Comparative study of Bitcoin price prediction using WaveNets, Recurrent Neural Networks and other Machine Learning Methods

1st Leonardo Felizardo
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
leonardo.felizardo@usp.br

2nd Roberth Oliveira
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
roberth.oliveira@usp.br

3nd Emilio Del-Moral-Hernandez
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
emilio.delmoral@usp.br

4nd Fabio Cozman
Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
fgcozman@usp.br

***Abstract*—Forecasting time series data is an important subject in economics, business, and finance. Traditionally, there are several techniques such as univariate Autoregressive (AR), univariate Moving Average (MA), Simple Exponential Smoothing (SES), and more notably Autoregressive Integrated Moving Average (ARIMA) with their many variations that can effectively forecast. However, with the recent advancement in the computational capacity of computers and more importantly developing more advanced machine learning algorithms and approaches such as deep learning, new algorithms have been developed to forecast time series data. This article compares different methodologies such as ARIMA, Random Forest (RF), Support Vector Machine (SVM), Long Short-Term Memory (LSTM) and WaveNets for estimating the future price of Bitcoin.**

***Index Terms*—Convolution Neural Networks, Recurrent Neural Networks, ARIMA, Time Series, Bitcoin**

I. Introduction

Inspired by recent studies such as Saad, Prokhorov and Wunsch [1]; Jou-fan Chen, Wei-lun Chen and Huang [2]; Selvin, Vinayakumar and Gopalakrishnan [3]; Gamboa [4], this paper aims to present a comparative study of neural network architectures - Long-Short Term Memory (LSTM), WaveNet, support vector machine (SVM) Random Forest (RF) and Auto regressive integrative moving average (ARIMA) to predict future prices in the Bitcoin time series. Bitcoin price has a high volatility and due Efficient Market Hypothesis, is theoretically difficult to predict. Models tend to perform badly since the future prices can be aleatory. We verify the results presented by many authors and explore the efficacy of the models to predict the Bitcoin price. We realized that, for this kind of time-series, linear models tend to perform better then recent machine learning models.

When dealing with machine learning models with a large number of hyper-parameters combination, the comparability between two models can be difficult because the search for hyper-parameters should be even. We propose a method to avoid this problem when testing the models and show how the choice of hyper-parameters can influence the results. We used the same technique to find hyper-parameters for all the models, thus eliminating subjective aspects of the hyper-parameter choices. Models are compared by a $t-student$ test to identify, which is the best statistically.

CAPES

II. Literature Review

A. Autoregressive Integrated Moving Average

The Box-Jenkins methodology refers to a set of procedures for identifying and estimating of time series models within the class of autoregressive integrated moving average (ARIMA) models. ARIMA models are a class of models that have capabilities to represent stationary time series and to produce accurate forecasts based on a description of historical data of a single variable. The ARIMA (p,d,q) model has three parts corresponding to three parameters [5]:

- p: The autoregressive part is a linear regression that relates past values of data series to future values;
- d: The integrated part indicates how many times the data series has to be differentiated to get a stationary series;
- q: The moving average part that relates past forecast errors to future values of data series.

The practical and pragmatic approach of Box-Jenkis methodology in order to build ARIMA models is based on the following steps: (1) Identification, (2) Parameter Estimation and Selection, (3) Diagnostic checking, (4) Model's use. Mathematically, the ARIMA (p,d,q) model can be expressed as [5]:

$$W_t = \mu + \frac{\Theta(B)}{\Phi(B)} a_t \tag{1}$$

where:

- t: Time index;
- W_t: D'nd difference of the variable of interest z_t;
- μ: Reference point of the process level;
- $\Theta(B)$: Moving averages operator $\Theta(B) = (1 - \Theta_1(B^1) - \Theta_2(B^2) - ... - \Theta_q(B^q))$;
- $\Phi(B)$: Auto regressive operator $\Phi(B) = (1 - \Phi_1(B^1) - \Phi_2(B^2) - ... - \Phi_p(B^p))$;
- B^p: Reverse operator $B^p z_t - z_{t-p}$;
- a_t: "white noise" or random error.

