Economic Forecasting

Forecasting Bitcoin Using Sentiment

Authors
Christoffer Lundgaard
201606313
Sebastian Emborg Stefansen
201609361

Supervisor
Mikkel Bennedsen
Adjunkt, Aarhus University

Aarhus University , Department of Economics and Buisness Economics

M.Sc. Oecon

Hand-in date: 15/05/2020

Authors take equal responsibility of the report and its content.

Contents

1 Introduction and Literature Review

The recent decade has seen a new and intriguing market materialize and gain traction at an unprecedented rate. The cryptocurrency market has emerged, and Bitcoin is undoubtedly at the center of it, being responsible for 65% of the entire market capitalization at the time of writing[1]. The cryptocurrency market is regarded as a highly speculative market [Bouoiyour & Selmi, 2015], and Bitcoin has experienced enormous fluctuations in prices during its lifetime. Forecasting in such an unstable environment is challenging. This paper strives to evaluate the forecasting performance of both traditional econometric forecasting methods, machine learning methods, and a combination of these. We seek to add to the current Bitcoin forecasting literature by conducting our analysis utilizing sentiment derived from the online forum website Reddit.com. This is done to accommodate the speculative nature of the cryptocurrency market, and especially the immensely discussed Bitcoin, by quantifying the atmosphere residing in the investor community.

Bitcoin has emerged as the main contender for a new type of world currency. A currency that unlike all its predecessors is not backed by any government, and operates entirely as a free market system, free of any central banking system. These attributes have helped it gain continuing traction in the investment field and at it's peak on December 17th, 2017, where the price of a single Bitcoin was USD 20.089. However, while these characteristics sound intriguing, they come with some caveats, as sinister speculation extends into the concern that Bitcoin could be an environment for money laundering and organized crime [Kristoufek, 2015]. Further, as Bitcoin has no intrinsic value, its price evolution could entirely be a result of speculative bubbles, [Bouoiyour & Selmi, 2015], and there is no consensus on the fundamental value of Bitcoin among scholars[Derbentsev *et al.* , 2019]. The fact that Bitcoin is proposed as a replacement for our current monetary systems, along with the immense traction the cryptocurrency market has received as an investment opportunity, warrants exploration and motivates forecasting.

The current literature surrounding the prediction of Bitcoin prices has moved in many exciting directions, for instance, an approach in which the price bubbles observed for Bitcoin are modeled using Hidden Markov Models, initially created by epidemiologists to track the spread of disease, has proven a useful foundation for an investment strategy.[Phillips & Gorse, 2017].

We delve into the part of the literature concerning the impact of investor sentiment on the price of Bitcoin, as we argue that the atmosphere amongst investors can, in large part be viewed as an expression for the future demand [Bouoiyour & Selmi, 2015, Jain *et al.* , 2018b, Kim *et al.* , 2016, Galeshchuk *et al.* , 2018]. This viewpoint is inspired by scholars regarding the prevailing thesis that the exchange rate of Bitcoin is determined by the ratio of demand and supply [Derbentsev *et al.* , 2019, Selmi *et al.* , 2018, Bouoiyour & Selmi, 2015, Cheah & Fry, 2015, Ciaian *et al.* , 2016, Blau, 2017, Balcilar *et al.* , 2017, Stiglitz & Rogoff, 2018]. Analysis of investor sentiment and asset returns has nonetheless been examined long before Bitcoin was created [Neal & Wheatley, 1998, Shiller, 2000,

[1] 15-03-2020

Baker & Wurgler, 2000, Baker & Wurgler, 2006].

Within the field of cryptocurrency, there have been found impressive results using sentiment to predict Bitcoin prices. It has been found that the exchange rate of Bitcoin depends on behavioral signals as opposed to any fundamental conditions, implying that investor sentiment can explain market shocks that can not be explained by the efficient market hypothesis. Specifically, a significant influence of Twitter signals on Bitcoin price fluctuations. Inclusion of the sentiment scores improved the accuracy of the Bitcoin directional changes [Galeshchuk *et al.* , 2018]. Research also found significant relations when using Bitcoin forum comments to forecast the price, where sentiment derived from user comments found to be a key driving factor in explaining Bitcoin price fluctuations. [Jain *et al.* , 2018b]. This use of sentiment, derived from user comments in online communities, to predict Bitcoin's price has shown to perform well enough to validate a method applicable to cryptocurrency trading [Kim *et al.* , 2016]. Sentiment has also shown to do well for prediction in the alternative cryptocurrency market [Li *et al.* , 2019].

The models we analyze in this paper have been selected based on their prevalence in the current Bitcoin forecasting literature, where they each have proven to perform well in the prediction of Bitcoin. There is a tendency for the machine learning methods, especially the Long Short-Term Memory Recurrent Neural Network (LSTM), to outperform the econometric models. We seek to analyze this recurrence ourselves, and also examine the possibility of a forecast combination.

The comparison between forecasting Bitcoin using an ARIMA model and using an LSTM, or other neural network specifications, is a prevalent one in the Bitcoin forecasting literature. As ARIMA is a heavily used econometric forecasting technique used across most forecasting fields, it has warranted itself as a proper benchmark to use against other forecasting techniques. This tendency in the Bitcoin forecasting literature is also what has inspired the choice of the models used in this paper.

Within machine learning techniques has LSTMs receive a large amount of attention, as they are specifically designed to better handle longer-term dependencies compared to their original recurrent neural network predecessors. They have shown to outperform ARIMA specifications and various machine learning techniques when predicting Bitcoin[Karakoyun & Cibikdiken, 2018, Rebane *et al.* , 2018, Huisu *et al.* , 2018, Alessandretti *et al.* , 2018]. However, other neural network specifications have also proven useful, as there is evidence that a sequence to sequence recurrent deep multi-layer neural network outperforms ARIMA specifications when predicting Bitcoin prices [Rebane *et al.* , 2018].

The further two forecasting techniques utilized in this paper, VAR and Boosting, are also present in the current Bitcoin forecasting literature. Vector Autoregression (VAR) is used to show the relation between Bitcoin prices and investor interest, as investor sentiment further increases (decreases) prices in times of upwards (downwards) explosive prices.[Kristoufek, 2013]. VAR methods have also shown to deliver sizable directional predictability within Bitcoin forecasting [Catania *et al.* , 2019]. Likewise to the LSTM literature, boosting is another popular fore-

casting technique from the world of machine learning, and has generated good prediction results, both when including measures for investor sentiment [Li *et al.* , 2019], and when compared to LSTMs. Relative to LSTMs, it is found that boosting performed better for short predictions, 5 to 10 days, indicating that their strength lies in short-term dependencies, while LSTM performed better on $\sim$ 50 prediction days, indicating their strength regarding longer term-dependencies [Alessandretti *et al.* , 2018].

When dealing with complex data with potentially different regimes, the notion of a single encompassing model is quite naive. Different forecasting frameworks have different strengths and weaknesses. The concept of combining different forecasts of the same outcome has increased in popularity since Bates and Granger's article from 1969 [Bates & Granger, 1969] and was later reviewed in a meta article by Clemen in 1989 [Clemen, 1989]. Researchers like Stock and Watson have on several occasions applied forecast combination for economic predictions [Stock & Watson, 1999, Stock & W Watson, 2003, Stock & Watson, 2004]. Combination has been established to be beneficial when individual models are likely to be miss-specified, and the underlying data process is unstable. The performance of an ensemble forecast further depends on what is combined. Forecasts have to be substantially different and can be based on different data sets. Due to the nature of Bitcoin's recent price development and the difference in forecasting methods applied in this paper, a study of the effect of combination is ideal. We investigate various weighting schemes analyzing their effect on forecast error when predicting the Bitcoin price using four different model frameworks made from two different information sets.

This paper seeks to determine the performance of the most prevalent forecasting models when incorporating social media sentiment within the field of Bitcoin. We further conduct forecast combinations, in order to examine whether optimal performance could be obtained by utilizing both the econometric models and the machine learning techniques with and without sentiment.

The following sections of this paper are constructed as follows; Section 2 will go into the data used for the prediction analysis, how the data was gathered and handled. Section 3 contains a short methodological run-through of each of the forecasting strategies we make use of, and the methodology regarding forecast combination. On the econometric side, this includes Autoregressive Integrated Moving Average Models (ARIMA) and Vector Autoregression (VAR) models. The machine learning methods used include Long Short-Term Memory Recurrent Neural Networks (LSTM) and Extreme Gradient Boosting (XGB). Section 4 is an evaluation of the forecasting performance of each model, conducted both including and excluding the sentiment factors followed by cross-comparison between the models, including tests regarding superior predictive abilities, and the evaluation of combination forecasts. Section 5 reflects over the limitations encountered and proposes extensions to this work. Finally, Section 6 concludes the paper and reflects on the results observed.

2 Data

This part seeks to give insight into the data used in the forecasting exercise. It covers the data source, how it was collected and processed. The time series consists of two parts; Bitcoin financial data and social media sentiment data. This paper focuses on daily granularity and covers the period 26-06-2017 to 18-09-2019 where 27-06-2018 to 18-09-2019 is put aside as test set for evaluating prediction models, as seen in 1.

To retrieve Bitcoin financial data, we utilize the REST API from cryptocompare.com. The time series consists of high, low, and close prices in U.S. Dollars and traded volume for the day as a volume-weighted average from leading cryptocurrency exchanges. Furthermore, we create an intraday volatility proxy as the difference between the daily high and low divided by the closing price. The use of these predictors is inspired by Balcilar [Balcilar *et al.* , 2017].

Sentiment indicators are constructed on user comments made on the website Reddit.com. Reddit is a platform that aggregates news, which users submit, rate, and discuss. It is divided into moderated theme-specific subreddits, where users express their opinion on a matter by up- or downvoting a post or comment submitted by other users. It is the 13th most visited website in the world and fifth in the united states, making Reddit comments excellent to use in sentiment analysis.

To collect the data, we use BigQuery's API. BigQuery is a branch of the cloud platform from Google for big data. From the database fh-bigquery.reddit_comments, we query raw comment data that contains either the word "Bitcoin" or abbreviation "BTC". As the goal is to derive sentiment from each comment as either positive, neutral, or negative, each comment is filtered in a four-step procedure to increase the performance of the natural language processor, NLP. First by deleting special characters and non-English letters, then filtering out all comments coming from known foreign specific subreddits, afterward processing each comment by Google's language detection algorithm keeping English comments only and finally we filtered out all comments from users flagged as known bots [Shuyo, 2010]. With the cleaned comments, we derive the sentiment using a rule-based sentiment analysis tool specifically designed for social media. In this paper, we use the Valance Aware Dictionary and sEntiment Reasoner made by C.J. Hutto and Eric Gilbert [Hutto & Gilbert, 2015]. It assigns each word in a string with a polarity score based on a lexicon approach where a set of words is given a determined sentiment. To enhance the performance of the NLP, we incorporate a broad financial corpus by Lounghran-McDonald and further manually add trading and Bitcoin-specific slang[2]. Subsequently, we classify comments using the polarity score as positive if larger than 0.05, negative if less than -0.05 and neutral if in-between. As a last step comments are grouped by date and joined onto the financial data set by the date.

