Berger, Theo; Koubová, Jana

Forecasting Bitcoin returns: Econometric time series analysis vs. machine learning

Journal of Forecasting

Provided in Cooperation with:
John Wiley & Sons

Suggested Citation: Berger, Theo; Koubová, Jana (2024) : Forecasting Bitcoin returns: Econometric time series analysis vs. machine learning, Journal of Forecasting, ISSN 1099-131X, Wiley, Hoboken, NJ, Vol. 43, Iss. 7, pp. 2904-2916, https://doi.org/10.1002/for.3165

Received: 5 June 2023 | Revised: 28 March 2024 | Accepted: 19 May 2024

DOI: 10.1002/for.3165

RESEARCH ARTICLE

WILEY

Forecasting Bitcoin returns: Econometric time series analysis vs. machine learning

Theo Berger[1,2] | **Jana Koubová**[3]

[1]Department of Business and Computer Science, University of Applied Sciences Hannover, Hanover, Germany

[2]Department of Business and Administration, University of Bremen, Bremen, Germany

[3]Sparkassen Rating und Risikosysteme, Berlin, Germany

Correspondence
Theo Berger, Department of Business and Computer Science, University of Applied Sciences Hannover, Ricklinger Stadtweg 120, D-30459 Hannover, Germany.
Email: theo.berger@hs-hannover.de

Abstract

We study the statistical properties of the Bitcoin return series and provide a thorough forecasting exercise. Also, we calibrate state-of-the-art machine learning techniques and compare the results with econometric time series models. The empirical assessment provides evidence that the application of machine learning techniques outperforms econometric benchmarks in terms of forecasting precision for both in- and out-of-sample forecasts. We find that both deep learning architectures as well as complex layers, such as LSTM, do not increase the precision of daily forecasts. Specifically, a simple recurrent neural network describes a sensible choice for forecasting daily return series.

KEYWORDS

forecasting, machine learning, risk measurement, time series analysis

JEL CLASSIFICATION

C33, C58, G17, G23

1 | INTRODUCTION

Finding the adequate methodological approach to forecast individual financial return series describes a staggering task. In order to achieve precise out-of-sample forecasts, it is necessary to understand the properties of the underlying time series. In this vein, the statistical properties of financial time series have been widely discussed and the application of Autoregressive Moving Average (ARMA) Models is widely accepted (Berger & Gencay, 2018; Halbleib & Pohlmeier, 2012).

Due to steadily growing computational power, as well as increasing data availability, forecasting economic time series via machine learning techniques describes a novel string of research. As discussed by Kraus et al. (2020), machine learning is less restrictive regarding the assumptions on the underlying data and current machine learning approaches can adjust to properties of economic time series individually. Therefore machine learning describes a fruitful alternative to econometric modelling. Although machine learning techniques are characterized as black boxes, recent studies provide empirical evidence that machine learning achieves higher forecasting precision than widely accepted econometric approaches. Gu et al. (2020) provide a thorough empirical assessment and discuss competing machine-learning techniques applied to economic data sets. As a result, adequately calibrated machine learning approaches outperform interpretable econometric models in terms of forecasting accuracy. Feng et al. (2020) confirm these results for stock returns, Longo et al. (2022) for GDP forecasts, and Makridakis et al. (2018) for various economic data sets.

This study adds to this string of literature, and we focus on forecasts for innovative economic return series, namely Bitcoin returns. As described in Alessandretti et al. (2018) and Tandon et al. (2019), Bitcoins belong to the asset class of cryptocurrencies and are characterized by higher volatility than classical currencies. Furthermore, as cryptocurrencies are not controlled by national central banks, Bitcoins are exposed to different economic determinants and hence historic return series can exhibit different time series characteristics. Alessandretti et al. (2018) provide empirical evidence that the application of neural networks to time series predictions is a fruitful approach, especially when it comes to predicting Bitcoins. Also, Lahmiri and Bekiros (2019) provide empirical evidence that forecasting Bitcoin return series via machine learning outperforms typical econometric benchmarks, such as ARMA models. Typically, recent studies assess forecasting performance via MAE (Mean Absolute Error) or RMSE (Root Mean Squared Error) to discuss the precision of competing machine learning models (see Jang & Lee, 2018; Lahmiri & Bekiros, 2019; Phaladisailoed & Numnonda, 2018; Tandon et al., 2019).

Our empirical assessment provides a thorough forecasting study with an exclusive focus on daily Bitcoin prices and the contribution of our study is twofold. First, we provide an economic assessment of statistical time series properties of daily Bitcoin prices to assess if the identified characteristics are in line with the underlying assumptions of econometric benchmark models. Second, we apply state-of-the-art machine learning approaches and analyze the performance of deep learning network architectures and complex recurrent neural network (RNN) layers, namely recurrent Long Short-Term Memory (LSTM). Also, we compare the forecasting performance of both econometric time series models and machine learning techniques to discuss state-of-the-art machine learning techniques against solid econometric benchmarks. Specifically, in addition to the widely used ARMA(1,1) approach, we also take into account conditional volatility clustering via generalized autoregressive conditional heteroscedasticity (GARCH) and assess a variety of competing ARMA-GARCH approaches.

The remainder of this paper is structured as follows. Section 2 provides the relevant literature, Section 3 gives an overview of the methodology, and Section 4 presents the investigated data. The results of the empirical assessment are in Section 5 and Section 6 concludes.

2 | LITERATURE REVIEW

The string of literature, dealing with time series forecasts of cryptocurrencies can be separated into two substrings. One string deals with the application of black-box machine learning approaches and one deals with interpretable econometric approaches.

2.1 | Machine Learning and Bitcoins

An extensive study on machine learning-based forecasts is presented by Alessandretti et al. (2018). The authors study the performance of three models and predict daily cryptocurrency prices of 1,681 currencies from time period between November 11th 2015 and April 24th 2018. Two of the applied approaches are based on gradient-boosting decision trees and one is based on recurrent Long Short-Term Memory (LSTM) neural networks. Then, investment portfolios are built based on the predictions, and the comparison of their performance in terms of return on investment is assessed. The authors provide evidence that all models outperform the baseline "simple moving average" approach. The authors find that the recurrent LSTM approach is characterized by superior forecasting performance. Also, Jang and Lee (2018) provide an empirical horse race between Bayesian Neural Networks (BNNs) with competing linear and non-linear benchmark models to predict Bitcoin price processes. They cover the daily data from September 11th 2011 to August 22th 2017. As a result, BNNs perform well in predicting Bitcoin price time series and explaining relevant volatility components of historic prices. Based on log-transformed market prices and volatility processes, the authors provide experimental evidence that the predictive performance of BNNs outperforms competing benchmark methods. Karasu et al. (2018) also assess Bitcoin price predictions via machine learning using daily data between January 2012 and December 2018. The authors study Support Vector Machines (SVM) and linear regressions to assess daily closing prices. As a result, SVM models are characterized by higher forecasting precision than linear regression models.