978-1-7281-4762-8/19/$31.00

ARIMA (p,d,q) stands for Auto Regressive Integrated Moving Average and can be expanded, as in equation 2

$$W_t = \Theta_0 + \Phi_1 W_{t-1} + ... + \Phi_p W_{t-p} + a_t - \Theta_1 a_{t-1} - ... - \Theta_q a_{t-q} \quad (2)$$

Where:

$$\Theta_0 = \mu(1 - \Phi_t - ... - \Phi_p). \quad (3)$$

B. Random Forest

The Random Forest [6] is a Machine Learning Algorithm based on Decision Trees. This algorithm lies in the class of ensemble classifiers, and it has gained a significant interest in the recent past, due to its quality performance in several areas [7], [8].

Given a training dataset, $Z = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, the Random Forest (RF) creates a set of B bootstrap samples, created from a random resampling on the training set itself ([9]). For each bootstrap sample $Z^b, b = 1, 2, ..., B$, a decision tree is constructed. When building these decision trees, each time a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

When growing the tree, the idea behind selecting a random sample of $m < p$ predictors to consider in each step leads to very different (or uncorrelated) trees from each sample. In this way, the bias increases in exchange for a decrease in variance.

Let $g^b(x)$ be the prediction function obtained from the b-th tree. The prediction function obtained by the Random Forest algorithm is expressed by equation 4 [10]:

$$g_{rf}(x) = \frac{1}{B} \sum_{b=1}^{B} g^b(x). \quad (4)$$

According to [11], the Random Forest technique is more accurate than other approaches, such as artificial neural networks and vector machines. In addition, this technique can avoid overfitting and is less sensitive to noise [6].

C. Support Vector Machine

Support Vector regression (SVM) is a powerful method for building a classifier for classification and regression,also know as Support Vector Regression (SVR),first identified by Vladimir Vapnik and his research team in 1992 [12].

The SVM algorithm creates a decision boundary, known as the hyperplane, between two classes that allow predicting labels from one or more feature vectors. This decision boundary is oriented such that it is as far as possible from the closest data points from each of the classes. These closest points are called support vectors.

Given a labeled training dataset, $Z = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), x_1 \in R^d$ and $y_i \in (-1, +1)$. The optimal hyperplane can then be defined as equation 5:

$$wx^T + b = 0. \quad (5)$$

where w is the weight vector, x is the input feature vector, and b is the bias.

The hyperplanes can be described by the following equations:

$$wx_i^T + b \geq +1 \text{ if } y_i = 1 \quad (6)$$

$$wx_i^T + b \leq -1 \text{ if } y_i = -1 \quad (7)$$

The objective of training an SVM algorithm is to find w and b in order that the hyperplanes separates the data and maximizes the margin $1/||w||^2$.

Originally proposed to construct a linear classifier, the SVM algorithm can be used to model higher dimensional or non-linear models using the kernel function ([10]). For instance, in a non-linear problem, the kernel function can be used to provide additional dimensions to the raw data and turn it into a linear problem in the resulting higher dimensional space. The kernel function can provide faster calculations which need computations in high dimensional space.

The kernel is an internal product, which can be linear or nonlinear (Polynomials), RBF (Radial-Basis Function), and the Sigmoid ([10]). It is defined as:

$$K(x, y) = < f(x), f(y) > \quad (8)$$

The results of this technique are comparable and often superior to those obtained by other learning algorithms, such as artificial neural networks ([13]).

D. Long-Short Term Memory

First proposed by Alex Graves [14], LSTM purpose it to deal with the vanish gradient problem perceived by the general Recurrent Neural Network architecture. To store information over extend time intervals the vanilla RNN suffers from the decay error backflow. The LSTM can learn dependencies in 1000 discrete time-steps with the enforcing error flow through special units (LSTM cells) with multiplicative gate units to open and close access to constant error flow.

Commonly used for stock price prediction, LSTM is in the state of art for time series prediction. The LSTM model prediction performance for financial time series is compared with many other model such as ARIMA [15] , GARCH [16], SVM [17], RF and vanilla MLP [18].