The quality of the derived sentiment is crucial in a study like this. If the comments contain

[2]I.e. trading slag as "bearish" as negative and "bullish" as positive and cryptocurrency slang as "hodl" which is the most used word on the Bitcoin subreddit and is a purposely misspelling of the world hold, and is interpreted as positive.

too much noise it is not possible to derive any information or potentially be fooled. Sarcasm and comments from bots can be distinguished by a human, but it is a capability the algorithm does not have. We do believe that the implementation of filters does validate the data quality in an automated way.

1 contains descriptive statistics for all considered variables.

As the econometric models require stationary data for proper predictions, this subsection covers pre-tests for the presence of unit root in the time series and how non-stationarity is handled. We apply a combination of the Augmented Dickey-Fuller test, ADF, and the KPSS test [Dickey & Fuller, 1979, Kwiatkowski *et al.* , 1992]. The ADF test defines the regression:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{j=1}^{q} \delta_j \Delta y_{t-j} + \varepsilon_t \tag{1}$$

Under the null hypothesis of the presence of unit root, $\gamma = 0$, against the alternative of no unit root, $\gamma < 0$, equation 2 follows the test statistic the Dickey-Fuller t-distribution:

$$DF = \frac{\hat{\gamma}}{SE\left(\hat{\gamma}\right)} \tag{2}$$

The KPSS test has the null hypothesis of stationarity and calculates a test statistic in three steps;

1. Perform an auxiliary regression of $y_t = r_t + \beta t + \varepsilon_t$ and save residuals $\{e_t\}_{t=1}^{T}$.
2. Compute cumulative residuals $S_t = \sum_{s=1}^{t} \varepsilon_s \forall t$
3. Compute the test statistic: $KPSS = T^{-2} \sum_{t=1}^{T} \frac{S_t^2}{\widehat{\sigma}^2}$

The test statistic follows a one sided LM statistic. If the associated $p-$value of a test is below 5% we reject the null of stationarity in favor of unit root.

We chose a decision rule that if either test result was non-stationary, the series would be differenced and re-tested for stationarity. As seen in A.2, the results from these tests conclude that all raw data is non-stationary and therefore adjusted.

Figure 1: Line plot of price data splot into training and test period. For interactive version in browser: `https://plotly.com/~Emborg/630/#/`

Table 1: Descriptive statistics for data

	Mean	Std	Skewness	Kurtosis
price	2,011.34	3,370.27	1.99	3.49
volatility	0.07	0.10	6.20	66.46
volume_number	360,761.90	638,508.49	2.77	8.23
positive_comment	679.13	7,135.50	3.03	16.10
neutral_comment	132.86	153.16	3.38	18.83
negative_comment	386.58	421.45	3.11	16.93

3 Methodology

The methodology section of this paper goes through each of the forecasting methods we utilize in this paper. These sections start by explaining the methodology behind the classical econometric models ARIMA and VAR, and two machine learning methods, Gradient Boosting and LSTM, and finish by introducing the weighting schemes for forecast combination.

3.1 Autoregressive Integrated Moving Average (ARIMA)

The first of the models we use is an AutoRegressive Integrated Moving Average, ARIMA, statistical regression model. This is a typical model for time series forecasting in economics, as it utilizes previous values of a time series whilst also handling non-stationarity of the time series. The model stems from the ARMA model, which is a combination of an autoregressive, AR, and a moving average, MA, model [Box *et al.* , 2015].

An $AR(p)$ model denotes an autoregressive model of order p, indicating that, for a univariate time series y with $y_i \in \mathbb{R}$, the time t value will be expressed as

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \varepsilon_t \tag{3}$$

with $\phi_1, ..., \phi_p$ being the model parameters, and ε_t as an expression for white noise. The autoregressive model is then a model where the time t predicted value is a function of the previous p values.

The moving average takes a similar expression, where the model predicts based on current and past shocks to the series. An $MA(q)$ model takes the following expression

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \tag{4}$$

With $\varepsilon_t, ..., \varepsilon_{t-q}$ indicating the shocks to the time series, and $\theta_1, ..., \theta_q$ being the parameters.

The combination of these two models then constitutes an $ARMA(p, q)$ model

$$y_t = \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \tag{5}$$

However, ARMA models can generally only be applied to stationary time series (REF), and thus differencing is introduced, $y_t^{(1)} = y_t - y_{t-1}$. Incorporating this into the $ARMA(p, q)$ model the constitutes an $ARIMA(p, d, q)$ specification (REF Terence C Mills. 1991. Time series techniques for economists. Cambridge University Press), where d is the order of differencing needed to make the series stationary, in our case $d = 1$. The $ARIMA(p, d, q)$ is expressed as

$$y_t^{(d)} = \phi_1 y_{t_1}^{(d)} + ... + \phi_p y_{t-p}^{(d)} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \tag{6}$$

In this analysis we optimize the parameters, p and q, by applying a grid search to choose the optimal set of parameters, evaluated using the Basyesian Information Criterion, BIC. Choosing the lag orders for p and q is done by looking at the estimated variance of the residuals, $\hat{\sigma}^2 = T^{-1}\sum_{\tau=1}^{T}\hat{\varepsilon}_\tau^2$, and checking the information criterion.

$$BIC = ln\left(\hat{\sigma}^2\right) + n\frac{ln\left(T\right)}{T} \tag{7}$$

with n being the number of parameters estimated, $p + q + 1$. This information criterion is calculated for each combination of parameters and the model with the lowest value is chosen. We also include exogenous variables in our model specification, our ARIMA specification thus becomes an ARIMAX specification.

3.2 Vector Autoregression (VAR)

This section delivers a brief introduction to the Vector AutoRegressions, VAR, popularized by Sims, who initially advocated for the ability to estimate economic relationships unconstrained by theory compared to, at the time, dominant structural models [Sims, 1980]. The model later caught on in the world of financial forecasting with dividend forecasting by Campell and Shiller [Campbell & Shiller, 1988]. VARs are now a central procedure for analyzing relationships between multivariate time series in modern econometrics.

The VAR model describes linear time dependent evolution for a set of n endogenous variables. As a generalization of the univariate auto-regression to a multivariate system where y_t represents the $(n \times 1)$ vector. We define the pth order VAR for y_t as:

$$y_t = c + \sum_{i=1}^{p} A_i y_{t-i} + u_t \tag{8}$$

Where y_{t-i} is the ith lag of y_t, A_i is an $(n \times n)$ matrix containing the time-invariant auto-regressive coefficients for the $i = 1, ..., p$ lags, c_t is a $(n \times 1)$ intercept vector and ε_t is a $(n \times 1)$ error term vector that satisfies; zero mean $E\left[\varepsilon_t\right] = 0$, contemporaneous covariance matrix $E\left[u_t u_t'\right] = \Sigma$ and no correlation across time $E\left[\varepsilon_t \varepsilon_{t-k}'\right] = 0$ where $t \neq 0$. The VAR model does require input variables to be mean-variance stationary for the assumptions on the error term to hold and estimation of parameters to be consistent. Implementation of the model requires two choices done by the forecaster; selection of variables $[y_1, .., y_n]$ and the lag structure p. Overparameterization is at high risk due to the amount of estimated parameters increasing at a quick pace.

We employ a search over a set range to select the models lag structure, where the selection criteria are the BIC.

The chosen model will be the most parsimonious within one standard deviation of the minimum as we seek to keep the complexity of the model at a minimum for the empirical forecasting exercise.

3.3 Gradient Boosting (XGB)

This section seeks to introduce and motivate the usage of gradient boosting as a forecasting model. In recent years gradient boosted trees have gained considerable popularity and proved particularly successful in time series prediction competitions. The fundamental idea of boosting was first introduced by Robert Schapire [Schapire, 1990]. Jerome H. Friedman later developed a branching method called gradient boosting. [Friedman, 2001]

Boosting is a tree-based method. Tree models centers around segmentation of the feature space into disjoint regions R_j, $j = 1, 2, ..., J$. The regions are the terminal nodes of the tree. The prediction of a tree model equals the mean within the region for which a given observation is associated with as $x \in R_j \Rightarrow f(x) = \gamma_j$. A complete grown tree can be expressed as:

$$T(x; \Theta) = \sum_{j=1}^{J} \gamma_j I\left(x \in R_j\right) \tag{9}$$

Where $\Theta = \{R_j, \gamma_j\}_1^J$ corresponds to the regions and the constant value used for prediction for that specific region.

The classic decision tree model offers a lot as it is useful for interpretation and fairly simple, but it often fails to offer adequate predictive power. We therefore employ the variant method of boosting. Boosting focuses on growing a multiple of trees which are grown sequentially where each tree will use the information of the one previously grown. The boosting algorithm slowly learns instead of fitting a single large tree with a high potential of overfitting. This is achieved by fitting trees to the residuals of the previous model instead of the predicted variable. Afterward, this newly estimated tree is added to the previous building an ensemble where each tree attempts to correct the faults of the preceding. Formally we define a boosted tree as:

$$f_M(x) = \sum_{m=1}^{M} T\left(x; \Theta_m\right) \tag{10}$$

Which is a specific case of a basis function expansion using decision trees. Following a forward step-wise procedure to solve for:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L\left(y_i, f_{m-1}\left(x_i\right) + T\left(x_i; \Theta_m\right)\right) \tag{11}$$

Being the contribution of tree m to the additive model. As we know from above for tree m and region j will be:

$$\hat{\gamma}_{jm} = \arg\min_{\gamma_{jm}} \sum_{i: x_i \in R_{im}} L\left(y_i, f_{m-1}\left(x_i\right) + \gamma_{jm}\right) \tag{12}$$

We employ a variant of boosting referred to as gradient tree boosting. Here the goal is to

fit models to the pseudo residuals set equal to the negative gradient of the loss function at step m for the ith observation instead of the residuals of the previous models as in the standard version. A pseudocode representation of the iterative method can be seen in 1. [Friedman, 2001, Hastie *et al.* , 2001]

To correctly tune the model the boosting algorithm requires setting three tuning parameters;

B, which is the number of grown trees. If set too large, the model can potentially overfit the data. If too low, the model does not have iterations enough to recognize the relationship between features and the predicted variable.

λ, is the shrinkage parameter that manages the learning rate of the ensemble. By setting this, the user can control the effect a newly grown tree has. If set at a low rate the method requires a high number of grown trees B.

d, is the number of splits in each tree. This determines the depth of the tree, and therefore the complexity of the B trees.