Lahmiri and Bekiros (2019) study the application of sophisticated machine learning applications for cryptocurrency prediction for the period between July 10th 2010 and October 1st 2018 and find that the accuracy of more complex RNN layers, namely Long Short-Term Memory (LSTM), significantly increases forecasting precision. As a benchmark, the authors apply a generalized neural regression architecture benchmark. In this vein, Muniye (2020) compares two deep learning techniques, LSTM and Gated Recurrent Unit (GRU) using daily data between January 1th 2014 and February 20th 2018. The results suggest that the GRU model describes an adequate approach for predicting Bitcoin prices as it requires

less compilation time than LSTM. Phaladisailoed and Numnonda (2018) provide similar conclusion based on high-frequency, 1-minute interval trading data from January 1th 2012 to January 8th 2018. The authors compare the performance of competing regression models with RNNs with both LSTM and GRU layers and find that GRU results in superior forecasting accuracy. Also, Tandon et al. (2019) provide a thorough study on Bitcoin price predictions for the period between 2013 and 2019 and compare RNN with LSTM with 10-fold cross validation, linear regression and random forests. The results indicate that cross-validation in RNN with LSTM contributes to the improvement of the efficiency of the model for Bitcoin prediction. The proposed RNN with LSTM using 10-fold cross validation leads to significantly lower MAE (mean absolute error) than random forest and linear regression models.

2.2 | Econometric Time Series Analysis and Bitcoins

In the context of econometric modeling, Chu et al. (2017) provide an extensive study on GARCH modeling dealing with the seven most popular cryptocurrencies using data between June 22nd 2014 and May 17th 2017. They find that IGARCH and GJR-GARCH models result in superior model fit for conditional volatility modeling. Among the assessed GARCH-type models, it is the IGARCH(1,1) model that provides the best fit for Bitcoins. Troster et al. (2019) compare the GARCH and GAS model's forecasting precision to predict the conditional mean and volatility of the Bitcoin return series. They cover the daily data from July 19th 2010 to April 16th 2018. The authors take fat tails into account and demonstrate that the assumption of normally distributed returns gets outperformed by the heavy-tailed GAS approach, measured via Value-at-Risk (VaR) forecasts. By fitting nonlinear econometric models to historical data, Figá-Talamanca and Patacca (2019) study the relative impact of attention measures on both the mean and the variance of Bitcoin returns using data between January 1th 2012 and December 31th 2017. The authors study competing models belonging to the family of ARMA(p,q)-X (E)GARCH(1,1)-X nonlinear models and include attention-related explanatory variables. The results provide evidence that attention measures have a significant impact on the conditional mean and conditional variance of Bitcoin returns, especially when attention is measured in terms of trading volume. The authors assess the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) as well as forecast performance.

2.3 | Machine Learning and Econometric Time Series Analysis

Shen et al. (2021) provide a pioneering study to compare both econometric and machine learning approaches. The authors study GARCH models and RNN with GRU layers to forecast Bitcoin's return volatility and VaR figures on data between April 30th 2013 and May 21th 2021. The authors provide empirical evidence that RNN outperforms GARCH and EWMA in terms of average forecasting performance. Furthermore, RNN shows poor performance in VaR forecasting, indicating that econometric models outperform machine learning in dealing with extreme volatility. Furthermore, this study suggests an alternative method of Bitcoin volatility analysis and finds that machine learning methods perform well in less volatile financial market conditions. McNally et al. (2018) compare both econometric and machine learning techniques and compare Bayesian optimized RNN, LSTM, and ARIMA models for predicting historic market prices of Bitcoins using data between August 19th 2013 and July 19th 2016. The authors demonstrate that non-linear deep learning methods outperform the ARIMA prediction. Also, Cortez et al. (2021) compare the predictions of the ARMA-GARCH model with the K-Nearest Neighbour approach (KNN) applied to market liquidity in cryptocurrencies from February 9th 2018 to February 8th 2019. The authors provide empirical evidence that KNN approaches outperform ARMA and GARCH models in predicting the log rates of the bid-ask spreads. Furthermore, the authors demonstrate that, compared to the ARMA and GARCH models, the KNN approach is more effective at capturing the short-term market liquidity of cryptocurrencies. Further studies that confirm the superiority of Neural Networks when it comes to predicting Bitcoin prices are given by Alessandretti et al. (2018), Jang and Lee (2018), Lahmiri and Bekiros (2019), Phaladisailoed and Numnonda (2018), and Tandon et al. (2019). Most of the papers apply MAE or RMSE to assess the performance of applied models, see Tandon et al. (2019). Furthermore, there are also studies that investigate other features, which could have an effect on the behavior of Bitcoin (Balcilar et al., 2017; Figá-Talamanca & Patacca, 2019; Saad & Mohaisen, 2018; Sin & Wang, 2017; Velankar et al., 2018). As well, Taskaya-Temizel and Casey (2005) also discuss the idea of hybrid approaches and combine time series models with machine learning models.

3 | METHODOLOGY

In this section, we introduce the notation used throughout the paper, define the desirable properties of

econometric time series models and recurrent neural networks, specify long short-term memory, and present the performance metrics.

3.1 | Econometric Time Series Models

In order to adequately capture the relevant characteristics of Bitcoin returns, we apply an autoregressive moving average (ARMA) approach to model the conditional mean of the daily return series. This approach describes a combination of the autoregressive model with p lags and the moving average model with q lags. Furthermore, we also take into account time-varying conditional volatility and autoregressive volatility clustering and apply the generalized autoregressive conditional heteroscedasticity (GARCH) model to the squared residuals of the ARMA approach, with m and n lags, respectively. Then the ARMA(p,q)-GARCH(m,n) model is defined as follows:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{j=1}^{q} \theta_j u_{t-j} + u_t, \text{ with: } u_t \sim iid\ D\left(0,\sigma_t^2\right), \tag{1}$$

$$\begin{aligned}\sigma_t^2 = {} & \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \ldots + \alpha_m u_{t-m}^2 + \beta_1 \sigma_{t-1}^2 \\ & + \beta_2 \sigma_{t-2}^2 + \ldots + \beta_n \sigma_{t-n}^2,\end{aligned} \tag{2}$$

with: $\alpha_0 > 0,\ \alpha_i \geq 0 (i = 1,\ldots,m), \beta_j \geq 0 (j = 1,\ldots,n)$ and $\sum_{i=1}^{max(m,n)} (\alpha_i + \beta_i) < 1$. Where u_t is white noise (has zero mean, constant variance σ^2, and is uncorrelated in time) and the time series y_t are the Bitcoin returns at time t. Also, m describes the order of the ARCH terms and n the order of the GARCH terms. The residuals u_t are characterized by distribution $D(0,\sigma^2)$. We will assess both a Gaussian and a t-distribution. Also, we apply a naive benchmark, which is simply the return from the previous period.

Further, this framework allows us to also assess competing approaches, that are nested in equation (2), namely ARMA(p,q), ARMA(p,q)-GARCH(m,n), and different parameterizations of p,q,m, and n. The parameters are estimated via the maximum likelihood method.