E. WaveNet

WaveNet is a neural network based on the Convolution Neural Network architeture; Therefore, to better undertand the WaveNet, we explain the origin and how convolution neural networks works. The convolutional neural networks (CNN) are algorithms that can also be considered as bioinspired, in the article published by Hubel and Wiesel [19] since they have the concept of using a receptive field and the cellular structure in layers. Using the Hubel and Wiesel model, Fukushima proposes the neocognitron [20], which follows the concept of hierarchical structure: LGB (lateral geniculate body) $\longrightarrow$ simple cells $\longrightarrow$ complex cells $\longrightarrow$ hypercomposite cells lower order $\longrightarrow$ higher order hypercomplex cells. Thus, larger

978-1-7281-4762-8/19/$31.00 © 2019 IEEE

receptive fields are more insensitive to the change of stimulus pattern, being the cells of greater order and greater complexity that responds more selectively. Neocognitron is able to recognize patterns in the image very similar to convolutional neural networks, however, if the concept of convolution between the layers is used.

Although CNN is famous for image classification and object recognition problems, recently this kind of neural networks has been employed for time series analysis. The model is almost the same, but some modification may be necessary (eg. using 1 dimensional convolution neural network). In the literature, other works have already explored CNN as a forecaster for time series stock prices, such as trend prediction with CNN of two dimensions [21], using CNN with a sliding window [3], and commom predictions using CNN [22].

WaveNet [23] is a variation of the CNN architecture fist created to solve text-to-speech problems. That architecture does not rely on neurons and activation functions but only on weights of the filters of each convolution layer. The window of weights is slide across the input series. The WaveNet relies on a causal structure, as show in Figure 1

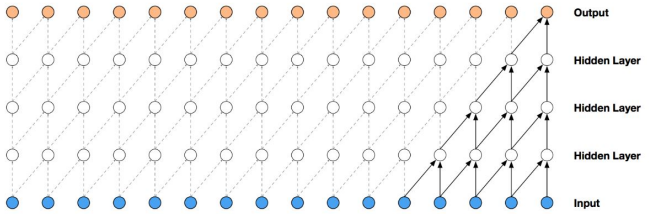

Fig. 1. Architecture proposed by the WaveNet autors [23]

III. Methodology

In this section the methods used in every model tested and the choices for hyper-parameters and variables are explored. We also explore the data processing used for the variables to remove the inconsistencies of the models and to improved training and test performance.

A. Data evaluation and processing

In this section we discuss the Bitcoin price time-series characteristics and we explore the main impacts of each choice on every aspect of the data processing.

We extracted the data from a repository base of financial time series (https:\investing.com). The data include the Bitcoin close price, open, high, low, percentage of change and the volume of transactions. This repository extract data from brokers Application Program Interface (API), in this case, from Bitfinex. After the collection, we checked data for outliers and possible inconsistencies. Since a very few registers were considered wrong after our analysis, we decided to exclude them from the dataset. The exclusion does not cause a great impact because of the volume of data available. The data is already extracted in csv format, is one of the standards for data to be worked with in Python. From the data we created others variables (further described) and exclude missing data. For data visualization we plotted the time series in a temporal graph. We chose different intervals to work with for working in a stable regime that could affect the model comparison. From a graph analysis we could identify the periods impacting the model accuracy randomly. The function present in those time periods seems to be completely different and should be modeled apart. From Figure 2, the parts excluded from analysis are explicit.

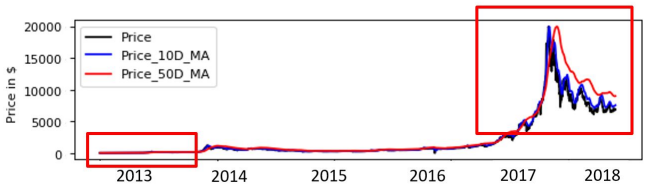

Fig. 2. Bitcoin price time series with respective moving averages 10D and 50D

We excluded the year 2018 and 2017 from the analysis since they are outliers and undergo great variance and random aspects. To model these last two years, a different model should be trained with fewer data. In our case, we used data from years 2012, 201, 2014, 2015 and 2016. From this, we normalized the data using minmax scaling. The minmax scaling is expressed by equation 9, where the $x = (x_1, ..., x_2)$ represents the data to be normalized and z_i is the i element normalized. We stored the data and to provide public access to it (https://doi.org/10.6084/m9.figshare.7445855).

