# Analysis of Machine Learning Methods for Predicting Stock Prices*

Oluwadurotimi Onibonoje[1], Kevin Djoussa[2], and Mark Roantree[3]

[1] VistaMilk SFI Research Centre, Dublin City University, Ireland
oluwadurotimi.onibonoje2@mail.dcu.ie
[2] School of Computing, Dublin City University, Ireland
[3] Insight Centre for Data Analytics, Dublin City University, Ireland
mark.roantree@dcu.ie

**Abstract.** In this research, we investigated the applicability of Long Short Term Memory (LSTM) and Convolutional Neural Networks (CNN) in forecasting the next day's closing price of four major stock indices and explored de-noising techniques to improve the performance of these models. Our experiments show the use of Kalman Filters with the LSTM model provide the best forecast accuracy, reducing forecast error by at least 30% in three of the four financial time series used in this study.

**Keywords:** Time Series Analysis · Neural Networks · Signal Processing.



## 1 Introduction

Predicting the future price of stock indices has been shown to be an extremely challenging endeavor, largely due to the noisy and non-stationary characteristics of their time series [4]. Several approaches have been investigated in forecasting stock indices, eg. [3], [13]. Statistical approaches such as the linear autoregressive Integrated Moving Average (ARIMA) were used in forecasting the monthly stock price of the S&P 500 [2]. Deep learning architectures such as the LSTM implemented in [1] and Convolutional Neural Networks (CNN) as used in [11] are some of the non-linear models that have shown promise in this domain. These research initiatives provide evidence that using more sophisticated models can deliver better results.

**Motivation.** Stock market indices such as the Standard and Poor's 500 (S&P500) have been shown, through the Granger-causality test, to have predictive power as a leading indicator of the economy [6]. Therefore, accurate forecasts of stock market indices will help economic policy makers reach more informed conclusions as regards the right economic policies for desired economic outcomes. Institutional investors will also benefit from accurate forecasts of a stock market

* This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and the Department of Agriculture, Food and Marine on behalf of the Government of Ireland under Grant Numbers 16/RC/3835 and SFI/12/RC/2289-P2.

index, as these predictions will help inform the portfolio optimization process across different financial asset classes.

**Contribution.** This project attempts to forecast the univariate time series of four major stock indices, namely, the S&P 500, Dow Jones Industrial Average (DJIA), Euro Stoxx 50 (Stoxx50E) and the National Association of Securities Dealers Automated Quotation (NASDAQ) exchange. We applied three deep learning algorithms, namely, LSTM, CNN and CNN-LSTM on the daily closing prices for each index. In addition to investigating and understanding the efficacy of neural network architectures in time series forecasting, our research attempts to examine the merits of using de-noising techniques such as wavelet transform and Kalman filters on these financial time series. Novel ensemble approaches composed of these de-noising techniques and the neural network models were introduce in this paper were developed in a bid to improve the accuracy of the time series forecasts of our baseline model.

**Paper Structure.** The remainder of this paper is structured as follows: in section 2, we provide an overview of related research; in section 3, we examine the theoretical concepts and models that underpin our implementation models; in section 4, we present our methodology for detecting the best model configuration for day ahead forecasts of stock Indices; In section 5, we present our evaluation and discuss our findings; and finally, in section 6, we conclude the paper.

## 2 Related Research

In [1], the authors applied Deep Neural Networks to predict one-month ahead stock returns of stocks in the MSCI Japan Index. Their approach used 25 fundamental analysis factors for each stock in the cross-section of the Japanese stock market. The experimental results, which were evaluated using Rank Correlation, Directional Accuracy and Mean Square Error (MSE), showed that Deep Neural Networks outperformed other models including Support Vector Regression (SVR), Random Forest(RF) and Shallow Neural Network models with a 30% average uplift using the rank correlation metric and a 2.6% reduction in MSE.

The efficacy of Deep Belief Networks (DBN) was investigated in [9] with Technical Analysis Indicators as features and a 2-Dimensional Principal Component Analysis (PCA) model in predicting the S&P 500. Three models were formulated and evaluated using the RMSE metric in order to properly evaluate the usefulness of the Technical Analysis Indicators. The first model is composed of a Back Propagation Neural Network (BPNN) and the basic features in the raw dataset while the second model is composed of a DBN, basic features and extracted Technical Indicator features. The final model adds the complexity of a 2-Dimensional PCA to the previous model. The experimental results indicate that Technical Indicators coupled with PCA can help improve the predictive power of Deep Learning Algorithms. The final model had a 43.5% reduction in RMSE in comparison to the first model and a 16.91% reduction in RMSE when compared to the second model.