3.2 | Recurrent Neural Networks

Our baseline machine-learning approach is a simple RNN. In comparison to typical Feed-Forward-Networks (FFNs), RNNs include backward connections and therefore this approach is predestined to capture time-dependent features of a time series. Also, by stacking more than one RNN layer, we are able to study Deep RNNs. Let the input and output at time step t be described by X_t and y_t, the weights for the hidden layer by W_{hx} and bias by b_x. The information from the previous time step is h_{t-1} with weights W_{hh} and bias b_h. Then, in order to calculate the hidden state at the time step h_t, that is the information that will be passed to $t+1$, is described as follows:

$$\begin{aligned}& h_t = f_W(h_{t-1}, X_t), \\ & \text{with:} \\ & t = 1,\ldots,M \text{ and } h_0 = 0.\end{aligned} \tag{3}$$

The hidden state at time t can be described as a function with parameters W of the previous hidden state and the input at time t. In our study, we apply the standard hyperbolic tangent function ($tanh$) and f_W. The amount of tuples of the train set is M. Then, we apply RNN and stacked RNNs which are then defined as follows:

$$h_t = tanh(W_{hh} h_{t-1} + W_{hx} X_t + b_h), \tag{4}$$

$$\begin{aligned}& y_t = tanh\left(W_{yh} h_t + b_y\right), \\ & \text{with:} \\ & t = 1,\ldots,M.\end{aligned} \tag{5}$$

For instance, for a Deep RNN with 2 hidden layers, 5 neurons, and a forecasting horizon of 10 days, the output of the last hidden layer (h_t for time step t) is a matrix 10×5, as it contains the results of each neuron. The output layer is a simple layer and transforms this matrix into 10×1 vector of forecasts with the weights W_{yh}, as described in equation 5. From this, the number of neurons can also be seen as a dimension of the hidden state of one layer. In this paper, we study 1, 2, and 3 hidden simple RNN layers in combination with 1, 5, and 10 units and we refer to Géron (2020) for a thorough introduction to RNNs and machine learning.

3.2.1 | Long Short-Term Memory

In addition to simple RNN, we also apply RNN with long short-term memory (LSTM) layer. LSTM was introduced by Hochreiter and Schmidhuer (1997) in order to overcome the problem of long-term dependency data causing a vanishing gradient within a simple RNN framework. In comparison to RNN, the training of LSTM converges

faster and the identification of long-term dependencies in the data is possible. In contradiction to RNN, where every layer gets 2 inputs, at the time step t: X_t and the output of the previous time step $t-1$, namely h_{t-1}, LSTM adds an additional input to each layer. That is, the current long-term memory of the network, known as the cell state c_{t-1} and the hidden state h_t becomes the short-term state, see Géron (2020). The parameterization of the LSTM approach is given as follows:

$$f_t = \sigma\left(W_{fx}X_t + W_{fh}h(t-1) + b_f\right) \tag{6}$$

$$g_t = tanh\left(W_{gx}X_t + W_{gh}h(t-1) + b_g\right) \tag{7}$$

$$i_t = \sigma(W_{ix}X_t + W_{ih}h(t-1) + b_i) \tag{8}$$

$$o_t = \sigma(W_{ox}X_t + W_{oh}h(t-1)h(t-1) + b_o) \tag{9}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{10}$$

$$y_t = h_t = o_t \otimes tanh(c_t) \tag{11}$$

The first step in the LSTM cell is the Forget Gate (f_t) as described in equation (6). Through this gate, certain information is deleted (forgotten) and the output of the Forget Gate ($f_t \otimes c_{t-1}$) is then added to the output of the Input Gate ($i_t \otimes g_t$). The result for c_t gets stored without further transformation as described in equation (10). Hence, at each time step, irrelevant information is discarded and relevant historical information is added, see equation (11). The long-term state c_t is copied after this addition and passed through the *tanh* function. As described in Yu et al. (2019). The number of neurons per LSTM layer denotes the dimensions of the hidden state and of the output state of one layer. Similar to the RNN framework, a dense layer with 1, 5, and 10 neurons is used as an Output Layer for the RNN with LSTM.

In order to train the networks, we apply Adam (Adaptive Moment Estimation) optimization. As described in Géron (2020) Adam optimization combines the idea of Momentum Optimization and RMSProp.

3.3 | Performance Metrics

In addition to the assessment of autocorrelated residual and partial autocorrelation functions, we draw on existing studies and assess the precision of out-of-sample forecasts via both root mean squared error (RMSE) and mean absolute errors (MAE). The metrics are calculated as follows:

$$RMSE_B^T = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{T}\left(y_{ij}^{B,forecast} - y_{ij}^{B,true}\right)^2}{M^*T}}, \tag{12}$$

$$MAE_B^T = \frac{\sum_{i=1}^{M}\sum_{j=1}^{T}\left|y_{ij}^{B,forecast} - y_{ij}^{B,true}\right|}{M^*T}. \tag{13}$$

Where M denotes the number of tuples, B the batch size, 100, 250, or 500 days, and T the number of days that describe the forecasting horizon, 1, 5, or 10 days. Furthermore, in our study, naive forecasting deals as a benchmark for both approaches. That is, the last day of the training sample is used as the forecast for the next consecutive day. Furthermore, in order to assess competing neural nets, we apply a naive fully connected neural net as an additional benchmark.

4 | DATA

The data starts with the first available market price of Bitcoin, April 28th 2013, and ranges until December 12th in 2021.[1] The graphical representation of the prices and logarithmic returns is displayed in Figure 1.

Figure 1 illustrates both daily market prices and daily log returns over time. Table 1 presents the descriptive statistics of the data. We find that daily returns are characterized by a standard deviation of 4.1698 and a mean of 0.1884. These results are in line with Cunha and Silva (2020). However, in comparison to other currencies, as reported by Mignot and Westerhoff (2024), daily Bitcoin returns are characterized by higher volatility.[2]

Table 2 gives the Ljung-Box (LB) test statistics to assess autocorrelation of the logarithmic returns and squared logarithmic returns, the Augmented Dickey-Fuller (ADF) test to investigate stationarity, and the Jarque-Bera (JB) test to study normality of the logarithmic returns. The ADF Test shows, that the time series of Bitcoin daily prices is not stationary. As a result, logarithmic returns are created (hereafter referred to only as returns). As can be seen in Table 2, the Bitcoin daily returns are stationary. As discussed in Box et al. (1994), we set the number of investigated lags for the Ljung-Box test to 20. The standard deviation of Bitcoin returns is described as 4.1698 and, as indicated by the LB test statistic is characterized by statistically significant volatility

[1]Data source is wwww.coinpaprika.com and the applied data set is also available upon request to the authors.

[2]For an in-depth discussion on stylized facts of exchange rates, we refer to Guillaume et al. (1997) and Grauwe and Grimaldi (2006).

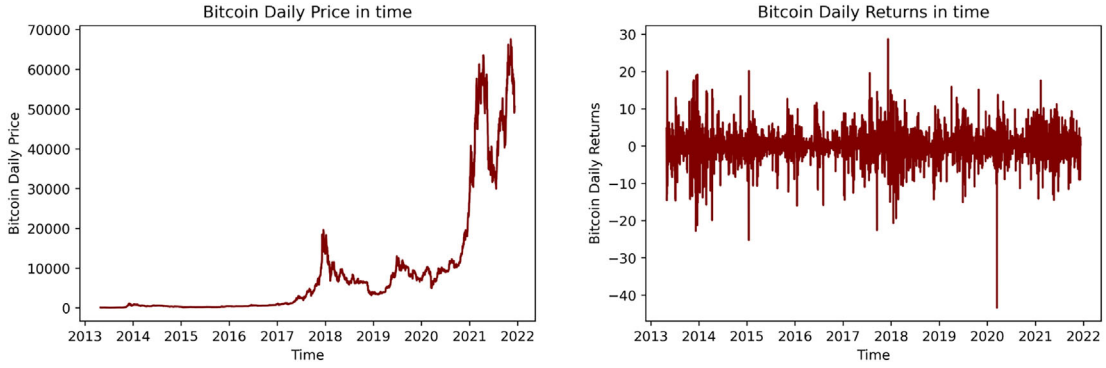

FIGURE 1 Bitcoin price and returns (in %) in time.