All tuning parameters have to be selected by forward cross-validation as we are working with time-series data. These are all influential for controlling the bias-variance trade-off of the model. B ensures enough trees are grown and thereby average out expected model variance but can also cause overfitting if set too high. Similar for d, where a tree with $d = 1$, also called a stump, will cause high bias compared to deeper trees with higher for d. Analogously for λ.

Comparison of gradient boosting to the related methods of bootstrap aggregating (bagging) and random forests in regards to expected test error and bias-variance trade-off. Bagging is a similar ensemble method which grows trees on bootstrapped training samples drawn from the original with replacement. This leads to reduced model variance by taking the average over models built on different training sets.A pitfall for bagging is if there exists a highly prominent feature, this will cause highly correlated trees as it will be used for all trees in the ensemble and cause a high bias [Breiman, 1996]. In contrast, random forests grow full trees but with a random sub-sample of the feature set available at each split point. These have high variance due to the complexity of the trees but low bias due to the randomization of the features decorrelating the trees from each other. Boosting seeks to achieve both low bias and variance by reducing tree complexity and fitting not the sample but iteratively the residual of the base model.

Algorithm 1 Boosting for Regression Trees

1. Set $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $b = 1, 2, ..., B$:
   (a) For $i = 1, 2..., N$ compute the negative gradient of the loss wrt. $f(x_i)$:
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$
   (b) Fit a regression tree to the targets r_{im} giving set terminal regions $R_{jm}, j = 1, 2, ..., J_m$
   (c) For $j = 1, 2, ..., J_m$ compute:
   $$\gamma_{jm} = \arg\min_{7} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
   (d) Update ensemble with new tree:
   $$f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{j_m} \gamma_{jm} I(x \in R_{jm})$$
3. Final model $\hat{f}(x) = f_M(x)$

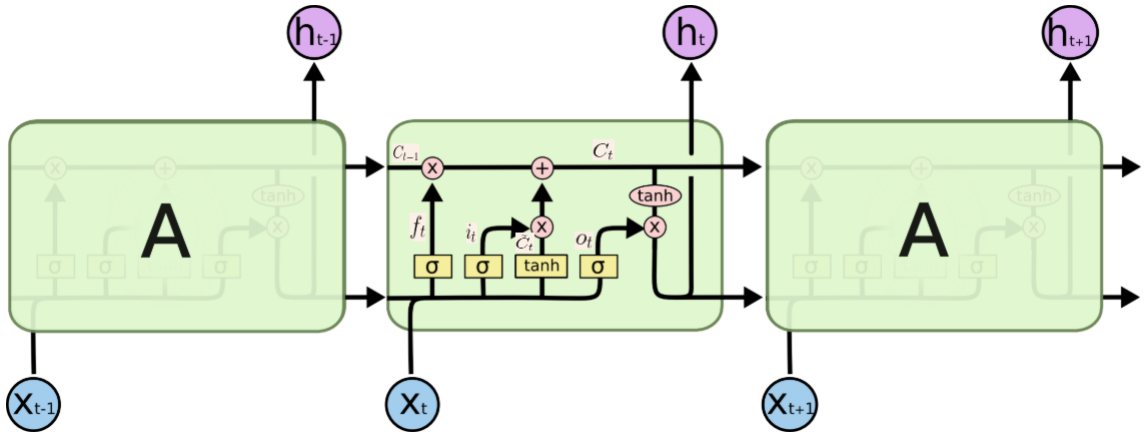
3.4 Long Short-Term Neural Network (LSTM)

As our final forecasting method, we utilize a Long Short-Term Memory Recurrent Neural Network, LSTM, as developed by Hochreiter[Hochreiter & Schmidhuber, 1997]. For brevity purposes, the following will focus on the properties of the LSTM that distinguishes it from a regular Recurrent Neural Network, for more information regarding Neural Networks and their Recurrent version, RNN, please see [Hastie *et al.* , 2001].

The LSTM is an extension to the more basic RNN, as the simpler RNN performs poorly when handed the task of modeling long-term dependencies in time-series data [Hochreiter & Schmidhuber, 1997, Bengio *et al.* , 1994]. LSTMs were constructed to handle this problem; they are designed for long-term dependencies and are widely used in the economic forecasting field[Karakoyun & Cibikdiken, 2018, Rebane *et al.* , 2018, Huisu *et al.* , 2018, Alessandretti *et al.* , 2018].

As with RNNs, LSTMs are formed by a chain of neural network modules, however, the difference lies in what happens within these modules. For a standard RNN, there would likely be a single neural network layer taking the exogenous variables and the output of the former link in the chain as inputs, using for instance a tanh function layer. In an LSTM, the interworkings of these modules become a bit more complex, as they are built up of four interacting layers, the structure can be seen in the Figure 2 below:

Figure 2: LSTM Module



[Colah, 2015]
A figure of the module design of an LSTM. The circles outside the module represent the respective exogenous variables and the neural network outputs for each time period. The arrows denote vector transfers within the module. The squares within the module denote learned neural network layers, where 'σ' denotes a sigmoid function and 'tanh' a tanh function. The circles denote pointwise vector operations.

The central part of the LSTM structure is the horizontal line running along the top of the module, which is referred to as the cell state, C_t. The cell state is the main differentiator between an LSTM and a regular RNN. The cell state runs through the entire module chain and is thus the primary retainer of past information. However, it is adjusted within each separate module along the way by three specific gates. Following the graphical representation seen in the above figure, the first vertical line represents the "forget gate," f_t. The forget gate takes the exogenous variables, x_t, and the output of the preceding module, h_{t-1}, as inputs and pass them through a neural network layer using the sigmoid function. This in effect outputs a vector of values between 0 and 1, which is pointwise multiplied onto the cell state. These values can be viewed as weights for the information passed from the previous cell state:

$$f_t = \sigma\left(b_f + W_f \cdot [h_{t-1}, x_t]\right) \tag{13}$$

Having weighted the information passed on from the previous cell state, the next two vertical lines define what new information is to be added to the cell state. Another sigmoid layer denoted the "input gate" i_t, determines the values to be updated by creating another weighting vector. The second part is then a tanh layer, scaling the values between -1 and 1, determining the candidate values to add to the cell state, $\tilde{C}_t$. These vectors are pointwise multiplied, and the resulting vector is added to form the time t cell state:

$$i_t = \sigma\left(b_i + W_i \cdot [h_{t-1}, x_t]\right), \ \tilde{C}_t = tanh\left(b_C + W_C \cdot [h_{t-1}, x_t]\right) \tag{14}$$

The updated cell state is thereby calculated as:

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

In effect, the cell state has been updated by forgetting the values determined by the forget gate and adding the new relevant information determined in the input gate. The last part of the module is then calculating the output of the model, h_t. This action can be seen in the righter part of the central module. The LSTM calculates its prediction by scaling the updated cell state using a tanh function, this is then weighed by the 'sigmoid gate', o_t, which works in the same manner as the previously explained gates.

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

The output of the module is then

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

hese interworkings of LSTMs make them capable of modeling longer-term relations, and why they are so heavily used in the forecasting community. However, the above stated is a standard version of an LSTM, the version used in this paper, but there exist further versions. Notably, the 'peephole' LSTM introduced in [Gers & Schmidhuber, 2000], and the highly popular Gated Recurrent Unit, GRU, LSTM introduced in [Cho *et al.* , 2014].

3.5 Weighting Schemes

This subsection covers how the different forecast combination weighting schemes studied in the empirical analysis are defined. Various methods exist for estimating weights in an ensemble forecast. This paper implements four different; simple equal-weighted average, mean square error-based, ranked error, and time-varying weights. The main difference between the first and the latter three is the attempt to improve the forecast by incorporating the known forecast error of the previous predictions. This active approach, yet sophisticated and sensible, has continuously in empirical findings been outperformed by the simple weighting scheme [Stock & Watson, 2004]. These results lead Stock and Watson to define this as the "forecast combination puzzle" in 2004.

As we are conducting time-series predictions, weights are only defined on past predictions; this means the weighting vector, ω_i, is calculated at time t and utilized as weights for predicting time $t+1$ to ensure fairness in prediction.

For good measure, the equal-weight scheme is for m forecasts defined as:

$$f^{\text{ew}} = \frac{1}{m}\sum_{i=1}^{m} f_i \tag{18}$$

The mean squared-error weights are calculated by:

$$\omega_i = \frac{\text{MSE}_i^{-1}}{\sum_{i=1}^{m} \text{MSE}_i^{-1}} \tag{19}$$

Where MSE is the sum of losses up to time, t, for model i and weights ω_i are applied for forecast of $t+1$. The weights are thereby proportional to the inverse of the individual models' MSE values.

The ranking scheme by Aiolfi and Timmermann in 2006 lets weight i be inversely proportional to the models' MSE_i rank, by ascending ranking the m models, thus the model with the lowest MSE_i attains the rank 1: [Aiolfi & Timmermann, 2006]

$$\omega_i = \frac{Rank_i^{-1}}{\sum_{i=1}^{m} Rank_i^{-1}} \tag{20}$$

Finally is the time-varying approach proposed by Bates and Granger [Bates & Granger, 1969]. This adaptive method uses a rolling window of size v recent observations.

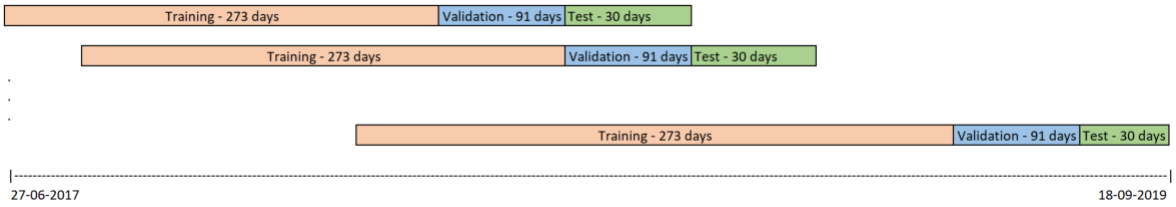
$$\hat{\omega}_{it} = \frac{\left(\sum_{\tau=t-v+1}^{t} e_{i\tau}^2\right)^{-1}}{\sum_{j=1}^{m} \left(\sum_{\tau=t-v+1}^{t} e_{j\tau}^2\right)^{-1}} \tag{21}$$

Where $e_{i\tau}$ is forecast error at time τ for model i. This allows for more dynamic weights as eq. 21 is in essence a rolling window version of eq. 19.

4 Empirical Analysis

Before going into the empirical results, we go through how each of the respective model's parameters was chosen, the procedure of model selection. As mentioned in the Data section, each of the models are using stationary data, so they are in actuality predicting Bitcoin returns; however, these forecasts are returned to their level state afterward. Forecasting is conducted from the 27th of June 2018 to enable our models to train on the large price fluctuations seen at the end of 2017 and the beginning of 2018. All forecasts are done with a rolling window of one year of training data. If the model requires parameter tuning, a 91 day period is set aside to conduct forward cross-validation. When model selection is finished, a final model is estimated on the entire year, and a one-step forecast is done for a 30 day period. After 30 days, models are re-tuned and re-estimated. This adds up to 15 rolling forecasts. This process is depicted in Figure 3.