Wavelet Transforms have been studied in different applications with non-stationary time series and signals [24]. Wavelet transforms decompose a signal into components of different time scales. Li and Tam used wavelets as a real-time de-noising technique for financial time series of East Asian Stock Indices as seen in [17]. Their approach applies different mother wavelets with grid-searched hyper-parameters like the decomposition level and sliding window size on these stock indices to obtain a smooth time series, which will be processed by an LSTM Neural Network. The results of their research show that there is merit in using wavelets as a de-noising technique for neural network models. De-noised data improved directional accuracy of LSTM forecasts on original data inputs by an average of 7.6%.

Kalman Filter is a state space model [15], which uses an optimal recursive algorithm typically found in signal processing research. In [8], the authors compared the use of wavelet transforms and Kalman filters in de-noising signals. Their results show that the Coiflet 2 wavelet transform outperforms Kalman Filters in signal de-noising. Ma and Teng predicted chaotic time series using a variation of the Kalman Filter known as Unscented Kalman Filter (UKF) [19]. Lima and Neto used Kalman filters in conjunction with wavelets to pre-process the time series of the Brazilian IBOVESPA index [18]. The pre-processed time series is fed into a Recurrent Neural Network (RNN) to generate forecasts. Their results indicate a Mean Absolute Percentage Error (MAPE) of 0.72%, beating other models including ARIMA.

**Summary.** From this review, we can interpret the most recent approaches taken by researchers in seeking better stock price predictions. Some approaches have included technical indicator features using Neural Networks [9], linear statistical models [2] and wavelet de-noising on the input time series as seen in [17]. While these approaches are richly varied in their methodologies, we have not found a study that attempted to focus solely on the input financial series and compare the effect of the de-noising techniques that we propose to improve the forecasting ability of Neural Network models.

## 3 Background Models

In this section, we provide a brief overview of the models used in our evaluation and the 2 denoising techniques used in an attempt to improve model performance.

### 3.1 Long Short Term Memory (LSTM)

Recurrent Neural Networks (RNN) maintain an internal loop that allows for information persistence. The output of a RNN is used in conjunction with the current element in the input tensor, to compute the next element in the output sequence[10]. In more simplistic RNN models, the memory unit or state of the RNN is often equivalent to the previous output while other complex models have different values for the state and the previous element in the output sequence.

Equation 1 [10] illustrates the computational model output in a RNN for a given time step $t$. In this equation, the activation function $\varphi$ is a nonlinear hyperbolic tangent function and $W_x, W_y$ are matrices, containing connection weights for the input of the current time step and the outputs for the previous time step respectively. The state matrix from the previous time step is $h_{(t-1)}$ while the bias vector $b$ contains the bias term for each neuron.

$$Y_t = \varphi(X_{(t)}W_x + h_{(t-1)}W_y + b) \tag{1}$$

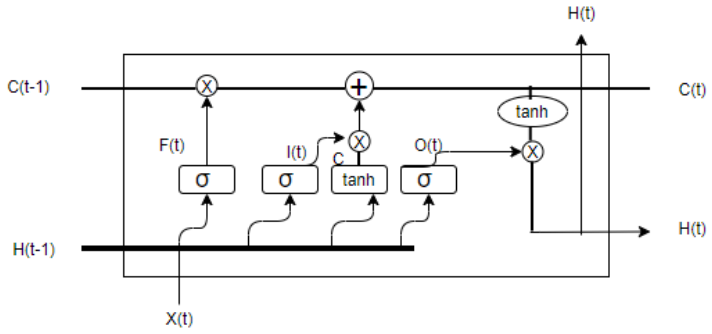


**Fig. 1.** Showing an LSTM cell.

LSTMs [14] address the long term dependency problem of RNNs by introducing three gate structures namely the *forget gate* $F$, *input gate* $I$, and *output gate* $O$ shown in Fig. 1. The forget gate function in equation 2 takes as input the previous state $h_{(t-1)}$ and the current input vector $X_{(t)}$ and passes these inputs into a sigmoid function which returns a value between 1 and 0 that represents the amount of information to flow through the gate.