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{9}$$

In order to better compare models we provided four prediction windows: prediction of the next day price,the price for 5 days in the future, price of 10 days in the future and price for 30 days in the future. We also assumed a window for past data we considered a hyper-parameter.

B. Variables selection

The variables selected were chosen based on the literature for time series of stock prices prediction or Forex (currency) prediction. The analyses test only variables associated with price such as Bitcoin close, open, high and low price and volumn (for one representative exchange). As Chen and Bahar [2], [24], we used moving average of the variables to generate new variables to capture different informations that could be hidden due to the high noise generated, characteristic of a highly volatile asset. Also, as [24], we used Gold an Death Cross, very common data for technical analysis. Some indexes, such as Stochastic Oscillator and Stochastic Momentum are used to measure the distances between assets. Stochastic Oscillator was used to calculate the distance between the Current Close and Recent High/Low Range for n-period and Stochastic Momentum Index calculated the distance of Current Close relative to the center of High/Low Range.

978-1-7281-4762-8/19/$31.00 © 2019 IEEE

C. Hyper-parameters optimization

To optimize the performance of a model, one important aspect to be considered is the choice of hyper-parameters. We here tried to perform the same kind of hyper-parameter optmization in order to be equally fair to the models compared. To select Hyper-parameters of the models, we used random search, which can be considered a more efficient way for the hyper-parameter search than grid search on the neural networks models, being a more natural base-line for the tests [25]. As proposed in Bergstra work [25], hyper-parameters random search is computationally more efficient since not all hyper-parameters are equally important to tune. Grid search or trial search from an expert can be better but does not seems fair for a model comparison.

To fairly compare all the machine-learning technique of this work, random search was a good option for the hyper-parameters search. The only model that we do not used random search was the ARIMA, since using Auto ARIMA is fairly standard and all is important to tune all parameters (p, d, q).

First all the hyper-parameters are defined for each model. This can be done somehow sense subjectively, but we here tried to work with as many as possible and relevant hyper-parameters. For each hyper-parameters we determined a subset of discrete possible values (eg. number of neurons of a neural network would assume only values in the subset of $50, 100, 200$).

D. Performance evaluation

To evaluate the system, different approaches can be employed and bench mark is normally an a index, a random walk or a similar risk asset Buy and Hold strategy. In this case, we compared the error metrics between the models and tested them with the t-student test to verify the null hypothesis. The error is evaluated through error metrics: mean error (ME), mean absolute error (MAE), root mean square error (RMSR), mean percentage error (MPE), mean absolute percentage error (MAPE).

For each metric error generated for a combination of hyper-parameters we also performed a 5-fold validation. This cross-validation is different since we work with time series and there is a temporal correlation between the train set and the test set. To solve this, we used a sliding train and test set in a larger data set. First, we fixed 600 registers as our minimum data set (MD), dividing it into train set (540 registers) and test set (60 registers). We have a total data set of 1006 (D) registers for the Bitcoin price time series along with the others variables correlated and created. To move train and test window, we considered the number of cross-validations (NCV) required and moved at a proportional rate.

The Figure 3 demonstrate the concept of how we do the cross-validation for time series without losing the time correlation of the train set and the test set. This is also used in some others works [26].