TABLE 1 Descriptive statistics.

	Bitcoin daily prices	Bitcoin daily returns
n	3145.00	3143.00
Mean	8818.48	0.19
Std	14568.24	4.17
Min	67.81	−43.37
25%	442.82	−1.37
50%	3410.45	0.21
75%	9239.90	1.90
Max	67617.02	28.71

Note: The table presents descriptive statistics for the daily market prices and return series of Bitcoins from April 28, 2013 to December 12, 2021. The amount of observations is n, Min and Max are the minimum and maximum value of each variable. 25%, 50% and 75% are the respective quartiles of the empirical return distribution and Std is the standard deviation of the mean.

TABLE 2 Ljung-Box, ADF and Jarque-Bera tests.

	Ljung-Box test statistic	P-value
Bitcoin returns	53.70	0.00
Bitcoin squared returns	356.65	0.00
	ADF test statistic	P-value
Bitcoin prices	−0.39	0.91
Bitcoin returns	−14.71	0.00
	Jarque-Bera test statistic	P-value
Bitcoin returns	−21.02	0.00

Note: This table provides the results of the Ljung-Box (LB), Augmented Dickey Fuller (ADF) and Jarque-Bera (JB) test. The Ljung-Box (LB) test is performed with 20 lags with H0: The data is independently distributed and H1: The data is not independently distributed, and exhibits serial correlation. The Augmented Dickey Fuller (ADF) test assesses H0: The series is non-stationary. H1: The series is stationary. And the hypothesis of the Jarque-Bera (JB) test test are H0: The series is normally distributed. H1: The series follows a non-normal distribution. For all test, the p-value is presented.

clusters. However, as illustrated in Figure 2 the autocorrelation of returns decays quickly.

These findings are consistent with Hu et al. (2019), Momtaz (2021), and Zhang et al. (2018) who report similar findings on volatility and volatility clusters. Furthermore, as presented by Zhang et al. (2018) and Borri (2019), based on the JB test, we can confirm that the distribution of Bitcoin returns is non-normal and exhibits fat tails, a stylized fact that is well-known from the stock market and can also be seen in Figure 3.

As described above, Bitcoin returns are in line with stylized facts of financial return series, as also shown in the literature. The facts that are shown in this paper are high volatility, volatility clustering, and fat tails.

In order to assess forecasting accuracy, we separate the data into overlapping tuples comprising 110, 260, and 510 days. Hence, the training set of the tuple consists of 100, 250, and 500 days, the additional 10 days are used for assessing the goodness of a forecast. All in all, there are 3,032 tuples of 110 days, 2,882 tuples of 260 days, and 2,632 tuples of 510 days for $T = 100{,}250$, and 500 and 10 days ahead forecast.

An example of this separation is displayed in Figure 4, the underlying data for the machine learning exercise is separated into train, validation, and test data sets. Following the study from Shen et al. (2021), the tuples are characterized as follows. The train dataset contains the first 70% of the tuples, the validation data set the next consecutive 20% and the test data set the remaining 10%. Then, based on the trained model, the train and validation data set describe the in-sample data, and the test data is the out-of-sample data set. As illustrated in Figure 4, the data can then be split into different data sets. In our study, we assess training and validation sets comprising $T = 100{,}250$, and 500 days and study forecast of 1,5 and 10 days ahead.

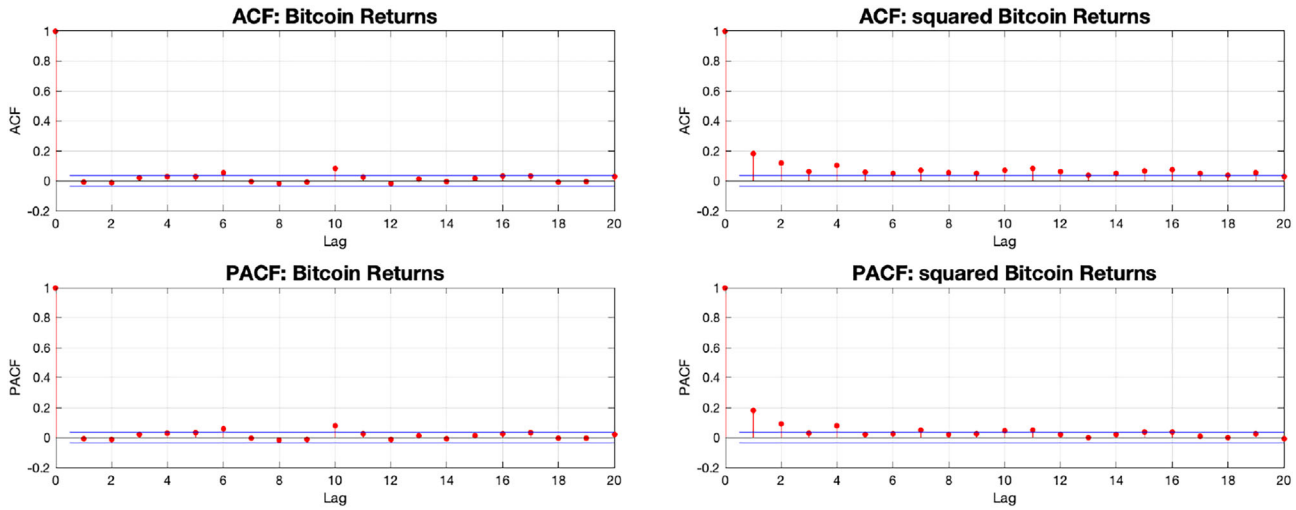

FIGURE 2 Autocorrelation function and partial autocorrelation function of daily returns and squared Returns. The upper graphs visualize the Autocorrelation function (ACF) of daily Bitcoin returns (left) and squared Bitcoin returns (right). The lower graphs provide information on the partial autocorrelation for Bitcoin returns (left) and squared Bitcoin returns (right).

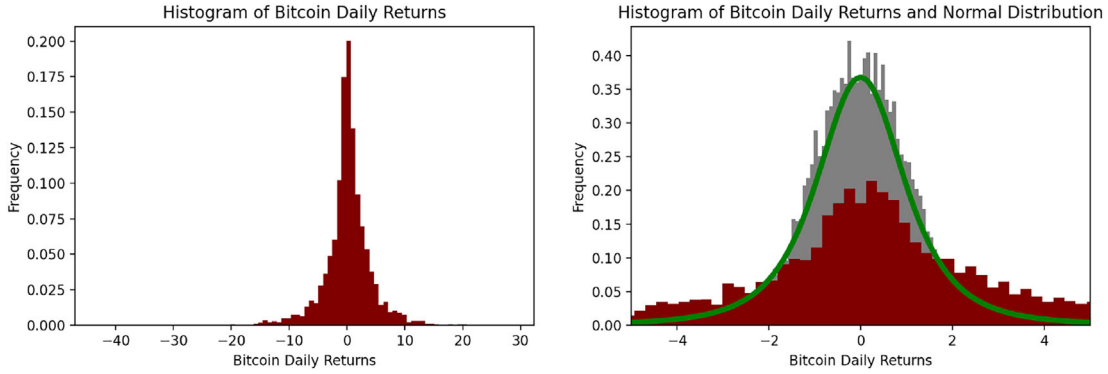

FIGURE 3 Histograms of Bitcoin Returns (in %) and comparison with Normal Distribution.