$$F_t = \sigma(W_F[h_{(t-1)}, x_t] + b_F) \tag{2}$$

The input gate $I_{(t)}$ function in equation 3, as with the forget state, takes as input the previous state $h_{(t-1)}$ and the current input vector $X_{(t)}$ and passes these inputs into a sigmoid function. The input gate helps to determine the value to be updated.

$$I_t = \sigma(W_I[h_{(t-1)}, x_t] + b_I) \tag{3}$$

The output gate function $O_{(t)}$ shown in equation 4 determines those parts of the long-term state $C_{(t)}$ to be passed as output to $H_{(t)}$ for the current time step.

$$O_t = \sigma(W_O[h_{(t-1)}, x_t] + b_O) \tag{4}$$

The matrices $W_F, W_I, W_O$ contain the connection weights for the gate layers while $b_F, b_I, b_O$ are the bias terms for these layers.

### 3.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms shown to be highly effective in tasks relating to visual perception [16]. The architecture of a CNN incorporates the convolution layer and the pooling layer. These layers explain why CNNs outperform and are more efficient than traditional neural network architectures in computer vision tasks. In this research, we applied a 1D CNN to our financial time series.

The convolutional layer allows the Neural Network to capture spatial and temporal dependencies in the input feature map by applying a convolutional kernel on this input tensor. The layer iteratively parses each element of the input feature map by applying a sliding convolution kernel to produce a convolved output feature map, which models the translation invariant nature of the input feature map. The size of the output feature map is influenced by the size of the convolution kernel and the padding added to the input feature map to avoid losing the edge elements of the feature space.

The pooling layer down samples the convolved feature map by applying tensor operations with a *nxn* sliding window. There are principally two types of pooling operations namely: maximum pooling and average pooling. The maximum pooling operation computes the highest value in the current *nxn* window of the convolved feature map while the average pooling operation computes the mean. The pooling layer allows CNNs to better model spatial hierarchies present in the input feature map. The layer reduces the number of parameters of the input feature map hence, reducing the risk of overfitting and giving CNNs generalization power. In

### 3.3 Wavelet Transformation

Wavelet Transformation is a signal processing technique that has been used in many applications such as image compression, time-frequency analysis and data de-noising. The Continuous Wavelet Transform (CWT) provides the time-frequency representation and overcomes the resolution problems of the Short Time Fourier Transform (STFT). The CWT differs from the STFT by offering a variable size window function for the spectral components. The wavelet function derived from the mother and father wavelet [4] can be expressed using equations 5 and 6, where $a$ is the scale factor and $b$ is the translation factor.

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}}\Psi\left(\frac{(t-b)}{a}\right) \tag{5}$$

$$\Phi_{a,b}(t) = \frac{1}{\sqrt{a}}\Phi\left(\frac{(t-b)}{a}\right) \tag{6}$$

The formula for the CWT and the Inverse CWT [4] of a function is shown in equations 7 and 8.

$$Wf_{(a,b)} = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} f(x)\Psi\left(\overline{\frac{x-b}{a}}\right)dx \tag{7}$$

$$f(x) = \frac{1}{C_\Psi} \int_0^\infty \int_{-\infty}^\infty W f_{(a,b)} \Psi \left( \frac{t-b}{a} \right) db \frac{da}{a^2} \tag{8}$$

The father ($\varphi$) and mother ($\Psi$) wavelets have the properties [7] shown in equation 9.

$$\int \phi(t)dt = 1 \int \psi(t)dt = 0 \tag{9}$$

When a given signal is decomposed into the approximation and detail coefficients using a Discrete Wavelet Transform (DWT) at $j$-level, the father wavelets and mother wavelets can be represented as in equation 10 and 11.

$$\varphi_{(j,k)}(t) = 2^{(\frac{-1}{2})} \varphi(2^{(-j)} - k) \tag{10}$$

$$\Psi_{(j,k)}(t) = 2^{(\frac{-1}{2})} \Psi(2^{(-j)} - k) \tag{11}$$

Here, we applied the Haar wavelet to de-noise our input financial time series. The Haar wavelet is computationally more efficient than other mother wavelets and has shown to be capable of improving results in this domain [4].