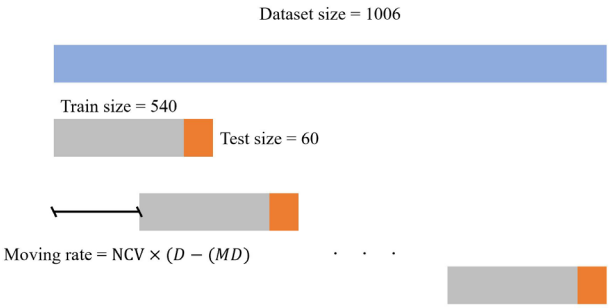

Fig. 3. The construction of a k-fold for time series maintaining the train and test size for a not limit data set

E. Autoregressive Integrated Moving Average (ARIMA)

For selecting the parameters for the ARIMA model, the Auto ARIMA procedure was used. The Auto ARIMA approach is used to perform a grid search over multiple values of p, d, q considering the AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) criteria. These generated values are used to determine the best combination of parameters. The mode parameters of the 5-fold were selected to fit the final model. A residual analysis was performed to verify if the final model adequately represented the data. The estimated residues of the model were verified to be uncorrelated.

F. Random Forest

For the Random Forest model, we chose the hyper-parameters to provide a better performance in the segmentation of the data so as not to obtain an over fitted model. The hyper-parameters varied were: *n_estimators*, which representing the number of trees in the forest; criterion, which corresponds to the function to measure the quality of a split; *max_depth*, the maximum depth of the tree; *min_samples_split*, the minimum number of samples required to split an internal node; min_samples_leaf, the minimum number of samples required to be at a leaf node; and *max_features*, the number of features to consider when looking for the best split.

G. Support Vector Machine

In the Support Vector Machine model, our choices of variable hyper-parameters were: kernel, which specifies the kernel method to be used in the algorithm for dividing the data edges; degree, degree of the polynomial kernel, but these are ignored by all the other kernels; gamma and C, responsible for defining a large or small edge between data segmentation.

H. Long-Short Term Memory

For the LSTM model, we fixed some of the hyper-parameter in the random search for providing a better performance in the search since they are more stabilized in the literature and can increase the process time without improve the metric errors. We assumed an Adam optmizer [27] that has been widely used in recent works. For the number of layers we assumed three

978-1-7281-4762-8/19/$31.00

layers because this satisfies the universal theorem requirements without compromising the performance. Some recent CNN networks using many layers (more than 20 layers) are having outstanding accuracy in the image classification problem, but for some problems, this increase in layers does not increase the accuracy as we verified for some small tests. Activation functions are many, but since the universal theorem is proved for many of than, we use the hyperbolic tangent activation function.

The hyper-parameters that varied were: dropout rate, window of time series input mentioned in the data processing section, number of neurons (equal for every layer), number of epochs and number of batches.

I. WaveNet

The input and output of the WaveNet model is the same of LSTM. We used a one-dimensional WaveNet for the time series prediction. All the hyper-parameters used for LSTM are also used for CNN. There are two main differences: the filter size and the number of convolutions. In each layer we made fewer convolutions since the main characteristics are extracted from the latest layers Regarding the padding, we adopted a padding that results in an output of the same size as the input. The activation function adopted is the Leaky ReLU. We added a batch regularization to improve the generalization power of the model. The convolutions occur with the one-dimensional vector with the size of the filter.

IV. Results

To compare all the results we considered all the error metrics. Since the MAE has the bias of the absolute value, it is difficult to compare depending on the period. in order to present all the results we also considered the best model of all hyper parameters tested for each future window of prediction (D1, D5, D10 and D30). The results in each table are the mean of all the 30 cross-validation of each model. We also highlight the best model(s) in bold. To consider the best model we did the $t - student$ test. If the null hypotheses is not satisfied, we highlight the best models among which we could not determine the best.

From Table I, we can identify that ARIMA and SVR are the best models considering most error metrics. The only error metric in which SVR was better was MAPE, a relative absolute metric. For the D1 prediction, we can infer that SVR is the best model, especially when contrasted with the compared neural networks.