Figure 3: Forward cross-validation procedure for rolling window forecasting



4.1 Model selection

4.1.1 ARIMA

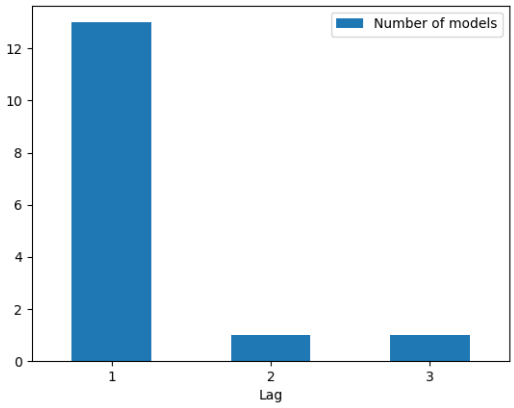
The first models run are the ARIMAX specifications. The exogenous variables included in the model are the one day lagged values of the variables specified in the Data section. From here, we conduct a grid search over parameter combinations from 0 to 3 lags of both autoregressive and moving average lags. A plot of resulting lags can be seen in Figure A.1. Each model is fit on the preceding year of data, and the model is chosen by achieving the smallest value of the BIC. The appertaining parameter specification is then used to forecast the following month of Bitcoin prices. Once these have been forecast, the model parameter specification is initialized again to select the optimal parameters for forecasting the next month. The size of the training data is fixed to the preceding year of data.

4.1.2 VAR

VAR model specification follows an equivalent process as the ARIMAX. The chosen model is the one with the lowest BIC from a grid search from 1 lag up to 3 lags of auto-regressive lags for all financial and sentiment variables with one year of training data. This selection rule is repeated

with the redefinition of lag structure and re-estimation of parameters each month. 4 displays the chosen lag structure of the 15 models estimated over the testing period.

Figure 4: Frequency of chosen lag structure



4.1.3 XGB

Modeling using the gradient boosting algorithm requires feature engineering as it is not a dedicated time series model. We create date related features as the day of the month, day of the year, week, etc. which is commonly done in related literature [Silva, 2014]. Further, we pass one day lagged variables of the financial and sentiment data. As covered in the methodology section for the gradient boosting model, it requires forward cross-validation for selection of tuning parameters; the number of trees, B, learning rate, λ, and split nodes d. This is done by following the depicted procedure in Figure 3. We conduct three one month one-step forecasts within the validation set and select the model configuration with the lowest average RMSE. 5 and 6graphs the gain in validation RMSE for the three tuning parameters for the first 30 days of forecasting. As a need to reduce computational costs we split tuning into two parts by doing a grid search over the $B = \{500, 200, 175, 150, 125, 100, 75, 50\}$ and $d = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and then a subsequent grid search over $\lambda = \{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3\}$. In general, we observed a low level of model complexity for the test set. Additionally, gradient boosting allows a measure of variable importance wrt reducing the residual squared error. This is plotted in 7, again for the first period of forecasting. We see the lagged value of the number of positive comments chosen most frequently, followed by two of the time indicators.

Figure 5: Forward cross-validation RMSE for different B and d tuning parameters
As alluded to in the methodology section regarding the time varying combination schemes, these are a rolling window version of the MSE weighting scheme

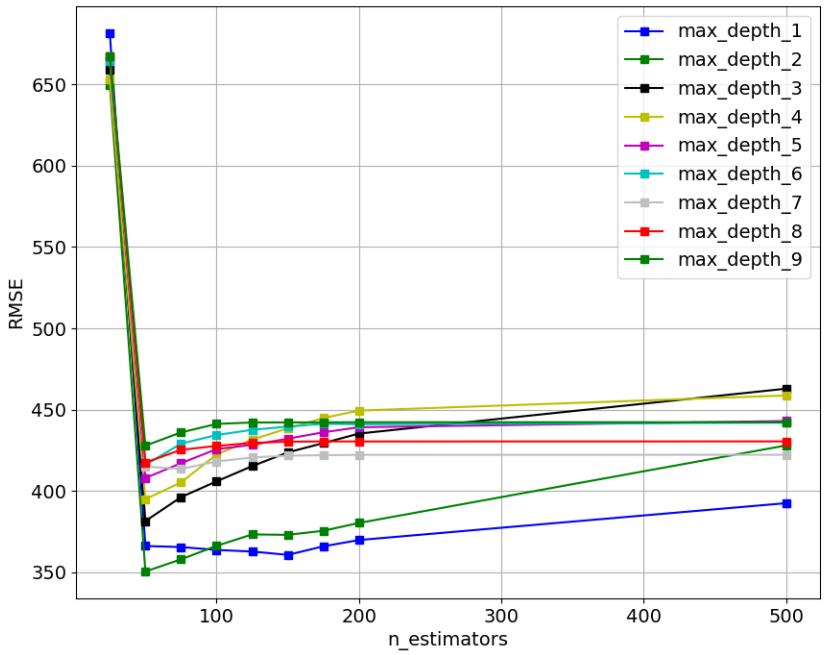


Figure 6: Forward cross-validation RMSE for different λ tuning parameter.

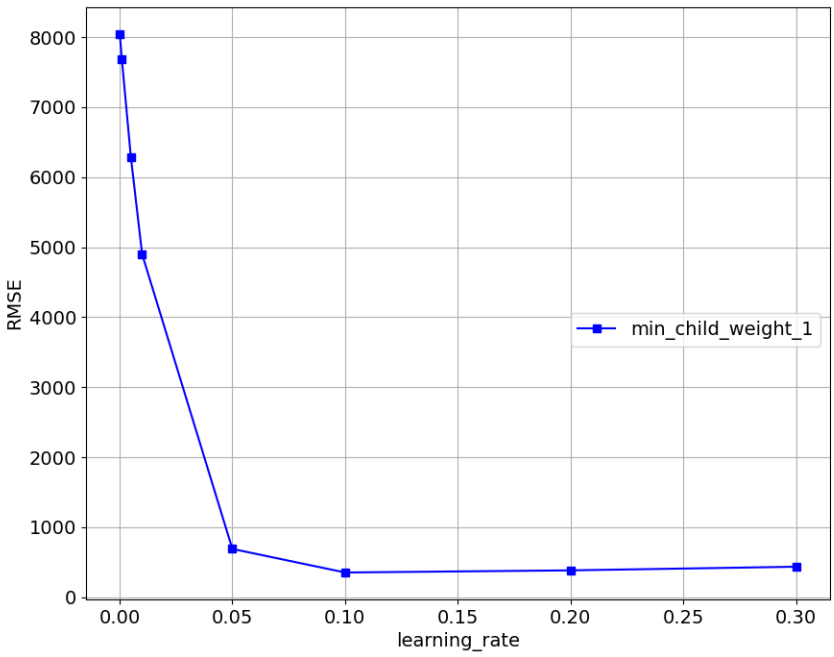
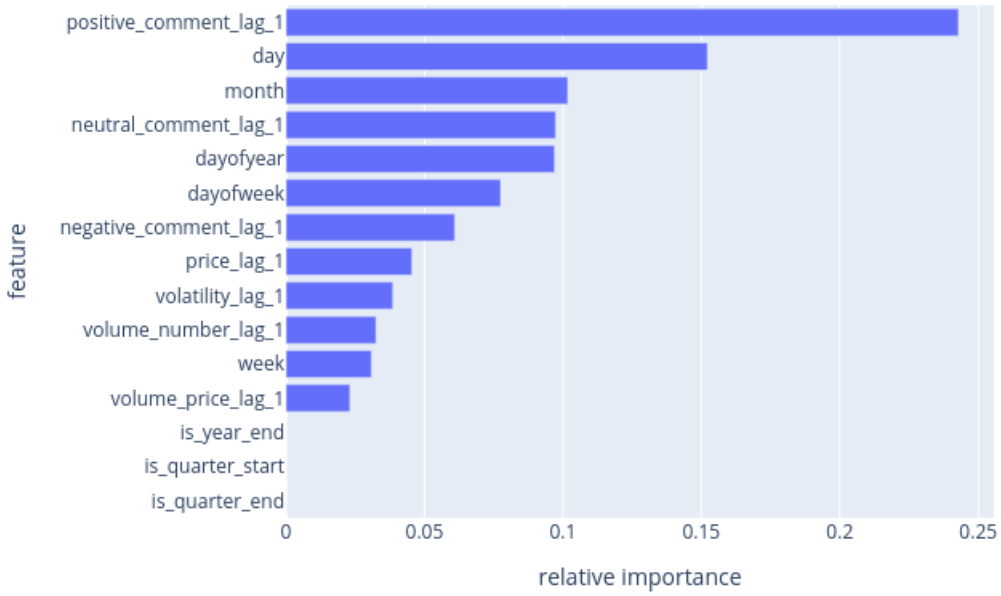


Table containing variable importance for first month of one step forecasts for the gradient boost model with sentiment variables. The measure proposed by Breiman for relevance for each predictor X_ℓ is the average of the B fitted trees $\mathcal{I}_\ell^2 = \frac{1}{B}\sum_{b=1}^{B}\sum_{t=1}^{J-1}\hat{\imath}_t^2 I(v(t)=\ell)$ [Breiman *et al.* , 1984]. It is the sum over the $J-1$ internal nodes for each tree. At each node t, one of variable in X_ℓ is used to split regions associated with a specific node. The chosen variable is the one that gives maximal improvement in the estimated $\hat{\imath}_t$ in squared error risk compared to a constant fit over the region. We define the squared relative importance of variable X_ℓ as the sum of that squared improvement for all internal nodes chosen by the gradient boosting algorithm as splitting variables. [Hastie *et al.* , 2001]

Figure 7: Variable Importance For interactive version in browser: `https://plotly.com/~Emborg/243/#/`



4.1.4 LSTM

Similarly to the preceding models, we also conduct a grid search for optimizing the forecasts obtained from the LSTM specification, predicting a months worth of one-day ahead forecasts for each specification chosen. For the LSTM, this entails choosing the optimal values for the number of neurons, the dropout rate, the batch size, and the number of epochs. The number of layers has been fixed to two layers, in order to enable the network to handle non-linearities while still keeping the computing time at a reasonable amount.