### 3.4 Kalman Filters

Kalman Filters estimate the state of a system given measurements with expected errors. Their efficiency in making time series forecasts make them widely applied in time series analysis and real-time applications [21]. A linear Gaussian model for the state and observation of a measured process is shown in equations 12 and 13, where $x_t$ is the real value at a given time $t$ for the measured system and $y_t$is the measured value at $t$.

$$x_t = F \times x_{t-1} + B \times x_t + w_t \tag{12}$$

$$x_t = A \times x_t + v_t \tag{13}$$

In order to determine the real state of the system at a given time t, there are three functional components. The first component, $F \times x_{t-1}$, shows the functional relationship between the value of the previous state $x_{t-1}$ and the current state $x_t$. The second component, $B \times u_t$, is an external force term [21]. The third component, $w_t$ is a stochastic term which captures dynamics not present in the previous state. The measured value, $y_t$, is determined by applying a function to the real value of the current state, $A \times x_t$, and adding a white Gaussian noise $v_t$. The Kalman filter forecasts the future value using equation 14, where $K_t$ is the Kalman gain.

$$\hat{x}_t = K_t \times y_t + (1 - K_t) \times \hat{x}_{t-1} \tag{14}$$

The Kalman Filter recursively iterates between the prediction and filtering phase [8] with the prediction phase described by equations 15, 16, and the filtering phase by equations: 17, 19, where $P_t$ the estimate of the state covariance; $R$ is the measurement error variance; and $Q$ is a tunable hyper-parameter for improving the performance of the model.

$$\hat{x}_t^- = F \times \hat{x}_{t-1} + B \times u_t \tag{15}$$

$$\hat{P}_t^- = F \times P_{t-1} \times F^T + Q \tag{16}$$

$$\hat{x}_t = \hat{x}_t^- + K_t \times (y_t - A \times \hat{x}_t^-) \tag{17}$$

$$P_t = (I - K_t \times A) \times P_t^- \tag{18}$$

The Kalman gain, $K_t$, attempts to determine the relative importance of the measured error of the estimate when compared to the error of the real value. The computation of the Kalman gain is described as follows [21] and shown in equation 19.

$$K_t = P_t^- \times A^t \times (A \times P_t^- \times A^T + R)^{-1} \tag{19}$$

## 4 Methodology

In this research, with the exception of our baseline ARIMA model, we relax the stationarity condition for the financial time series in our neural network models as in [15,24]. The implementation logic for our NN models was adapted from the approach presented in [5].

**Data Preprocessing.** The dataset used in this research was obtained from Yahoo! Finance from 01-01-2004 to 31-12-2019 for the following stock indices: NASDAQ 100 (NDX), S&P 500, Euro Stoxx 50 and the Dow Jones Industrial Average. The daily closing price series for each index was transformed into a 1-D tensor composed of 100 successive daily values and mapping our independent variable $\hat{x} = x_{t+1}, x_{t+2} \ldots x_n$ to the corresponding dependent variable $\hat{y} = x_{n+1}$. The values of the independent variable were standardized using Min-Max normalization to have values within the range of (0,1).

### 4.1 Evaluation Models

**Baseline Model** ARIMA was selected as a baseline model where the result is a benchmark to measure the improvement in forecast accuracy from other more sophisticated models. The steps are as follows:

- Step 1: The dataset is split into a training set and a test set using a 70:30 ratio;
- Step 2: The hyper-parameters, $p$,$d$,$q$ are grid searched on the training set to produce the optimal model with the lowest Mean Squared Error;

– Step 3: predictions from the ARIMA model are compared with the values in the test set and evaluated using the MSE.