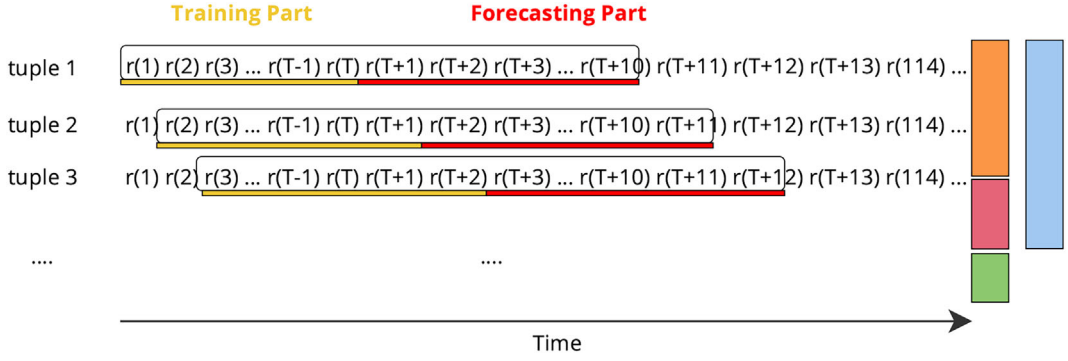

FIGURE 4 Separating the data into tuples for recurrent neural nets (RNN). Each tuple contains training data (yellow) and labeled forecasting data (red). The data is a time series and is described by consecutive daily Bitcoin returns. In order to train a neural net, 90% of the tuples are used to train and validate the neural net, and 10% are applied to test the results. The training and test data is split into 70% training and 20% testing data the test data presents the basis for the evaluation of out-of-sample forecasts.

TABLE 3 AIC values for ARMA(p,q) and GARCH(m,n) model.

ARMA Gaussian distribution						
p/q	**0**	**1**	**2**	**3**	**4**	**5**
1	−11,046.28	−11,044.39	−11,043.19	−11,046.69	−11,047.48	−11,046.82
2	−11,044.72	−11,043.24	−11,044.73	−11,076.25	−11,052.37	−11,073.12
3	−11,043.90	−11,047.88	−11,050.74	−11,051.88	−11,067.92	−11,067.21
4	−11,044.79	−11,048.39	−11,053.40	−11,035.87	−11,044.69	−11,067.93
5	−11,045.94	−11,048.40	−11,051.41	−11,077.58	−11,035.10	−11,035.78
t-distribution						
p/q	**0**	**1**	**2**	**3**	**4**	**5**
1	−12,055.74	−12,054.43	−12,055.72	−12,054.39	−12,056.16	−12,086.41
2	−12,055.07	−12,055.66	−12,058.30	−12,085.20	−12,089.96	−12,083.32
3	−12,054.64	−12,054.46	−12,089.46	−12,060.73	−12,090.63	−12,088.63
4	−12,054.49	−12,056.17	−12,085.86	−12,090.29	−12,094.27	−12,089.88
5	−12,053.56	−12,088.41	−12,056.28	−12,092.36	−12,092.34	−12,091.47
GARCH						
m/n	**0**	**1**	**2**	**3**	**4**	**5**
1	−12237.04	−12547.45	−12529.13	−12551.13	−12549.13	−12531.89
2	−12310.91	−12543.04	−12527.54	−12548.28	−12538.42	−12531.98
3	−12361.34	−12525.42	−12526.80	−12548.17	−12529.90	−12525.40
4	−12421.43	−12539.38	−12473.29	−12524.98	−12534.34	−12543.49
5	−12452.35	−12536.82	−12539.46	−12536.74	−12144.82	−12538.43

Note: This table presents the AIC criterion for different combinations of lags under the assumption of Gaussian and t-distributed residuals.

TABLE 4 Forecasting performance: Econometric approach.

	Naive benchmark			**ARMA-GARCH**		
RMSE	**1 day**	**5 days**	**10 days**	**1 day**	**5 days**	**10 days**
100 days	0.0585	0.0578	0.0576	0.0427	0.0431	0.0427
250 days	0.0567	0.0558	0.0556	0.0405	0.0402	0.0401
500 days	0.0563	0.0555	0.0554	0.0398	0.0401	0.0408
Mean	0.0572	0.0564	0.0562	0.0410	0.0411	0.0412
MAE	**1 day**	**5 days**	**10 days**	**1 day**	**5 days**	**10 days**
100 days	0.0391	0.0391	0.0391	0.0284	0.0282	0.0279
250 days	0.0382	0.0381	0.0381	0.0265	0.0263	0.0262
500 days	0.0380	0.0379	0.0379	0.0259	0.0260	0.0260
Mean	0.0384	0.0384	0.0384	0.0269	0.0268	0.0267

Note: This table presents the out-of-sample forecasting performance of the naive benchmark is the return from the previous period and the applied ARMA(4,4)-GARCH(1,3) model. The forecasting horizon is 1 day, 5 days, and 10 days and the metrics are the root mean squared errors (RMSE) and mean absolute error (MAE).

5 | EMPIRICAL ASSESSMENT

In this section, we discuss the empirical results of our forecasting study. Due to the fact, that we compare two competing methodological approaches, namely econometric modeling and machine learning, we provide a discussion on the optimal parameterization of both approaches, econometric and machine learning,

TABLE 5 Forecasting performance: Neural nets and different batch sizes.

RMSE			
Batch size	**100 days**	**250 days**	**500 days**
Naive benchmark	0.0588	0.0588	0.0572
Fully connected NN	0.0423	0.0435	0.0445
Simple RNN	0.0414	0.0402	0.0396
Deep RNN	0.0414	0.0403	0.0397
RNN with LSTM	0.0414	0.0404	0.0395
MAE			
Batch size	**100 days**	**250 days**	**500 days**
Naive	0.0457	0.0447	0.0400
Fully connected NN	0.0322	0.0328	0.0341
Simple RNN	0.0306	0.0300	0.0295
Deep RNN	0.0308	0.0301	0.0296
RNN with LSTM	0.0308	0.0303	0.0296

Note: This table presents the out-of-sample forecasting performance of the naive benchmark and the applied machine-learning techniques. The naive benchmark is the return from the previous period, fully connected NN is the basic fully connected neural net, simple RNN is the recurrent neural network with one layer, and deep RNN is the recurrent neural net with max. Three layers and RNN with LSTM is a recurrent neural network with a Long Short-Term layer. The forecasting metrics are the root mean squared errors (RMSE) and mean absolute error (MAE) and are the average of 1 day, 5 days, and 10 days forecasting horizon.

separately. Then, we assess the differences between the two identified optimal approaches jointly.

5.1 | ARMA-GARCH

In order to identify the optimal ARMA-GARCH approach for daily Bitcoin returns, we begin with the identification of optimal lags within the ARMA framework. Therefore, different combinations of the ARMA(p,q) model are estimated based on the full training set. We assess an autoregressive order p varying from 1 to 5 and a moving average order q varying from 0 to 5 and assess both Gaussian and t-distributed residuals.[3] Then, we choose the optimal ARMA (p,q) order according to the AIC criterion. In the next step, we calibrate the optimal lag size of GARCH(m,n) approach, again based on the AIC criterion.

