train: (859, 1), X for validation: (46, 90, 4), Y for validation: (46, 1), X for test: (182, 90, 4), Y for test: (182, 1). Stacking data set size: 182, Train size: 146, Test size: 36
7	X for train: (913, 30, 4), Y for train: (913, 1), X for validation: (100, 30, 4), Y for validation: (100, 1), X for test: (236, 30, 4), Y for test: (236, 1). Stacking data set size: 236, Train size: 189, Test size: 47	X for train: (883, 60, 4), Y for train: (883, 1), X for validation: (70, 60, 4), Y for validation: (70, 1), X for test: (206, 60, 4), Y for test: (206, 1). Stacking data set size: 206, Train size: 165, Test size: 41	X for train: (853, 90, 4), Y for train: (853, 1), X for validation: (40, 90, 4), Y for validation: (40, 1), X for test: (176, 90, 4), Y for test: (176, 1). Stacking data set size: 176, Train size: 141, Test size: 35
14	X for train: (906, 30, 4), Y for train: (906, 1), X for validation: (93, 30, 4), Y for validation: (93, 1), X for test: (229, 30, 4), Y for test: (229, 1). Stacking data set size: 229, Train size: 183, Test size: 46	X for train: (876, 60, 4), Y for train: (876, 1), X for validation: (63, 60, 4), Y for validation: (63, 1), X for test: (199, 60, 4), Y for test: (199, 1). Stacking data set size: 199, Train size: 159, Test size: 40	X for train: (846, 90, 4), Y for train: (846, 1), X for validation: (33, 90, 4), Y for validation: (33, 1), X for test: (169, 90, 4), Y for test: (169, 1). Stacking data set size: 169, Train size: 135, Test size: 34

Table 8 – S&P data shapes after transformation

Obs. Ahead	Observations to look back to the future		
	30	60	90
1	X for train: (1348, 30, 4), Y for train: (1348, 1), X for validation: (167, 30, 4), Y for validation: (167, 1), X for test: (364, 30, 4), Y for test: (364, 1).	X for train: (1318, 60, 4), Y for train: (1318, 1), X for validation: (137, 60, 4), Y for validation: (137, 1), X for test: (334, 60, 4), Y for test: (334, 1).	X for train: (1288, 90, 4), Y for train: (1288, 1), X for validation: (107, 90, 4), Y for validation: (107, 1), X for test: (304, 90, 4), Y for test: (304, 1).

	Stacking data set size: 364, Train size: 291, Test size: 73.	Stacking data set size: 334, Train size: 267, Test size: 67.	Stacking data set size: 304, Train size: 243, Test size: 61.
7	X for train: (1342, 30, 4), Y for train: (1342, 1), X for validation: (161, 30, 4), Y for validation: (161, 1), X for test: (358, 30, 4), Y for test: (358, 1). Stacking data set size: 358, Train size: 286, Test size: 72.	X for train: (1312, 60, 4), Y for train: (1312, 1), X for validation: (131, 60, 4), Y for validation: (131, 1), X for test: (328, 60, 4), Y for test: (328, 1). Stacking data set size: 328, Train size: 262, Test size: 66.	X for train: (1282, 90, 4), Y for train: (1282, 1), X for validation: (101, 90, 4), Y for validation: (101, 1), X for test: (298, 90, 4), Y for test: (298, 1). Stacking data set size: 298, Train size: 238, Test size: 60.
14	X for train: (1335, 30, 4), Y for train: (1335, 1), X for validation: (154, 30, 4), Y for validation: (154, 1), X for test: (351, 30, 4), Y for test: (351, 1). Stacking data set size: 351, Train size: 281, Test size: 70.	X for train: (1305, 60, 4), Y for train: (1305, 1), X for validation: (124, 60, 4), Y for validation: (124, 1), X for test: (321, 60, 4), Y for test: (321, 1). Stacking data set size: 321, Train size: 257, Test size: 64.	X for train: (1275, 90, 4), Y for train: (1275, 1), X for validation: (94, 90, 4), Y for validation: (94, 1), X for test: (291, 90, 4), Y for test: (291, 1). Stacking data set size: 291, Train size: 233, Test size: 58.

Table 9 – BTC data shapes after transformation