TABLE I
D1 Results

Model	MAE	MSE	RMSE	MAPE	MPE
ARIMA	**10.73**	**260.125**	**15.048**	3.342	**-0.552**
RF	42.490	5401.505	53.628	7.656	6.714
SVR	**11.416**	**376.615**	**17.154**	**2.615**	**0.024**
LSTM	47.043	4320.829	54.815	10.934	4.357
WaveNet	61.294	7252.899	72.992	13.534	11.155

For the D5 prediction presented in Table II, we verify a high increase in the error, as expected. WaveNet, LSTM and RF did suffer that impact since they also performed badly for D1. As expected, all the results are negatively impacted by the increase in the prediction distance from the present, since present information values decay. As observed in D1, the ARIMA model has a negative MPE that indicates a bias different from the other models.

TABLE II
D5 Results - Model Error Comparison

Model	MAE	MSE	RMSE	MAPE	MPE
ARIMA	**22.494**	**1076.328**	**29.437**	**5.985**	**-1.076**
RF	57.162	7617.765	68.015	11.040	9.401
SVR	**26.412**	**1642.078**	**34.927**	**5.897**	3.225
LSTM	52.273	5572.884	62.864	12.159	3.477
WaveNet	51.704	5755.911	63.488	11.840	6.235

Notice that with the prevision gap increases, the error between the models get closer. The ARIMA and SVR still perform better than the other models but the difference between them is diminishing. Two hypotheses can be raised: Information is lost as we try to predict more distant data in the future and all the models tend to perform equally bad; neural networks are best suited for long-term predictions since they capture the non-linear aspects of the time-series that are not relevant in the short-term; in this case, ARIMA and SVR that can estimate linearly, are best suited for the short-term.

TABLE III
D10 Results - Model Error Comparison

Model	MAE	MSE	RMSE	MAPE	MPE
ARIMA	**36.267**	**2188.530**	**41.409**	**8.6901**	**-2.108**
RF	65.097	9488.238	76.525	12.919	11.784
SVR	**42.493**	**3798.064**	**54.151**	**9.161**	**5.598**
LSTM	59.812	7316.629	72.203	14.390	7.742
WaveNet	53.432	5640.600	65.340	12.546	6.341

For previsions of D30, the results are interesting since we expected WaveNet or the LSTM to perform better for long gaps of prevision, but it seems that all the models perform equally badly, and they are not statistically different in terms of the error metrics. This can also be a problem related to the complexity to optimize the neural networks architectures for an specific problem. All the results can also be tested using codes provided in https://github.com/leokan92/model_comparison.

TABLE IV
D30 Results - Model Error Comparison

Model	MAE	MSE	RMSE	MAPE	MPE
ARIMA	**69.074**	**9035.703**	**80.466**	**17.586**	**-7.141**
RF	**77.512**	**13153.947**	**94.599**	**15.628**	**11.585**
SVR	**67.894**	**8619.621**	**80.560**	**14.121**	**11.018**
LSTM	**64.854**	**8990.969**	**76.763**	**14.193**	**9.260**
WaveNet	**70.758**	**9651.507**	**82.918**	**15.739**	**8.909**

V. Conclusion

As expected, the error metrics indicate that all models perform worst as the prediction gap increases. The models of ARIMA and SVR tend to perform equally well in all

978-1-7281-4762-8/19/$31.00 © 2019 IEEE

the predictions gaps except for D30. For long-term predictions all models tend perform equally badly, which indicates the random component gets more important and affect the accuracy. These results are compatible with the Efficient Market Hypothesis confirming that linear models or auto regressive models can be more efficient in the a time-series that has a important random component. AIRIMA also has less important hyper-parameters then neural networks and machine learning methods in general, which facilitates the hyper-parameter search for better performance converging to a minimum faster.

For future works, the tests on neural network hyper-parameters could be improved to match the SVR and ARIMA models for understand how to search hyper-parameters for a problem of cryptocurrency time series prediction. Other techniques such as adaptive searches can be used to fairly search for hyper-parameters. Also, a wide range of hyper-parameters could also be tested to validate the results here presented. Another approach that could be followed is to take the best models and combine them to verify if this approach can improve accuracy. The bias can be a indication of which models we can combine and how.