Table 3 provides the results for the choice of optimal ARMA-GARCH parameterization. As a result, we find that an ARMA(4,4)-GARCH(1,3) approach with t-distributed residuals provides the best fit to the underlying data. For the sake of robustness, we also estimate ARMA(p,q)-EGARCH(m,n) and ARMA(p,q)-GJR-GARCH(m,n) approaches. Both approaches take an asymmetric impact from negative returns on conditional volatility into account, however, there is no improvement in terms of model fit. Then, we apply the ARMA(4,4)-GARCH(1,3) model to each tuple (100, 250, 500 days) in order to forecast the upcoming 10 days and compare the forecasting precision against the naive benchmark.

Table 4 provides information on both forecasting metrics, RMSE and MAE, for the ARMA(4,4)-GARCH(1,3) and the naive benchmark model. It displays the results for all three tuple sizes (100 days, 250 days, and 500 days) as well as for all three forecasting horizons (1 day, 5 days, and 10 days). In comparison to the naive benchmark, the estimated ARMA-GARCH approaches outperform the benchmark. Hence, the results unveal, that modeling economic dynamics of conditional mean and variance of Bitcoin returns series via ARMA and GARCH approaches, increases the precision of forecasts.

5.2 | Recurrent Neural Networks

In the next step, we identify the optimal RNN structure for our forecasting exercises. In contradiction to the econometric ARMA-GARCH approaches, the hyperparameter tuning of RNN comprises more parameters and is, therefore, more complex. First, the number of epochs, layers, and neurons is determined based on the train and validation data set. With a focus on model loss, we choose the minimal number of epochs in order to avoid overfitting. Specifically, for each RNN, we assess the model loss after each training epoch in order to identify the lowest model loss in combination with the smallest amount of epochs.

Then, the number of neurons and hidden layers is determined for Deep RNN and RNN with LSTM. The number of epochs is the same for all combinations, as this proves to be sufficient also for other combinations. The RNNs are trained and evaluated for different batch sizes (100, 250, and 500 days), different forecasting horizons (1, 5, and 10 days), different depths (1, 2, and 3 layers), different complexity (5, 10, 15, 20, and 80 neurons in each layer), and different structure (simple layer and LSTM).

5.2.1 | Epochs

The results of the assessment of epochs are fourfold. First, we find typical performance curves characterized by decreasing model loss at the beginning for all

[3]The range for the autoregressive and moving average orders was chosen according to correlograms. The correlograms, as well as a detailed list of all configurations, are available upon request to the authors. Results are available upon request to the authors.

TABLE 6 Forecasting performance: Neural nets and different forecasting horizons.

RMSE						
Sample and horizon	**Tr + Val 1 day**	**Test 1 day**	**Tr + Val 5 days**	**Test 5 days**	**Tr + Val 10 days**	**Test 10 days**
Naive	0.0573	0.0592	0.0574	0.0579	0.0574	0.0578
Fully connected NN	0.0376	0.0434	0.0376	0.0434	0.0376	0.0434
Simple	0.0400	0.0404	0.0400	0.0404	0.0399	0.0404
Deep	0.0400	0.0406	0.0398	0.0403	0.0398	0.0405
LSTM	0.0394	0.0404	0.0397	0.0406	0.0398	0.0404
MAE						
Sample and horizon	**Tr + Val 1d**	**Test 1d**	**Tr + Val 5d**	**Test 5d**	**Tr + Val 10d**	**Test 10d**
Naive	0.0393	0.0441	0.0396	0.0433	0.0396	0.0431
Fully connected NN	0.0253	0.0331	0.0254	0.0330	0.0254	0.0330
Simple	0.0255	0.0301	0.0255	0.0300	0.0256	0.0301
Deep	0.0256	0.0301	0.0255	0.0302	0.0255	0.0302
LSTM	0.0252	0.0303	0.0254	0.0303	0.0253	0.0300

Note: This table presents the in-sample and out-of-sample forecasting performance of the naive benchmark (the return from the previous period) and the applied machine learning techniques. Fully connected NN is the basic fully connected neural net, Simple RNN is the recurrent neural network with one layer, deep RNN is the recurrent neural net with max. Three layers and RNN with LSTM is a recurrent neural network with a Long Short-Term layer. The forecasting metrics are the root mean squared errors (RMSE) and mean absolute error (MAE) and are the average of the assessed batch sizes comprising 100 days, 250 days, and 500 days. The forecasting horizon is 1, 5, and 10 days and Tr + Val is the training and validation data, namely the in-sample assessment, and Test is the test data set, the out-of-sample performance.

combinations of Fully connected NNs, and for Simple RNNs and Deep RNNs with batch size of 100 days. Second, based on batch sizes 250 and 500 days, in Simple RNNs, Deep RNNs, and RNNs with LSTMs, we observe loss curves, which imply, that the number of epochs does not impact the model loss measured in terms of RMSE and MAE. Third, loss plots show patterns of overfitting, however, the loss difference between the train and validation sample is not remarkable (maximal 0.00014), this is relevant for fully connected NNs, Simple RNNs, Deep RNNs, and RNNs with LSTM, all with a batch size of 500 days. Fourth, some graphs display a higher model loss in the train sample than in the validation sample, nevertheless, the maximal difference is 0.0002 (Simple RNN: batch size of 250 days, Deep RNN: batch size of 250 days, RNN with LSTM: batch size of 100 and 250 days).[4]

5.2.2 | Neurons and Layers

The RMSE of Deep RNN with two hidden layers and five neurons in each layer is smaller than that of Deep RNN with two hidden layers and ten neurons for both the train and validation sample for all days ahead.[5] This also applies to the MAE criterion. The additional third layer also does not increase the performance. As a result, Deep RNN with two hidden layers and five neurons presents an adequate choice, evaluated on both RMSE and MAE. The average RMSE and MAE of RNN with LSTM with two hidden layers and five neurons are also lower than those with two layers and ten neurons. The same applies to an additional third layer for RNN with LSTM.

5.2.3 | Batch size

We find that an increasing batch size goes in line with the superior forecasting performance of the models, except fully connected NN. Both RMSE and MAE of the Fully Connected NN do rise with increasing batch size.

Table 5 presents the average out-of-sample forecasting performance for all approaches with respect to batch size. We can see that RNN is described by the lowest RMSE for 100 and 250 days as well as by the lowest MAE for all batch sizes. The performance of Simple, Deep RNN, and RNN with LSTM are comparable and the RMSE is equal for all three approaches for a batch size of 100 days. Fully connected NN is described by higher RMSE and MAE than the

[4]For sake of page constraints, all graphs are available upon request to the authors.

[5]We discuss the average RMSE of all batch sizes.

remaining three models; however, it is still lower than that of Naive Forecasting.

5.2.4 | Evaluation

Based on the previous findings, we assess the forecasting performance for 1 day, 5 days, and 10 day ahead forecasts, for each fitted RNN, namely simple RNN, Deep RNN, and RNN with LSTM. Also, we assess naive forecasting and a fully connected neural net as a benchmark.