The number of neurons in the network impacts the learning capacity of the LSTM. As the amount of neurons increases, the LSTM can learn more structure from the data; however, this results in increased training times and increases the risk of overfitting to the training data. We

have chosen to search in the space of $\{2, 4, 6\}$ neurons. Next, we allow the model to include a dropout layer, the purpose of which is to slow down the learning. The dropout layer helps against the risk of overfitting, as it ignores randomly selected neurons during training, thus reducing sensitivity to specific weights of individual neurons. Our model is set to choose a dropout rate of either 0.0 or 0.2, as to allow it not to include any dropout, or to include a dropout rate of 20%, which is often used as a compromise between retaining model accuracy whilst keeping overfitting at bay[3]. The batch size of the network is how often the weights of the network are updated. For this parameter, we let the model choose between a batch size of either 7 or 14, to set the updating of weights to occur at a weekly or a bi-weekly rate. The final parameter to be specified is the number of epochs for training. The amount of epochs defines the number of times the network will run through the training set, each time adjusting the model weights. We have set the model to choose either 100 or 200 epochs. Choosing between each parameter configuration then implies that the model has to compute $\#neurons \cdot \#dropout\ rates \cdot \#batch\ sizes \cdot \#epochs = 3 \cdot 2 \cdot 2 \cdot 2 = 24\ models$.

As with the method used for the Gradient Boosting a forward cross-validation approach is used to specify the optimal parameters. However, for the LSTM, we are computationally restricted to only conducting a single validation for each specification, unlike the 3-step forward cross-validation. This then entails training each of the 24 models on the first eleven months of the previous year and using the last month of the preceding year as validation data. The parameter specification of the model with the lowest validation error is chosen. This model specification trains on the entire data set from the previous year. This procedure is redone each month for the remainder of the data set.

4.2 Model evaluation

The following section evaluates the performance of each of the forecasting techniques. We compare the performance when including variables for investor sentiment, and when excluding them, to determine whether the specifications were successful in extracting predictive power out of the measures of investor sentiment or if investor sentiment, in fact, does not improve the individual forecasts. Each forecast will have an accompanying error plot and a cumulative squared error plot, in which they are compared to a benchmark model of an AR\left(1\right) specification. In the form of RMSE and MAE, error metrics are shown for all models in Table 2 below. The two forecasts from each respective technique will is with a Diebold-Mariano, DM, test for superior predictive ability [Diebold & Mariano, 1995]. The problem of using a Diebold-Mariano test on potentially nested forecasts is avoided as forecasts are conducted using a rolling window [Giacomini & White, 2006]. A plot of the actual price and the developed forecasts is seen in 8; we recommend you use the in-browser version using the link in the figure for an easier overview of the forecasts.

[3]https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046

Table 2: Error metrics for each forecast

Model	RMSE	MAE
$AR(1)$	698.75	528.16
Models including sentiment		
$ARIMAX$	303.51	189.20
VAR	289.97	179.33
XGB	384.60	241.68
$LSTM$	285.86	170.49
Models excluding sentiment		
$ARIMAX$	292.75	173.60
VAR	289.51	176.67
XGB	381.38	239.69
$LSTM$	287.68	170.06

Figure 8: Plot of actual price and forecasts. For interactive in browser version: `https://plotly.com/~Emborg/647/`

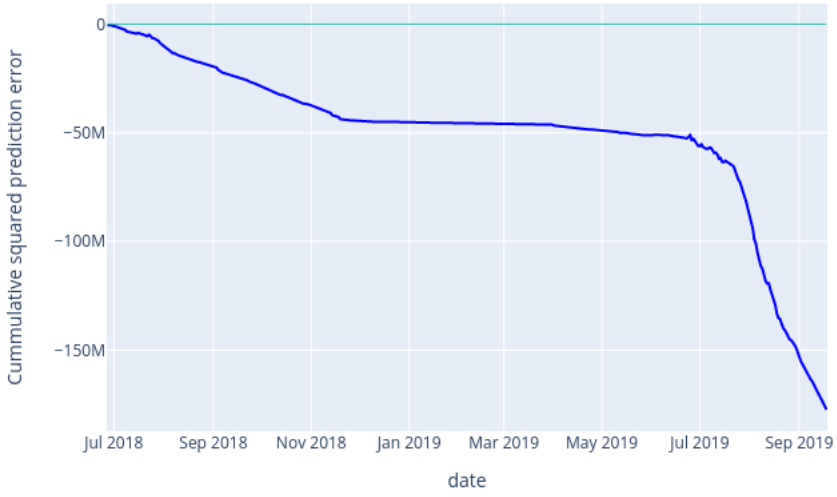
4.2.1 ARIMA

As the initial empirical result, we run ARIMAX forecasting both including the amount of positive, neutral and negative comments, the sentiment, as exogenous variables, and a specification excluding these, in order to analyze if the ARIMAX model succeeds in utilizing the investor sentiment to better forecast future Bitcoin prices.

In the initial specification, we include the sentiment variables. The resulting root mean squared error is then; $RMSE = 303.51$, and the mean absolute error is; $MAE = 189.20$. The cumulative

squared prediction error, as compared to our benchmark model, an $AR(1)$, reveals that this specification performs a lot better than a simple single lag auto-regression.

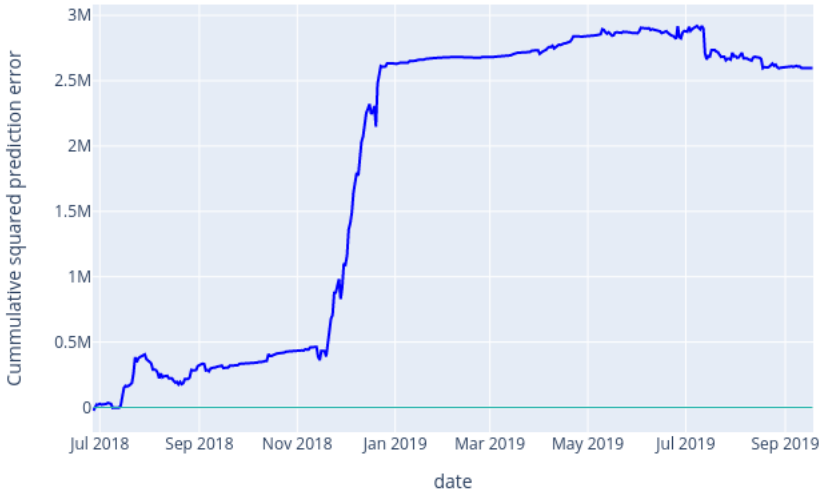
Figure 9: CSPE ARIMAX with sentiment compared to AR(1)
Upward movement of the CSPE line corresponds to lower squared forecasting error of the AR(1) and vice versa.



Excluding the sentiment exogenous variables in the ARIMAX comes with some interesting results, in the sense that the RMSE is actually reduced by excluding these variables. The resulting error metrics are in this case; $RMSE = 292.75$ and $MAE = 173.60$, thus lower than the previous specification.

Comparing the two forecasts by their cumulative squared predictive errors shows that the ARIMAX specification excluding sentiment performs better than its counter on almost all of the forecasting period, with few exceptions.

Figure 10: CSPE between ARIMAX specifications
Upward movement of the CSPE line corresponds to lower squared forecasting error of the $ARIMAX_{NS}$ and vice versa.



What especially stands out is the explosive behavior in December 2018, where the CSPE displays the better performance of the excluding ARIMAX forecast. This specific period can be seen in Figure 11, where it is clear that the $ARIMAX_{NS}$ follows the price better. This period is at the end of the first sizeable downward trajectory seen in the forecasting period, and it seems the sentiment including model is too willing to increase again. However, when again looking at their CSPE comparison, towards the end, the sentiment including model seems to outperform its counter, especially around the first half of July 2019, an occurrence that persists through all the models.

Figure 11: Price and ARIMA forecasts December 2018

To examine whether the ARIMAX specification excluding sentiment variables has a statistically superior predictive ability compared to the initial model, we run a DM test. With a $p - value = 0.0012$. We thereby reject the null hypothesis that the two forecasts have equal predictive power. This implies that the forecast without the sentiment, in fact, is statistically better than the ARIMAX model including the investor sentiment measures.

4.2.2 VAR

For the second econometric forecasts, we observe the same pattern in the CSPE compared to the benchmark. There is no case in time where the VAR model with sentiment data is outperformed by the $AR(1)$, as seen in Figure 12. It achieves an overall RMSE of 289.97. This is marginally higher than the specification without which ends at a RMSE of 289.51. This mirrors the result of the ARIMAX model. Figure 13 depicts the CSPE of the VAR models. Here a good performance by the VAR including sentiment compared to VAR without can be seen by a downwards movement in the graph. It goes back and forth with substantial spikes in both ways at the beginning of 2019 and ending June of 2019. In the latter period, the specification with sentiment performs better for the downward spike in price. Oppositely we observe that models without sentiment are better at forecasting the upward spike. We observe no further patterns for when one performs better than the other. Statistically, the DM test reveals that there is no difference in forecasting performance in this sample between the two models with a $p - value = 0.8646$.

Figure 12: CSPE VAR with sentiment compared to AR(1)
Upward movement of the CSPE line corresponds to lower squared forecasting error of the AR(1) and vice versa.

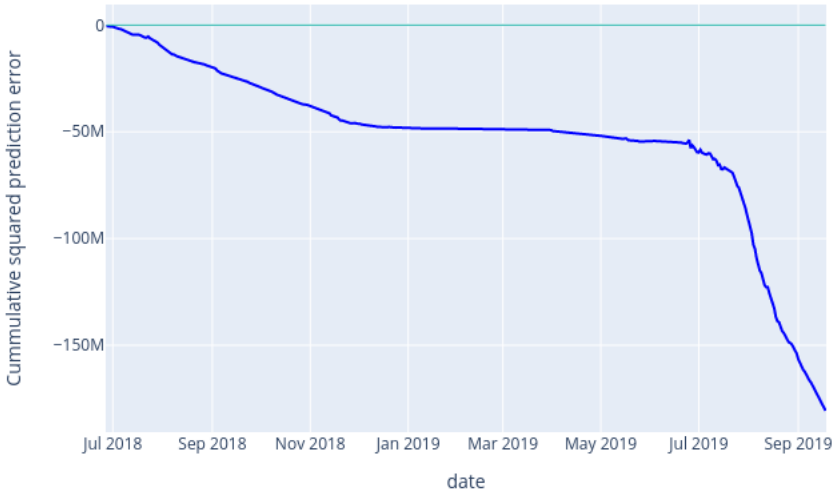


Figure 13: CSPE between VAR specifications
Upward movement of the CSPE line corresponds to lower squared forecasting error of the VAR_{NS} and vice versa.

4.2.3 XGB

The first machine learning model also outperforms the benchmark model, as seen in Figure 14. The gradient boosting with sentiment features results in a RMSE of 384.60, and the same story as the two previous models when excluding sentiment as estimation without sentiment obtains a RMSE of 381.38. The DM test does not result in a difference in forecasting ability for the sample with a $p - value = 0.7308$. Figure 15 reveals a very low difference in model performance for the first three-quarters of the test set. This is likely due to the algorithm being able to select relevant features and come up with identical models. The CPSE graph shows a similar event as for the VAR models from the end of June to mid-July. From the 9th of June to 25th, the Bitcoin price increases as it close to doubles from $8,000$ USD to $13,000$ USD, and as seen in the CSPE, the upwards movement indicates superior performance by the non-sentiment model. The subsequent decrease from the 8th of July to the 16th unveil that the configuration with sentiment outperforms its counterpart in downward trending periods.