The team have found that work with time series is so complex that one should consider the continuous aspect of the time series for the cross validation. Also, data with great variation due external facts could impact the models negatively; we thus decided to work with fewer data to turn the experiment and comparison more reliable. In the Bitcoin time-series, is clear the importance of a compatible model for different time-series.

ACKNOWLEDGMENT

We acknowledge the support from Coordination for the Improvement of Higher Education Personnel (CAPES) and Escola Politécnica da Universidade de São Paulo (EP-USP) - Brazil.

REFERENCES

[1] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 9, no. 6, pp. 1456–1470, 1998.

[2] J.-f. Chen, W.-l. Chen, and C.-p. Huang, "Financial Time-series Data Analysis using Deep Convolutional Neural Networks," in *2016 7th International Conference on Cloud Computing and Big Data*, 2016, pp. 99–104.

[3] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 1643–1647, 2017.

[4] J. C. B. Gamboa, "Deep Learning for Time-Series Analysis," 2017. [Online]. Available: http://arxiv.org/abs/1701.01887

[5] G. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[7] M. L. Calle and V. Urrea, "Letter to the editor: Stability of random forest importance measures," *Briefings in bioinformatics*, vol. 12, no. 1, p. 8689, January 2011. [Online]. Available: http://bib.oxfordjournals.org/cgi/content/full/12/1/86

[8] X. Chen and H. Ishwaran, "Random forests for genomic data analysis," *Genomics*, vol. 99, no. 6, pp. 323 – 329, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888754312000626

[9] J. Han, M. Kamber, and J. Pei. (2012) Data mining concepts and techniques, third edition. Waltham, Mass. [Online]. Available: http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[11] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 161–168. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143865

[12] I. Guyon, B. E. Boser, and V. Vapnik, "Automatic capacity tuning of very large vc-dimension classifiers," in *NIPS*. Morgan Kaufmann, 1992, pp. 147–155.

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[14] A. Graves, "Long Short-Term Memory," *Neural Computation*, vol. 1780, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[15] S. Siami Namin and A. S. Namin, "Forecasting Economic and Financial Time Series: Arima Vs. Lstm," Tech. Rep., 2018. [Online]. Available: https://arxiv.org/pdf/1803.06386.pdf

[16] H. Y. Kim and C. H. Won, *Forecasting the Volatility of Stock Price Index: A Hybrid Model Integrating LSTM with Multiple GARCH-Type Models*. Elsevier Ltd, 2018. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0957417418301416

[17] T. Gao, Y. Chai, and Y. Liu, "Applying long short term momory neural networks for predicting stock closing price," *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, vol. 2017-Novem, pp. 575–578, 2018.

[18] D. M. Q. Nelson, A. C. M. Pereira, and R. A. D. Oliveira, "Stock Market ' s Price Movement Prediction With LSTM Neural Networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, no. Dcc, 2017, pp. 1419–1426.

[19] T. N. Wiesel, "Receptive Fields and Functional Architecture of Monkey Striate Cortex," *J.Physiol*, pp. 215–243, 1968.

[20] K. Fukushima, "Neocognition: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," vol. 202, 1980.

[21] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-D CNN trend detection," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8285188/

[22] S. Ghoshal and S. Roberts, "Reading the Tea Leaves: A Neural Network Perspective on Technical Trading," in *KDD 2017*. University of Oxford, 2017. [Online]. Available: http://www-bcf.usc.edu/{~}liu32/milets17/paper/MiLeTS17{_}paper{_}4.pdf

[23] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," pp. 1–15, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[24] H. H. Bahar, M. H. F. Zarandi, and A. Esfahanipour, "Generating ternary stock trading signals using fuzzy genetic network programming," in *2016 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS)*, 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7851630/

[25] J. Bergstra and U. Yoshua Bengio YOSHUABENGIO, "Random Search for HyperParameter Optimization," *Journal of Machine Learning Research*, vol. 13, p. 281305, 2012.

[26] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock Market Prediction on High Frequency Data Using Generative Adversarial Nets," *Mathematical Problems in Engineering*, vol. 0, no. 0, 2018.

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," pp. 1–15, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

978-1-7281-4762-8/19/$31.00