**Wavelet Transform - Convolutional Neural Network (WT-CNN).** The specification of this sequential CNN architecture is as follows: Layer 1 is a 1-D convolutional layer with 3 filters and 3 kernels with the Rectified Linear Unit (ReLU) as the activation function. Layer 2 is also a 1-D convolutional layer that has identical hyper-parameters to the preceeding layer. Layer 3 is a 1-D maximum pooling layer followed by a 4th layer which flattens the two dimensional tensor into a vector fed into a fully connected layer, not unlike the data model approach taken in [12]. The loss function which the model optimizes using the Adam Optimizer is the Mean Squared Error (MSE). The following steps were followed for the implementation of the WT-CNN model:

– Step 1: The time series is de-noised by Haar Wavelet with soft thresholding.
– Step 2: The de-noised time series is scaled to the range {0,1}. This is to allow the CNN to converge faster.
– Step 3: The dataset is divided into a training and test set using a 70:30 split.
– Step 4: Both the training and test sets are converted to a supervised learning problem with 100 past sequences representing the independent variable used to predict the next value in the sequence.
– Step 5: The training and test input tensors are reshaped to have the following dimensions: $[samples, timesteps, features]$.
– Step 6: The input tensors are fed into the CNN and the network is trained over 100 epochs.
– Step 7: The predictions generated from the CNN are standardized to their normal range and then compared to the test set to generate values for the evaluation metrics.

**WT-CNN-Long Short Term Memory (WT-CNN-LSTM).** The specification of the sequential LSTM architecture is as follows: Layer 1 is a 1-D convolutional layer with 64 filters and a kernel with the Rectified Linear Unit as the activation function. Layer 2 is a 1-D maximum pooling layer with a pooling size of 2, followed by a layer which flattens the two dimensional tensor into a vector fed into a LSTM layer, with 50 neurons and a ReLu activation function. The LSTM layer outputs a tensor to a fully connected layer. Once again, the MSE loss function is optimized using the Adam Optimizer. The steps followed in the implementation of the WT-CNN-LSTM model mirror those of the WT-CNN model with the major differences being the change in dimension of the input tensors from $[samples, timesteps]$ into $[samples, subsequences, timesteps, features]$.

**WT-LSTM.** The specification of the sequential LSTM architecture is as follows: the first three layers are LSTM layers with 50 neurons and the final layer is a fully connected layer. The implementation logic of the WT-LSTM is not too different from previous models, where the major difference being the dimension of the input tensors: $[samples, timesteps]$.

**Kalman Filter-LSTM (KF-LSTM).** In this model, the input time series is first passed into the Kalman filter as a 1-D tensor. The output of the de-noising

process returns a 2-D tensor that is reshaped into a 1-D shape before it is fed into an LSTM network.

## 5 Evaluation

In this section, we present the results of our model implementations, evaluated using the RMSE and MAE metrics obtained for each stock index. These evaluation metrics measure the distance between predictions from the algorithm and the actual values in the test set.

**Table 1.** Evaluation metrics for baseline model - ARIMA

| Index | RMSE | MAE |
|---|---|---|
| S&P 500 | 20.11 | 14.05 |
| DJIA | 181.88 | 126.17 |
| STOXX50E | 34.39 | 21.26 |
| NDX | 63.09 | 44.75 |

### 5.1 Results

**Table 2.** Neural Network Model Performance before Denoising

| **Model** | Index | Model RMSE | Model MAE |
|---|---|---|---|
| LSTM | S&P 500 | 52.25 | 45.90 |
| CNN-LSTM | S&P 500 | 30.40 | 20.18 |
| CNN | S&P 500 | 62.40 | 44.91 |
| LSTM | DJIA | 280.86 | 214.70 |
| CNN-LSTM | DJIA | 420.93 | 298.29 |
| CNN | DJIA | 617.98 | 456.07 |
| LSTM | STOXX50E | 29.63 | 22.23 |
| CNN-LSTM | STOXX50E | 51.61 | 41.31 |
| CNN | STOXX50E | 60.60 | 53.65 |
| CNN | NDX | 100.22 | 76.60 |
| LSTM | NDX | 144.80 | 119.1 |
| CNN-LSTM | NDX | 188.55 | 147.27 |

Our experiments show LSTMs outperform other Neural Network architectures in forecasting the univariate time series of stock indices. With the exception of the NDX stock index, LSTMs improve the forecast performance of the One Dimensional CNN by over 50% for each stock index. When compared with the CNN-LSTM architecture, LSTMs marginally outperform in forecasting the time series of the S&P 500 and the Euro Stoxx50E index. There is a more widened gap between these two architectures, however, for the Dow Jones index and the NDX index.