Table 6 gives the averaged RMSE and MSE of each approach. With consideration of RMSE, Simple RNN, and RNN with LSTM seem to result in similar forecasting performance for 1 and 10 days. For 5-day ahead forecasts, it is the Deep RNN that results in the highest forecasting precision. The difference between Simple RNN and RNN with LSTM is marginal. Simple RNN describes the adequate approach in terms of MAE for the forecasts of 1 and 5 days. The 10-day ahead predictions are adequately estimated by RNN with LSTM, however, the difference between this approach and Simple RNN is 0.0001. Also, the forecasting performance of Simple RNN, Deep RNN, and RNN with LSTM is similar in terms of RMSE and MAE. On the other hand, the performance of Fully connected NN is characterized by lower accuracy. However, the RMSE and MAE of Fully connected NN is still lower than that of Naive Forecasting and the performance of Simple RNN and RNN with LSTM does not differ for 1 and 10-day forecasts.

5.3 | Comparison

Based on the previous sections, we now discuss the optimal models and compare interpretable econometric approaches with black-box machine learning techniques. In order to assess the robustness of our results, we assess both different forecasting horizons and different batch sizes for model evaluation. The average forecasting performance of all approaches for 1 day, 5 day, and 10 day ahead forecasts is presented in Table 7.

With respect to RMSE, neural nets outperform the econometric ARMA-GARCH approach. This finding is

TABLE 7 Forecasting performance: ARMA-GARCH and neural nets for different forecasting horizons.

RMSE			
Forecasting horizon	**1 day**	**5 day**	**10 day**
Naive benchmark	0.0592	0.0579	0.0578
Fully connected NN	0.0434	0.0434	0.0434
Simple RNN	0.0404	0.0404	0.0404
Deep RNN	0.0406	0.0403	0.0405
RNN with LSTM	0.0404	0.0406	0.0404
ARMA-GARCH	0.0410	0.0411	0.0412
MAE			
Forecasting horizon	**1 day**	**5 day**	**10 day**
Naive benchmark	0.0441	0.0433	0.0431
Fully connected NN	0.0331	0.0330	0.0330
Simple RNN	0.0301	0.0300	0.0301
Deep RNN	0.0301	0.0302	0.0302
RNN with LSTM	0.0303	0.0303	0.0300
ARMA-GARCH	0.0269	0.0268	0.0267

Note: This table presents the comparison of the out-of-sample forecasting performance between the naive benchmark (return from the previous period), the econometric ARMA-GARCH approach, and the applied machine learning techniques. ARMA-GARCH is the optimal ARMA (p,q)-GARCH(m,n) approach, Fully connected NN is the basic fully connected neural net, Simple RNN is the recurrent neural network with one layer, Deep RNN is the recurrent neural net with max. Three layers and RNN with LSTM is a recurrent neural network with a Long Short-Term layer. The forecasting metrics are the root mean squared errors (RMSE) and mean absolute error (MAE) and present the average of the assessed batch sizes comprising 100 days, 250 days, and 500 days. The forecasting horizon is 1, 5, and 10 days.

TABLE 8 Forecasting performance: ARMA-GARCH and neural nets for different batch sizes.

RMSE			
Batch size	**100 days**	**250 days**	**500 days**
Naive	0.0588	0.0588	0.0572
Fully connected NN	0.0423	0.0435	0.0445
Simple RNN	0.0414	0.0402	0.0396
Deep RNN	0.0414	0.0403	0.0397
RNN with LSTM	0.0414	0.0404	0.0395
ARMA-GARCH	0.0428	0.0403	0.0402
MAE			
Batch size	**100 days**	**250 days**	**500 days**
Naive	0.0457	0.0447	0.0400
Fully connected NN	0.0322	0.0328	0.0341
Simple RNN	0.0306	0.0300	0.0295
Deep RNN	0.0308	0.0301	0.0296
RNN with LSTM	0.0308	0.0303	0.0296
ARMA-GARCH	0.0281	0.0263	0.0260

Note: This table presents the comparison of the out-of-sample forecasting performance between the naive benchmark (the return from the previous period), the econometric ARMA-GARCH approach, and the applied machine learning techniques. ARMA-GARCH is the optimal ARMA (p,q)-GARCH(m,n) approach, Fully connected NN is the basic fully connected neural net, Simple RNN is the recurrent neural network with one layer, Deep RNN is the recurrent neural net with max. Three layers and RNN with LSTM is a recurrent neural network with a Long Short-Term layer. The forecasting metrics are the root mean squared errors (RMSE) and mean absolute error (MAE) and are the average of 1 day, 5 days, and 10 days forecasting horizon.

consistent with the findings presented in McNally et al. (2018), where Deep Learning methods outperform the ARIMA predictions. However, in contradiction to these findings, the performance of Simple RNN and RNN with LSTM for 1 and 10 days ahead (0.0404) and that of Deep RNN for 5 days ahead (0.0403) is similar. For the forecast of 5 days, Deep RNN is characterized by superior out-of-sample performance. However, the ARMA-GARCH approach outperforms both benchmarks, Fully connected NN and Naive Forecasting, in terms of RMSE. Furthermore, ARMA-GARCH results in the lower MAE for 1, 5, and 10 days ahead (0.0269, 0.0268, and 0.0267).

Table 8 provides information on the averaged out-of-sample forecasting performance, based on the test data, for 1, 5, and 10 days ahead for each batch size (100, 250, and 500 days). According to RMSE, the trained RNN approaches outperform the assessed optimal ARMA-GARCH approach. Furthermore, all three RNN approaches are characterized by similar performance.

Our results confirm the findings of Lahmiri and Bekiros (2019) and Makridakis et al. (2018), complex LSTM layers provide adequate forecasting performance in comparison to competing neural nets. However, as we also compare complex approaches against simpler benchmarks, we find that due to less complexity, it is the simple RNN approach that describes a sensible choice when it comes to forecasting daily Bitcoin returns. Also, when the performance metrics change from RMSE to MAE, it is the interpretable econometric ARMA-GARCH model that results in the lowest MAE.

Consequently, our results indicate that less restrictive RNNs describe a fruitful approach for forecasting daily financial Bitcoin returns. However, when the evaluation metric changes from RMSE to MAE, a parsimonious econometric time series approach cannot be outperformed by competing neural nets.

6 | CONCLUSION

In this paper, we present a thorough comparison between widely accepted econometric time series models and state-of-the-art machine learning algorithms. We provide empirical evidence that properties of daily Bitcoin returns are in line with stylized facts of financial return series, which means that underlying assumptions of econometric models hold for daily Bitcoin returns.

Furthermore, our study provides an extensive empirical assessment of competing forecasting approaches for daily Bitcoin returns. We demonstrate that the applied machine learning techniques can outperform econometric ARMA-GARCH approaches. Specifically, simple RNNs, Deep RNNs, and RNNs with LSTM layers are characterized by lower RMSE than the econometric benchmarks. Due to the fact, that the forecasting precision of the competing machine learning techniques differs marginally, our results indicate that simple RNNs as well as econometric time series approaches describe a solid choice for daily Bitcoin return forecasts. Hence, our findings contradict Lahmiri and Bekiros (2019), increasing complexity does not impact the precision of the assessed daily forecasts.

