Available online at http://isas.org.in/isa/jisas
JOURNAL OF THE INDIAN SOCIETY OF AGRICULTURAL STATISTICS 77(2) 2023 155–170

# A Comparative Study of Advance Forecasting Models on Volatile Time Series Price Data

**Rabsanjani Pramanik[1,2], Md. Wasi Alam[2], K.N. Singh[2], Mrinmoy Ray[2], Harish Nayak[2], Rajeev Ranjan Kumar [2] and K.K. Chaturvedi[2]**

*[1]The Graduate School, ICAR-Indian Agricultural Research Institute, New Delhi*
*[2]ICAR-Indian Agricultural Statistics Research Institute, New Delhi-110012*

*Received 01 December 2022; Revised 15 February 2023; Accepted 04 March 2023*

## SUMMARY

Efficient and reliable forecasting techniques for commodities with volatile price series are indispensable in agriculture dependent country like India. In this context, choosing a universally accepted model for forecasting precisely the price series of commodities like onion is one of the most challenging tasks because of the existence of seasonality, non-linearity and complexity in the data, simultaneously. Time series models like GARCH, machine learning techniques like TDNN, SVM and deep learning models like LSTM, Stacked LSTM and Bi-LSTM have been extensively studied in this research work to judge their performance on volatile weekly price series of onion for two different markets in India. The models were tuned with the training dataset to forecast the values for the next twelve horizons and eventually the forecasted values have been compared with the testing dataset. It was found that deep learning models outperformed the machine learning techniques as well as conventionally used time series models in dealing with the two volatile datasets.

*Keywords:* Non-linearity; Machine learning techniques; Long Short term memory; Price forecasting.

## 1. INTRODUCTION

When making decisions about production and marketing that potentially have a financial impact, farmers rely on price projections. Time series forecasting is the projection of future values using historical records, data and various models. Choosing a universally accepted model for forecasting precisely the price series of commodities like onion is one of the most challenging tasks because of the simultaneous existence of seasonality, non-linearity and complexity. A lot of agricultural items have been damaged recently owing to various climatic changes, which has caused many issues around the world. It's interesting how noisy and fluctuating the pricing data for many agricultural goods are by their very nature. This is due to the fact that agricultural commodity prices react quickly to changes in supply and demand situations, both actual and speculative; additionally, weather-related fluctuations in farm productivity make the issue worse.

Volatility is a series' abrupt, unexpected rise or collapses which could enrage stakeholders. It is a well-known truth that price volatility can make agricultural revenue unstable and prevent farmers from making investments and making the best use of their resources. It may ultimately cause the agriculture sector to lose access to vital resources. The series is said to be volatile when a few error terms are larger than the others and are responsible for the unique behaviour of the series, such a phenomenon is known as heteroscedasticity. The popular and non-linear autoregressive conditional heteroscedastic (ARCH) model was developed by Engle (1982) to cope with heteroscedasticity. Bollerslev (1986) expanded the model and created the Generalized ARCH (GARCH) model for a sparse representation of ARCH. Time Delay Neural Network models can be useful for non-linear processes that have an unknown functional relationship and, as a result, are difficult to fit. It is a specific general-purpose learning algorithm that deals with the process of building the relationship between

*Corresponding author*: Md. Wasi Alam
*E-mail address*: alamwasi01@gmail.com

the input and output variables. The behaviour of the central nervous system of the human brain is mimicked by computational techniques called neural network models. They are a group of data-driven, generalized non-linear, nonparametric statistical approaches. A supervised machine learning approach called Support Vector Machine (SVM) is used for both classification and regression. Although we also refer to regression concerns, categorization is the most appropriate term. Finding a hyperplane in an N-dimensional space that clearly classifies the data points is the goal of the SVM method. The number of features determines the hyperplane's size. The hyperplane that depicts the most significant gap or margin between the two classes is a logical option for the best hyperplane. When searching for the ideal hyperplane to maximize the margin, the SVM algorithm has the ability to ignore outliers.