From Table 2, we can see that the LSTM model configuration achieved superior RMSE scores when compared to the CNN and CNN-LSTM for both the S&P500 and the STOXX50E stock index. The CNN network had the worst performance in trying to predict all of the stock indices, with the exception of the NDX. Interestingly, Only the LSTM model for the STOXX50E outperformed the baseline model, ARIMA.

**Table 3.** Neural Network Model Performance after Denoising using Wavelet Transform

| Model | Index | RMSE | MAE |
|---|---|---|---|
| WT-LSTM | S&P 500 | 24.88 | 18.710 |
| WT-CNN-LSTM | S&P 500 | 28.96 | 20.70 |
| WT-CNN | S&P 500 | 127.34 | 103.11 |
| WT-LSTM | DJIA | 206.71 | 144.922 |
| WT-CNN | DJIA | 364.33 | 253.05 |
| WT-CNN-LSTM | DJIA | 613.31 | 527 |
| WT-CNN-LSTM | STOXX50E | 33.52 | 24.13 |
| WT-LSTM | STOXX50E | 35.04 | 29.51 |
| WT-CNN | STOXX50E | 78.06 | 64.98 |
| WT-CNN | NDX | 131.94 | 100.28 |
| WT-LSTM | NDX | 161.08 | 117.25 |
| WT-CNN-LSTM | NDX | 532.14 | 428 |

Using Table 3, we find similar patterns to those which were obtained in Table 2 with respect to LSTMs outperforming other models for stock indices such as the S&P 500 and DJIA. For the best forming neural network models on stock indices such as the STOXX50E and S&P 500, wavelet transform delivered similar results with the ARIMA model. The worst model performance came from the CNN-LSTM model with the exception of the STOXX50E dataset.

**Table 4.** Neural Network Model Performance after Denoising using Kalman Filters

| Model | Index | RMSE | MAE |
|---|---|---|---|
| KF-LSTM | S&P 500 | 16.60 | 13.43 |
| KF-LSTM | DJIA | 132.11 | 100.29 |
| KF-LSTM | STOXX50E | 9.52 | 8.51 |
| KF-LSTM | NDX | 95.69 | 57.075 |

### 5.2 Analysis and Findings

In many tasks involving time series analysis, CNNs generally outperform LSTMs but this was not the case for the evaluated results of the time series forecasts presented in Table 4. Our assumption is that this is due to the two-dimensional architecture of the network that allows it to capture both spatial and temporal information inherent in the time series.

We found that using wavelets to remove noise in our univariate time series to be inconclusive. On one hand, predictions using the Dow Jones index showed a 20% average reduction of in forecast error when de-noised using wavelet transforms. However, predictions using the NDX and Stoxx50E indices showed no substantial reduction in forecast error. We interpret that the application of Haar wavelet decomposition may not be suitable for all types of financial time series as suggested by [17]. Indeed, each financial time series may require a unique mother wavelet for its decomposition. With the exception of models that used Kalman filtering, no model could consistently beat the (baseline) ARIMA model in forecasting for each index. These results show that increasing model complexity does not guarantee improved performance. Many studies show that statistical

forecasting techniques such as ARIMA can often outperform Neural Network architectures [5]. Kalman filters reduced the forecast error of the LSTM on the S&P 500 and Stoxx50E by 68%. The forecast errors in the Dow Jones index reduced by 53% while those in NDX reduced by 34%.

In summary, we found de-noising performance of Kalman Filters to outperform wavelets for the financial time series in our evaluation. Our Neural Network models used the previous $N = 100$ samples to predict the next price in the sequence, with the choice of $N$, being arbitrary.

## 6 Conclusions

In this paper, we applied three deep learning algorithms, LSTM, CNN, and CNN-LSTM to forecast the univariate time series of four stock indices; S&P 500, Dow Jones Industrial Average, Euro Stoxx 50 and the Nasdaq Exchange. Initially, we attempted to forecast the future prices of the stock indices using Neural Networks; we then investigated the efficacy of Discrete Wavelet Transforms, particularly the Haar Mother Wavelet to de-noise the input financial time series; and finally, we investigated the use of Kalman Filters and discovered better performance when compared to the wavelet transform approach. Results were evaluated using the RMSE and MAE metrics. While our evaluation does provide support for ARIMA, for forecasting using time series, we believe that our results using some de-noising techniques suggest that other approaches may outperform ARIMA, given the appropriate experimental configurations.