Figure 14: CSPE for Gradient Boosting model compared to $AR(1)$
Upward movement of the CSPE line corresponds to lower squared forecasting error of the AR(1) and vice versa.

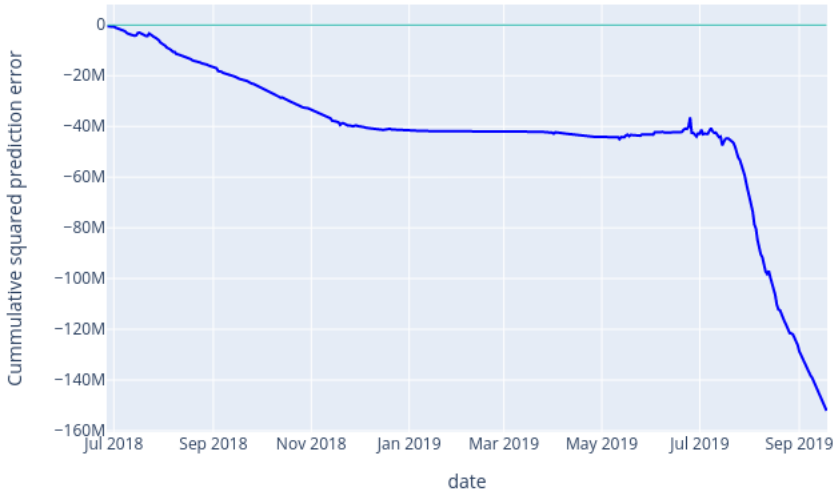
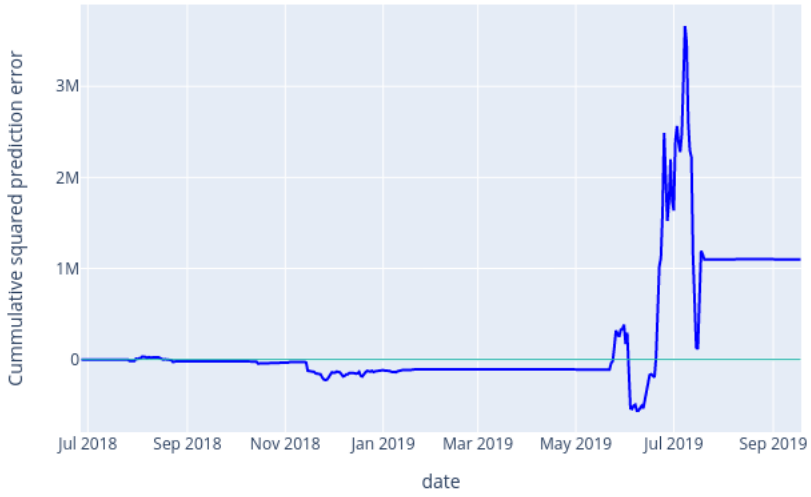


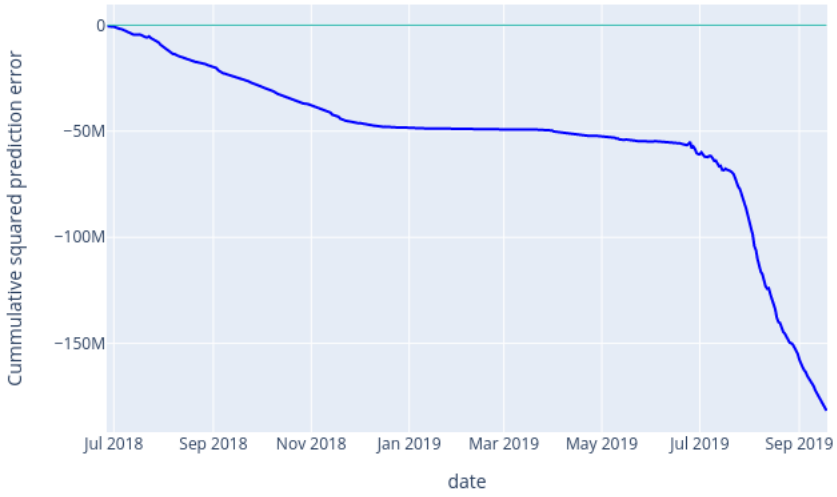
Figure 15: CSPE comparison between Gradient Boosting with and without sentiment
Upward movement of the CSPE line corresponds to lower squared forecasting error of the XGB_{NS} and vice versa.



4.2.4 LSTM

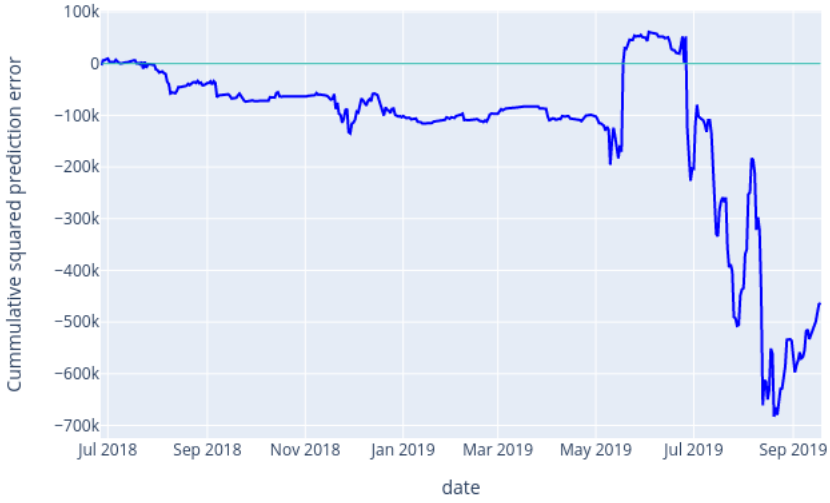
As the final model, we run the LSTM with and without the sentiment variables as inputs. The model obtains a $RMSE = 285.86$ and $MAE = 170.49$; the lowest error metrics attained for the sample. The cumulative squared prediction error when compared to the benchmark $AR(1)$ model, is depicted in Figure 16 and follows the same pattern as the previous models.

Figure 16: CSPE LSTM including sentiment compared to $AR(1)$
Upward movement of the CSPE line corresponds to lower squared forecasting error of the AR(1) and vice versa.



When forecasting without the sentiment variables $RMSE = 287.68$ increases, however, we see a decrease in $MAE = 170.06$. This implies that the forecast excluding sentiment must have one or more large deviations when compared to the forecast including sentiment, as the excluding model is punished in the RMSE. The cumulative squared prediction error when comparing the LSTM models is seen in Figure17. This graph displays that the two models follow along for most of the forecasting period, but with a downward trend favoring the sentiment including model. What is interesting about this depiction, however, is that all the upward spikes in the plot occur on the same dates as the substantial price increases seen in the price plot, Figure 8, and likewise for the next downward spikes. This implies that the model excluding measures for investor sentiment performs well when the price is increasing, and vice versa for the model including investor sentiment. The errors incurred when failing to predict the downward trajectory as closely as the sentiment including model, causes the shift in the error metrics alluded to earlier, as the RMSE punishes this harder than the MAE error metric.

Figure 17: CSPE comapared between LSTM specifications
Upward movement of the CSPE line corresponds to lower squared forecasting error of the $LSTM_{NS}$ and vice versa.



Worth noting about the above graph is the values on the $y - axis$, implying that the potential superiority of the sentiment including model, may not be as prevalent as the curve implies. To test the difference in the two forecasts, we once again run a DM test. This fails to reject the null of equal predictive power, with $p - value = 0.4118$, and thus we conclude that neither of our models display superior predictive ability compared to each other.

Thus none of the tested models have shown superior predictive ability once we include measures for investor sentiment, as we sought out to examine. However, it is fascinating that it seems these measures help prediction during price drops. The following section will cross-compare the above models and examine whether we can obtain the best of both worlds of sentiment by creating forecast combinations.

4.2.5 Cross evaluation of the models

We continue to test the models across methods taking the best model of each type based on the error metrics for this sample. The DM test statistics and appertaining p-values for all combinations are found in A.2. As also shown above, all models outperformed the benchmark of $AR(1)$ in regards to the error metrics in 2 and deemed to be statistically better by the DM-test in A.2 at all common levels of significance.

Focusing on the econometric models with no sentiment, both their error metrics are ambiguous, as to which could be performing better. The VAR has a lower RMSE, but ARIMAX has the smallest MAE. The DM test reveals no statistically significant difference in forecasting ability

between the two, with a $p - value = 0.5047$. Thus, we can infer no superior relationship in predicting Bitcoin's price between the best performing econometric models. When comparing the machine learning techniques, the better performing XGB is deemed to be the one excluding sentiment. No clear relationship can be inferred between the two LSTM specifications, using error metrics. The XGB specification is found to be statistically inferior to both LSTM specifications using a DM test, with a $p - value = 0.0000$ in both cases. We thereby deem the LSTM procedure to be the better of the machine learning techniques utilized on our sample. XGB is a clear loser relative to all other models but the benchmark. A reason for this could be that XGB is not specifically designed for time series forecasting, as the other are.

This leads to the comparison between the econometric models and machine learning techniques. Whilst no superior relationship could be determined between the two LSTM specifications, the specification, including sentiment measures, is found to be statistically superior, on a 5% significance level, to the $ARIMAX_{NS}$ model, attaining a $p - value = 0.0328$ in the DM test. This relationship is not seen when conducting the DM test between the sentiment excluding LSTM and the $ARIMAX_{NS}$, as we can not reject the null hypothesis of equal predictive power with a $p - value = 0.1479$. Thus, as also found in the literature, we find the LSTM model, including sentiment, outperforms both of our ARIMAX specifications. No superior model emerges when testing against the VAR_{NS} model. The evidence fails to determine a preferred model in this study unanimously.

To verify the above result, we further examine performance by using the model confidence set method by Hansen, Lunde and Nason [Hansen *et al.* , 2011]. By iteratively testing predictive ability under the null hypothesis of equal performance. If the null is rejected, inferior models are removed from the confidence set. This decision rule is repeated until a set level of confidence is reached. We use a confidence level of 10%. The model confidence set contains the ARIMAX without sentiment and both LSTM with and without. With corresponding p-values of $[0.333, 0.333, 1]$. According to the interpretation of a low p-value, implying a given forecast model is unlikely to belong to the set of superior models.

4.2.6 Forecast combination

As motivated in the introduction to this paper, the Bitcoin price and our distinct models set the stage for a study into the effects of forecast combination. This section seeks to answer this research question by evaluating the combination schemes covered in the methodology section.