ACKNOWLEDGEMENTS

Open Access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST STATEMENT

None of the authors have a conflict of interest to disclose.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Theo Berger https://orcid.org/0000-0003-0968-8262

REFERENCES

Alessandretti, L., ElBahrawy A., Aiello L. M., & Baronchelli A. (2018). Anticipating cryptocurrency prices using machine learning. *Complexity 2018*, 8983590. https://doi.org/10.1155/2018/8983590

Balcilar, M., Bouri, E., Gupta, R., & Roubaud, D. (2017). Can volume predict Bitcoin returns and volatility? A quantiles-based approach. *Economic Modelling*, *64*, 74–81. https://doi.org/10.1016/j.econmod.2017.03.019

Berger, T., & Gencay, R. (2018). Improving daily value-at-risk forecasts: The relevance of short-run volatility for regulatory quality assessment. *Journal of Economic Dynamics and Control*, *92.C*, 36–46. https://doi.org/10.1016/j.jedc.2018.03.016

Borri, N. (2019). Conditional tail-risk in cryptocurrency markets. *Journal of Empirical Finance*, *50*, 1–19. https://doi.org/10.1016/j.jempfin.2018.11.002

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time series analysis: Forecasting and control* (3rd ed.). Prentice Hall.

Chu, J., Chan, S., Nadarajah, S., & Osterrieder, J. (2017). GARCH modelling of cryptocurrencies. *Journal of Risk and Financial Management*, *10*(4), 17. https://doi.org/10.3390/jrfm10040017

Cortez, K., Rodríguez-García, M. P., & Mongrut, S. (2021). Exchange market liquidity prediction with the K-nearest neighbor approach: Crypto vs. fiat currencies. *Mathematics*, *9*, 1.

Feng, G., He, J., & Polson, N. G. (2020). Deep learning for predicting asset returns". In: *arXiv preprint arXiv:1804.09314*. *1*(1), 1.

Figá-Talamanca, G., & Patacca, M. (2019). Does market attention affect Bitcoin returns and volatility? *Decisions Econ Finan*, *42*, 135–155. https://doi.org/10.1007/s10203-019-00258-7

Géron, A. (2020). *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. 2. Auflage. O'Reilly Media.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Guillaume, D., Dacorogna, M., Davé, R., Müller, U., Olsen, R., & Pictet, O. (1997). From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. *Finance and Stochastics*, *1*, 95–129. https://doi.org/10.1007/s007800050018

Halbleib, R., & Pohlmeier, W. (2012). Improving the value at risk forecasts: Theory and evidence from financial crisis. *Journal of Economic Dynamics and Control*, *36*, 1212–1228. https://doi.org/10.1016/j.jedc.2011.10.005

Hochreiter, S., & Schmidhuer, J. (1997). Long short-term memory. *Neural Computation*, *80*(12), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, A. S., Parlour, C. A., & Rajan, U. (2019). Cryptocurrencies: Stylized facts on a new investible instrument. *Financial Management*, *48*(4), 1049–1068. https://doi.org/10.1111/fima.12300

Jang, H., & Lee, J. (2018). An empirical study on modeling and prediction of bitcoin prices with Bayesian neural networks based on blockchain information. *IEEE Access*, *6*, 5427–5437. https://doi.org/10.1109/ACCESS.2017.2779181

Karasu, S., Altan, A., Saraç, Z., & Hacıoğlu, R. (2018). Prediction of Bitcoin prices with machine learning methods using time series data. In *2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018* (pp. 1–4). https://doi.org/10.1109/SIU.2018.8404760

Kraus, M., Feuerriegel, S., & Oztekin, A. (2020). Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, *281*, 628–641. https://doi.org/10.1016/j.ejor.2019.09.018

Lahmiri, S., & Bekiros, S. (2019). Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons and Fractals*, *118*, 35–40. https://doi.org/10.1016/j.chaos.2018.11.014

Longo, L., Riccaboni, M., & Rungi, A. (2022). A neural network ensemble approach for GDP forecasting. *Journal of Economic Dynamics and Control*, *134*(1), 104278. https://doi.org/10.1016/j.jedc.2021.104278

Makridakis, S., Spiliotis, S., & Assimakopoulos, E. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *In: PloS ONE*, *13*(3), e0194889.

McNally, S., Roche, J., & Caton, S. (2018). Predicting the price of bitcoin using machine Learning. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), Cambridge, UK, 21–23 March 2018* (pp. 339–343). https://doi.org/10.1109/PDP2018.2018.00060

Mignot, S., & Westerhoff, F. (2024). Explaining the stylized facts of foreign exchange markets with a simple agent-based version of Paul de Grauwe's chaotic exchange rate model. *Computational Economics*. https://doi.org/10.1007/s10614-024-10546-z

Momtaz, P. P. (2021). The pricing and performance of cryptocurrency. *The European Journal of Finance*, *27*(4–5), 367–380. https://doi.org/10.1080/1351847X.2019.1647259

Muniye, Temesgen (Nov. 2020). "Bitcoin price prediction and analysis using deep learning models". In.

Phaladisailoed, T., & Numnonda. T. (2018). Machine learning models comparison for Bitcoin Price Prediction. In *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), Bali, Indonesia, 2018* (pp. 506–511). https://doi.org/10.1109/ICITEED.2018.8534911

Saad, M. and Mohaisen D. (Apr. 2018). "Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions". In: pp. 704–709.

Shen, Z., Wan, Q., & Leatham, D. J. (2021). Bitcoin return volatility forecasting: A comparative study between GARCH and RNN. *Journal of Risk and Financial Management*, *14*(7), 337. https://doi.org/10.3390/jrfm14070337

Sin, E. and Wang L. (July 2017). "Bitcoin price prediction using ensembles of neural networks". In: pp. 666–671.

Tandon, S., Tripathi, S., Saraswat, P., & Dabas, C. (2019). Bitcoin price forecasting using LSTM and 10-fold cross validation. In *2019 International Conference on Signal Processing and Communication (ICSC)* (pp. 323–328).

Taskaya-Temizel, T., & Casey, M. C. (2005). "A comparative study of autoregressive neural network hybrids". In: *Neural Networks* 18.5. *IJCNN*, *2005*, 781–789. https://doi.org/10.1016/j.neunet.2005.06.003

Troster, V., Tiwari, A. K., Shahbaz, M., & Macedo, D. N. (2019). Bitcoin returns and risk: A general GARCH and GAS analysis. *Finance Research Letters*, *30.C*, 187–193. https://doi.org/10.1016/j.frl.2018.09.014

Velankar, S., Valecha, S., & Maji, S. (2018). Bitcoin price prediction using machine learning. In *2018 20th International Conference on Advanced Communication Technology (ICACT)* (pp. 144–147).

Yu, Y., Si X., Hu C., and Zhang J. (July 2019). "A review of recurrent neural networks: LSTM cells and network architectures". In: *Neural Computation 31*.7, pp. 1235–1270. https://doi.org/10.1162/neco_a_01199

Zhang, W., Wang, P., Li, X., & Shen, D. (2018). Some stylized facts of the cryptocurrency market. *Applied Economics*, *50*(55), 5950–5965. https://doi.org/10.1080/00036846.2018.1488076

AUTHOR BIOGRAPHIES

Theo Berger is a Professor of quantitative methods at the University of Applied Sciences Hannover, with interests in risk management, forecasting and econometric approaches to problems in Finance. Prior to this, he worked at DHL Corporate Treasury Department and developed a risk management concept based on value-at-risk figures. He has a PhD in Financial Econometrics.

Jana Koubova is a Data Scientist at Sparkasse Rating and Risikosysteme Berlin, with interest in machine learning, risk management and data analytics. She holds a master's degree in Business Administration.

How to cite this article: Berger, T., & Koubová, J. (2024). Forecasting Bitcoin returns: Econometric time series analysis vs. machine learning. *Journal of Forecasting*, *43*(7), 2904–2916. https://doi.org/10.1002/for.3165