Our current work is focused on the exploration of other input features to enhance the performance of the neural network models: configuring the type of mother wavelet applied, decomposition level and window size may well deliver improved performance. We are also seeking to investigate the optimal configurations of DWT for more frequent observations of these time series.

## References

1. M. Abe and H. Nakayama, "Deep Learning for Forecasting Stock Returns in the Cross-Section", ArXiv, 2018. [Accessed 14 August 2020].
2. A. Ariyo, A. Adewumi and C. Ayo, "Stock Price Prediction Using the ARIMA Model", 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, 2014. Available: 10.1109/uksim.2014.67 [Accessed 1 August 2020
3. Ken Bailey, Mark Roantree, Martin Crane, and Andrew McCarren. Data Mining in Agri Warehouses Using MODWT Wavelet Analysis. 23rd Intl. Conf. on Information and Software Technologies, pp. 241-253, Springer, 2017.
4. W. Bao, J. Yue and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory", PLOS ONE, vol. 12, no. 7, p. e0180944, 2017.
5. J. Brownlee, Deep Learning for Time series Forecasting. 1st edn. 2020.
6. B. Comincioli, "The Stock Market As A Leading Indicator: An Application Of Granger Causality", University Avenue Undergraduate Journal of Economics, vol. 1, no. 1, 1996. Available: https://digitalcommons.iwu.edu/uauje/vol1/iss1/1.

7. A. Dghais and M. Ismail, "A study of stationarity in time series by using wavelet transform", 2014.
8. K. Erkan and E. Bolat, "Comparison of Kalman Filter and Wavelet Filter for Denoising", 2005 International Conference on Neural Networks and Brain, 2005.
9. T. Gao, X. Li, Y. Chai and Y. Tang, "Deep learning with stock indicators and two-dimensional principal component analysis for closing price prediction system", 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016.
10. A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras Tensorflow. 2nd edn. O'Reilly Media, Inc. Sebastopol(2019).
11. H. Gunduz, Y. Yaslan and Z. Cataltepe, "Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations", Knowledge-Based Systems, vol. 137, pp. 138-148, 2017. Available: 10.1016/j.knosys.2017.09.023.
12. Piotr Habela, Mark Roantree, Kazimierz Subieta. Flattening the metamodel for object databases. Proceedings of ADBIS, Lecture Notes in Computer Science vol. 2435, pages 263-276, Springer, 2002.
13. B. Henrique, V. Sobreiro and H. Kimura, "Literature review: Machine learning techniques applied to financial market prediction", Expert Systems with Applications, vol. 124, pp. 226-251, 2019. Available: 10.1016/j.eswa.2019.01.012.
14. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997. Available: 10.1162/neco.1997.9.8.1735
15. R. Kalman, "A New Approach to Linear Filtering and Prediction Problems", Journal of Basic Engineering, vol. 82, no. 1, pp. 35-45, 1960.
16. A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.
17. Z. Li and V. Tam, "Combining the real-time wavelet denoising and long-short-term-memory neural network for predicting stock indexes", 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017.
18. F. Lima and A. Neto, "Combining Wavelet and Kalman Filters for Financial Time Series Forecasting", Asian Economic and Financial Review, 2014.
19. J. Ma and J. Teng, "Predict chaotic time-series using unscented Kalman filter", Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), 2004.
20. A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network", Procedia Computer Science, vol. 170, pp. 1168-1173, 2020. Available: 10.1016/j.procs.2020.03.049.
21. A. Nielsen, Practical time series analysis. 1st edn. O'Reilly Media, Inc, Sebastopol(2019).
22. R. Polikar, Wavelet Tutorial - Part 2, http://users.rowan.edu/ polikar/WTpart2.html. Last accessed 01 Aug 2020
23. J. Rankin, "Kalman filtering approach to market price forecasting."
24. M. Rhif, A. Ben Abbes, I. Farah, B. Martínez and Y. Sang, "Wavelet Transform Application for/in Non-Stationary Time-Series Analysis: A Review", Applied Sciences, vol. 9, no. 7, p. 1345, 2019.