The next step is to figure out which models should be used as we have decided on weighting schemes. Ex-ante, we decide on three sets of models based on our study, one with every model, one with all containing sentiment parameters and one without. The resulting error metrics can be seen in Table 3. Compared to the best individual model, the LSTM, including sentiment data, which obtained an RMSE of 285.86, the best performing combination, Rank, based on all models

obtains a slightly lower RMSE of 285.07 for the test period. Worth noting is that the ensemble with sentiment outperforms the one without, even though that the previous between model comparison showed that three out of the four best models, according to the error metrics, were the ones without sentiment. As seen previously in the comparison between the LSTM models, the MAE error metric implies the reverse relationship, indicating that there has been a larger deviation in the combined forecast, which is punished harder by the RMSE error metric.

In general, all forecast combination schemes have resulted in low error metrics relative to the individual models, apart from the combination using a one-period MSE rolling window scheme, supporting the philosophy of combining linear and non-linear models in forecasting. Nevertheless, we again underline that we fail to pinpoint a single best method. We choose to examine the relationship between the individual forecast and the combined forecast that attains the lowest RMSE. This is the sentiment including LSTM model and the ranked combination forecast using all forecasts. The resulting DM test has a $p - value = 0.9911$, and we conclude that there has been no significant improvement.

Curiously we do not experience the "forecast combination puzzle" as the simple equal-weighted forecast does not achieve the lowest error metrics. Also, the time dynamic schemes do not outperform MSE and Rank weighting schemes. Their error metrics imply no gain in forecasting accuracy when weighing according to neither the MSEs of the previous day, nor the previous week, as compared to weighting according to the entire past of MSEs.

Table 3: Error metrics for forecast combinations conditional on weighting scheme and included models

RMSE	Equal	MSE	Rank	Time(1)	Time(7)
All	288.85	286.30	285.07	306.95	287.25
With sentiment	289.69	286.80	285.08	302.49	287.82
Without sentiment	288.97	286.58	285.82	302.47	287.52

MAE	Equal	MSE	Rank	Time(1)	Time(7)
All	171.44	169.31	169.23	183.01	170.52
With sentiment	173.54	171.17	170.12	182.93	171.63
Without sentiment	170.60	168.68	168.50	179.26	170.04

We conclude that whilst almost all the combination techniques display relatively low error metrics, they do not result in a statistically preferable forecast for the Bitcoin price.

5 Limitations and Extensions

The analysis conducted in this paper looks into the idea of incorporating measures for investor sentiment in order to improve Bitcoin forecasting by scraping Reddit.com for user comments and deriving sentiment from these. As the empirical results found in this paper fail to significantly improve forecasts by incorporating such measures for investor sentiment, it is essential to mention some of the limitations encountered when conducting this analysis, and highlight potential extensions to this field spurred on by these.

One of the main limitations of this paper is the use of only a single way of measuring sentiment from the extracted comments. As explained in the Data section, we have utilized the VADER lexicon, modified with words especially relevant for the Bitcoin field, to determine the sentiment of each comment. Having obtained our results, it could be interesting to see whether there could be improved performance if we were to use another type of Natural Language Processing, such as the Python TextBlob or Pattern package, which have revealed significant results in other papers [Galeshchuk *et al.* , 2018, Jain *et al.* , 2018a]. Further, our results show that the LSTM model was closest to improving its forecast by incorporating investor sentiment, in the sense that this was the only case where we experienced a decrease in the RMSE when incorporating sentiment. However, it was insignificant in the Diebold-Mariano test. However, we were computationally restricted in the parameterization of this machine learning technique, as the tuning involved was considerably less rigorous than the Gradient Boosting. Perhaps better performance could have been obtained if we had been able to enlarge the searching space, and applying the same type of 3-fold forward cross-validation as used in Gradient Boosting. Further, we did not look into whether the data exhibited signs of herding or overconfidence by the investors, both of which have been shown to cause pricing instability with excess volatility, miss-pricing, bubble formation, trading volume, and market crashes [Vidal-Tomas *et al.* , 2019], [Bouri *et al.* , 2019], [Murray Leclair, 2018]. The Bitcoin market has been shown to have symptoms of these characteristics, and the highly volatile period seen in 2018 has proven to be very similar to the psychologically-based market cycle seen during the Dot-Com bubble [Tran, 2019]. Extensions to the work done in this paper could entail the handling of these limitations.

Additional extensions could entail looking further into the concept of sentiment, in the sense of analyzing whether investor sentiment is perhaps more effectful on other granularity definitions. This could be including lags from more than just the previous day of comments or setting up constructs for the evolution of the sentiment, such as the percentage of positive comments as compared to the average of the previous week. In general, more rigorous testing of the optimal way to utilize this new type of data. This could also entail a more specific examination of the positive versus the negative comments' usefulness to see if there is a quantifiable difference in their effect. A combination of sentiment from different platforms could also be an interesting extension, as platforms such as Twitter and Google have also proven useful in this context

[Galeshchuk *et al.* , 2018, Kristoufek, 2013]. Lastly, of course, an interesting extension could be a further look into the unexpected side result of this paper, wherein our CSPE plots seemed to display the stylized fact that models including sentiment performed better than their counterparts when forecasting downturns in the Bitcoin price.

6 Conclusion

This paper studies the effect of including measures of social media sentiment when forecasting the Bitcoin price. In conclusion, we find no significant effect of including sentiment when using different model frameworks, by evaluating the prediction error—thereby rejecting the value of including social media sentiment from Reddit comments for our sample. In addition, we sought to examine the effect of combining forecasts to increase prediction ability. The results exhibited do not show significant improvement using any of the weighting schemes tested. Though we attain insignificant results, further study of sentiment for price forecasting is advised as the best performing model, in terms of error metrics, was based on a machine learning technique using sentiment data. Additionally, we found signs of sentiment-based models performing better during price decreases.

References

[Aiolfi & Timmermann, 2006] Aiolfi, Marco, & Timmermann, Allan. 2006. Persistence in forecasting performance and conditional combination strategies. *Journal of Econometrics*, **135**(1-2), 31–53.

[Alessandretti *et al.* , 2018] Alessandretti, Laura, ElBahrawy, Abeer, Aiello, Luca Maria, & Baronchelli, Andrea. 2018. Anticipating cryptocurrency prices using machine learning. *Complexity*, **2018**.

[Baker & Wurgler, 2000] Baker, Malcolm, & Wurgler, Jeffrey. 2000. The equity share in new issues and aggregate stock returns. *the Journal of Finance*, **55**(5), 2219–2257.

[Baker & Wurgler, 2006] Baker, Malcolm, & Wurgler, Jeffrey. 2006. Investor sentiment and the crossâsection of stock returns. *The journal of Finance*, **61**(4), 1645–1680.

[Balcilar *et al.* , 2017] Balcilar, Mehmet, Bouri, Elie, Gupta, Rangan, & Roubaud, David. 2017. Can volume predict Bitcoin returns and volatility? A quantiles-based approach. *Economic Modelling*, **64**, 74–81.

[Bates & Granger, 1969] Bates, John M., & Granger, Clive W. J. 1969. The combination of forecasts. *Journal of the Operational Research Society*, **20**(4), 451–468.

[Bengio *et al.* , 1994] Bengio, Yoshua, Simard, Patrice, & Frasconi, Paolo. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, **5**(2), 157–166.

[Blau, 2017] Blau, Benjamin M. 2017. Price dynamics and speculative trading in bitcoin. *Research in International Business and Finance*, **41**, 493–499.

[Bouoiyour & Selmi, 2015] Bouoiyour, Jamal, & Selmi, Refk. 2015. What does Bitcoin look like? *Annals of Economics and Finance*, **16**(2), 449–492.

[Bouri *et al.* , 2019] Bouri, Elie, Gupta, Rangan, & Roubaud, David. 2019. Herding behaviour in cryptocurrencies. *Finance Research Letters*, **29**, 216–221.

[Box *et al.* , 2015] Box, George E. P., Jenkins, Gwilym M., Reinsel, Gregory C., & Ljung, Greta M. 2015. *Time series analysis: forecasting and control*. John Wiley Sons.

[Breiman, 1996] Breiman, Leo. 1996. *Bias, variance, and arcing classifiers*. Tech. rept.

[Breiman *et al.* , 1984] Breiman, Leo, Friedman, Jerome, Stone, Charles J., & Olshen, Richard A. 1984. *Classification and regression trees*. CRC press.

[Campbell & Shiller, 1988] Campbell, John Y., & Shiller, Robert J. 1988. Stock prices, earnings, and expected dividends. *The Journal of Finance*, **43**(3), 661–676.

[Catania *et al.* , 2019] Catania, Leopoldo, Grassi, Stefano, & Ravazzolo, Francesco. 2019. Forecasting cryptocurrencies under model and parameter instability. *International Journal of Forecasting*, **35**(2), 485–501.

[Cheah & Fry, 2015] Cheah, Eng-Tuck, & Fry, John. 2015. Speculative bubbles in Bitcoin markets? An empirical investigation into the fundamental value of Bitcoin. *Economics Letters*, **130**, 32–36.

[Cho *et al.* , 2014] Cho, Kyunghyun, Van MerriÃ«nboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, & Bengio, Yoshua. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Ciaian *et al.* , 2016] Ciaian, Pavel, Rajcaniova, Miroslava, & Kancs, dâArtis. 2016. The economics of BitCoin price formation. *Applied Economics*, **48**(19), 1799–1815.

[Clemen, 1989] Clemen, Robert T. 1989. Combining forecasts: A review and annotated bibliography. *International journal of forecasting*, **5**(4), 559–583.

[Colah, 2015] Colah. 2015.

[Derbentsev *et al.* , 2019] Derbentsev, Vasily, Datsenko, Natalia, Stepanenko, Olga, & Bezkorovainyi, Vitaly. 2019. Forecasting cryptocurrency prices time series using machine learning approach. vol. 65. EDP Sciences.

[Dickey & Fuller, 1979] Dickey, David A., & Fuller, Wayne A. 1979. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, **74**(366a), 427–431.

[Diebold & Mariano, 1995] Diebold, Francis X., & Mariano, Roberto S. 1995. Comparing Predictive Accuracy. *Journal of Business Economic Statistics*, **13**(3), 253–263.

[Friedman, 2001] Friedman, Jerome H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

[Galeshchuk *et al.* , 2018] Galeshchuk, Svitlana, Vasylchyshyn, Oleksandra, & Krysovatyy, Andriy. 2018. Bitcoin Response to Twitter Sentiments.

[Gers & Schmidhuber, 2000] Gers, Felix A., & Schmidhuber, JÃŒrgen. 2000. Recurrent nets that time and count. vol. 3. IEEE.

[Giacomini & White, 2006] Giacomini, Raffaella, & White, Halbert. 2006. Tests of conditional predictive ability. *Econometrica*, **74**(6), 1545–1578.

[Hansen *et al.* , 2011] Hansen, Peter R., Lunde, Asger, & Nason, James M. 2011. The model confidence set. *Econometrica*, **79**(2), 453–497.

[Hastie *et al.* , 2001] Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

[Hochreiter & Schmidhuber, 1997] Hochreiter, Sepp, & Schmidhuber, JÃŒrgen. 1997. Long short-term memory. *Neural computation*, **9**(8), 1735–1780.

[Huisu *et al.* , 2018] Huisu, Jang, Lee, Jaewook, Ko, Hyungjin, & Lee, Woojin. 2018. Predicting bitcoin prices by using rolling window lstm model.

[Hutto & Gilbert, 2015] Hutto, C. J., & Gilbert, Eric. 2015. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text.*

[Jain *et al.* , 2018a] Jain, A., Tripathi, S., Dwivedi, H. D., & Saxena, P. 2018a. Forecasting Price of Cryptocurrencies Using Tweets Sentiment Analysis. *Pages 1–7 of: 2018 Eleventh International Conference on Contemporary Computing (IC3).*

[Jain *et al.* , 2018b] Jain, Rishanki, Nguyen, Rosie, Tang, Linyi, & Miller, Travis. 2018b. Bitcoin Price Forecasting using Web Search and Social Media Data.

[Karakoyun & Cibikdiken, 2018] Karakoyun, E. S., & Cibikdiken, A. O. 2018. Comparison of arima time series model and lstm deep learning algorithm for bitcoin price forecasting.

[Kim *et al.* , 2016] Kim, Young Bin, Kim, Jun Gi, Kim, Wook, Im, Jae Ho, Kim, Tae Hyeong, Kang, Shin Jin, & Kim, Chang Hun. 2016. Predicting fluctuations in cryptocurrency transactions based on user comments and replies. *PloS one*, **11**(8).

[Kristoufek, 2013] Kristoufek, Ladislav. 2013. BitCoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the Internet era. *Scientific Reports*, **3**(1), 3415.

[Kristoufek, 2015] Kristoufek, Ladislav. 2015. What are the main drivers of the Bitcoin price? Evidence from wavelet coherence analysis. *PloS one*, **10**(4).

[Kwiatkowski *et al.* , 1992] Kwiatkowski, Denis, Phillips, Peter C. B., Schmidt, Peter, & Shin, Yongcheol. 1992. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of econometrics*, **54**(1-3), 159–178.

[Li *et al.* , 2019] Li, Tianyu Ray, Chamrajnagar, Anup, Fong, Xander, Rizik, Nicholas, & Fu, Feng. 2019. Sentiment-based prediction of alternative cryptocurrency price fluctuations using gradient boosting tree model. *Frontiers in Physics*, **7**, 98.

[Murray Leclair, 2018] Murray Leclair, Emmanuel. 2018. *Herding in the cryptocurrency market.*

[Neal & Wheatley, 1998] Neal, Robert, & Wheatley, Simon. 1998. Do Measures of Investor Sentiment Predict Returns? *Journal of Financial and Quantitative Analysis*, **33**(Dec.), 523–547.

[Phillips & Gorse, 2017] Phillips, Ross C., & Gorse, Denise. 2017. Predicting cryptocurrency price bubbles using social media data and epidemic modelling. IEEE.

[Rebane *et al.* , 2018] Rebane, Jonathan, Karlsson, Isak, Denic, Stojan, & Papapetrou, Panagiotis. 2018. Seq2Seq RNNs and ARIMA models for cryptocurrency prediction: A comparative study. *SIGKDD Fintech*, **18**.

[Schapire, 1990] Schapire, Robert E. 1990. The strength of weak learnability. *Machine learning*, **5**(2), 197–227.

[Selmi *et al.* , 2018] Selmi, Refk, Tiwari, Aviral Kumar, & Hammoudeh, Shawkat. 2018. Efficiency or speculation? A dynamic analysis of the Bitcoin market. *Economics Bulletin*, **38**(4), 2037–2046.

[Shiller, 2000] Shiller, Robert C. 2000. Irrational exuberance. *Philosophy and Public Policy Quarterly*, **20**(1), 18–23.

[Shuyo, 2010] Shuyo, Nakatani. 2010. *Language Detection Library for Java.*

[Silva, 2014] Silva, Lucas. 2014. A feature engineering approach to wind power forecasting: GEFCom 2012. *International Journal of Forecasting*, **30**(2), 395–401.

[Sims, 1980] Sims, Christopher A. 1980. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, 1–48.

[Stiglitz & Rogoff, 2018] Stiglitz, Roubini, & Rogoff. 2018. Top Economists Stiglitz, Roubini and Rogoff Renew Bitcoin Doom Scenarios. *CNBC*. Accessed 15 Feb 2020.

[Stock & W Watson, 2003] Stock, James H., & W Watson, Mark. 2003. Forecasting output and inflation: The role of asset prices. *Journal of Economic Literature*, **41**(3), 788–829.

[Stock & Watson, 1999] Stock, James H., & Watson, Mark W. 1999. Forecasting inflation. *Journal of Monetary Economics*, **44**(2), 293–335.

[Stock & Watson, 2004] Stock, James H., & Watson, Mark W. 2004. Combination forecasts of output growth in a sevenâcountry data set. *Journal of forecasting*, **23**(6), 405–430.

[Tran, 2019] Tran, Hai Yen. 2019. Overconfidence Test in Cryptocurrencies Markets Using VAR Analysis. *Available at SSRN 3416394*.

[Vidal-Tomas *et al.* , 2019] Vidal-Tomas, David, Ibanez, Ana M., & Farinos, Jose E. 2019. Herding in the cryptocurrency market: CSSD and CSAD approaches. *Finance Research Letters*, **30**, 181–186.

Note: The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root, with the alternative that there is no unit root. If the p-value is above a critical size, then we cannot reject that there is a unit root.
The p-values are obtained through regression surface approximation from MacKinnon 1994, but using the updated 2010 tables. If the p-value is close to significant, then the critical values should be used to judge whether to reject the null.
KPSS: The p-values are interpolated from Table 1 of Kwiatkowski et al. (1992).

Table A.1: Test results of Augmented Dicky-Fuller on considered variables

Variable	ADF		KPSS	
Levels	t-staistic	p-value	t-staistic	p-value
Price	-1.52	0.52	1.24	0.01
Volatility	−7.04	0.00	0.31	0.01
Volume Price	−0.94	0.77	1.33	0.01
Volume Number	−0.67	0.86	1.55	0.01
Positive Comment	−3.24	0.02	0.23	0.01
Neutral Comment	−3.21	0.02	0.23	0.01
Negative Comment	−3.17	0.02	0.25	0.01
Differenced	t-staistic	p-value	t-staistic	p-value
Price	−10.97	0.00	0.02	0.10
Volatility	−16.34	0.00	0.04	0.10
Volume Price	−14.75	0.00	0.02	0.10
Volume Number	−13.71	0.00	0.02	0.10
Positive Comment	−11.73	0.00	0.02	0.10
Neutral Comment	−11.24	0.00	0.02	0.10
Negative Comment	−11.79	0.00	0.02	0.10

Figure A.1: Lag structure of ARIMAX models

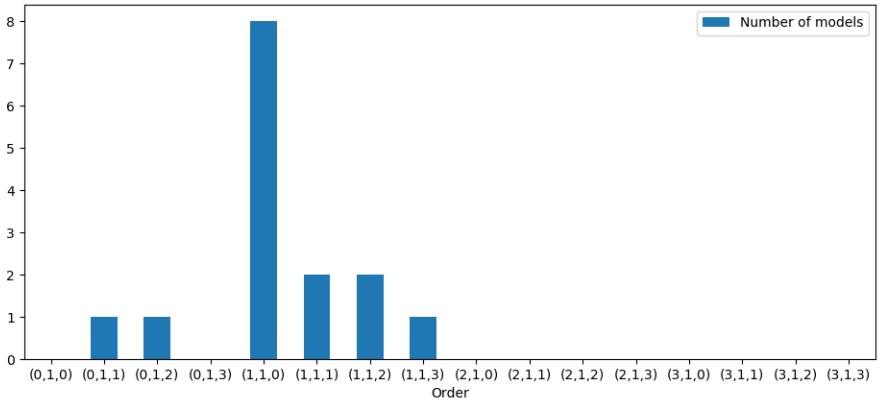


Table A.2: Diebold-Mariano tests

Model 1	Model 2	$t-statistic$	$p-value$
$AR(1)$	$ARIMAX$	-8.54	0.0000
$AR(1)$	$ARIMAX_{NS}$	-8.73	0.0000
$AR(1)$	VAR	-8.68	0.0000
$AR(1)$	VAR_{NS}	-8.70	0.0000
$AR(1)$	XGB	-7.15	0.0000
$AR(1)$	XGB_{NS}	-7.38	0.0000
$AR(1)$	$LSTM$	-8.76	0.0000
$AR(1)$	$LSTM_{NS}$	-8.73	0.0000
$ARIMAX$	$ARIMAX_{NS}$	-3.24	0.0012
$ARIMAX$	VAR	-1.96	0.0505
$ARIMAX$	VAR_{NS}	-2.17	0.0304
$ARIMAX$	XGB	3.97	0.0001
$ARIMAX$	XGB_{NS}	4.03	0.0001
$ARIMAX$	$LSTM$	-3.64	0.0003
$ARIMAX$	$LSTM_{NS}$	-3.00	0.0027
$ARIMAX_{NS}$	VAR	-0.49	0.6242
$ARIMAX_{NS}$	VAR_{NS}	-0.67	0.5047
$ARIMAX_{NS}$	XGB	4.42	0.0000
$ARIMAX_{NS}$	XGB_{NS}	4.56	0.0000
$ARIMAX_{NS}$	$LSTM$	-2.14	0.0328
$ARIMAX_{NS}$	$LSTM_{NS}$	-1.45	0.1479
VAR	VAR_{NS}	-0.17	0.8646
VAR	XGB	4.74	0.0000
VAR	XGB_{NS}	4.64	0.0000
VAR	$LSTM$	-0.84	0.4017
VAR	$LSTM_{NS}$	-0.46	0.6446
VAR_{NS}	XGB	4.78	0.0000
VAR_{NS}	XGB_{NS}	4.81	0.0000
VAR_{NS}	$LSTM$	-0.89	0.3730
VAR_{NS}	$LSTM_{NS}$	-0.44	0.6586
XGB	XGB_{NS}	-0.34	0.7308
XGB	$LSTM$	-4.80	0.0000
XGB	$LSTM_{NS}$	-4.71	0.0000
XGB_{NS}	$LSTM$	-4.93	0.0000
XGB_{NS}	$LSTM_{NS}$	-4.89	0.0000
$LSTM$	$LSTM_{NS}$	0.82	0.4118