Long Short Term Memory is a deep learning model in which the activation patterns in the network change once per time step, analogous to how physiological changes in synaptic strengths store short-term memories. The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories. The "long short-term memory" of the LSTM architecture is intended to give RNN a short-term memory that can endure thousands of time steps. Since there may be lags of uncertain length between significant occurrences in a time series, LSTM networks are well-suited to categorizing, processing, and making predictions based on time series data. Different variants of LSTM like Deep LSTM, Stacked LSTM, Bi-LSTM, etc. are known in literature.

## 2. DATA AND METHODOLOGY

### Data

In this study, two datasets have been used for evaluating and comparing forecast performance of different individual models. We have taken weekly price series of onion (Rs./quintal) of Azadpur market of Delhi and Kolhapur market of Maharashtra. The dataset was obtained from AGMARK website (https://agmarknet.gov.in/) for the period of 2010 to 2018.

For avoiding over fitting of the model and evaluating the model with good accuracy, the dataset was split into training and testing sets. The training dataset has been used for building model and in-sample prediction whereas the testing dataset has been used for validation purpose and out-sample forecasting. In case of Azadpur price series, there are total 469 observations (457 for training and 12 for testing) and there are 412 observations in Kolhapur price series (400 for training and 12 for testing). The last 12 weeks data has been used for validation of model. In this study, all estimation procedures have been done by using R software. Two unit-root tests such as Augmented Dickey Fuller test and Phillips-Perron test have been used to test stationarity of time series.

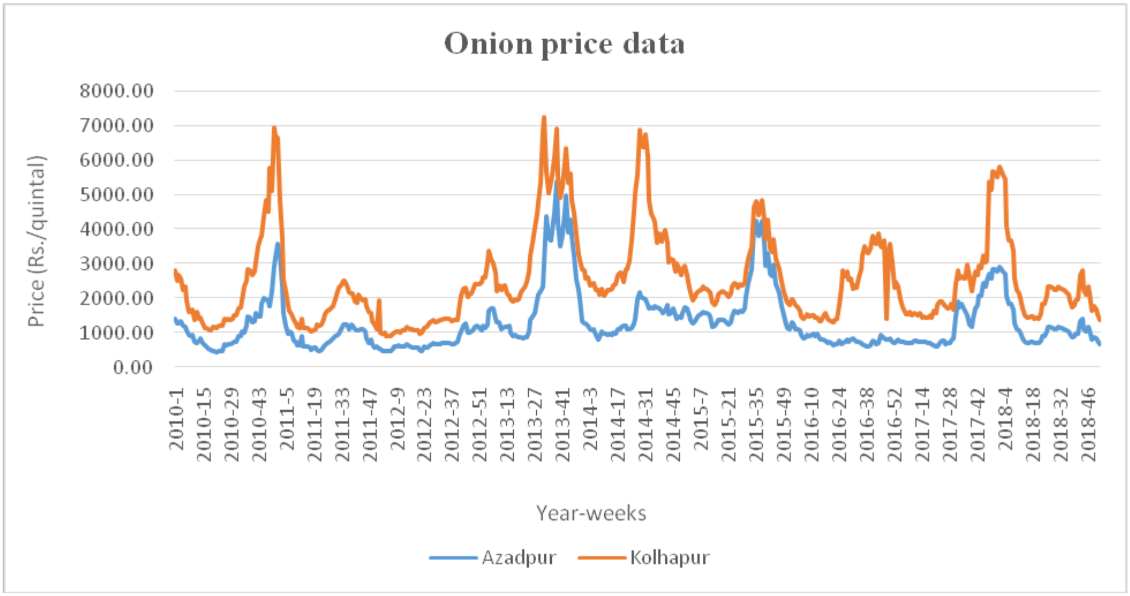

**Fig. 1.** Graphical plot of the weekly price series data for both the markets

## 3. METHODOLOGY

### Augmented Dickey Fuller (ADF) Test

If the mean and variance of the stochastic time series data are constant over the time, then the time series is said to be stationary. It means that there is no growth or decline in the data. The data must be roughly horizontal along the time axis. Some of the statistical tests for testing stationarity of time series are Augmented Dickey Fuller (ADF) Test and Phillips-Perron test (PP test).The ADF test consists of estimating the following regression equation:

$$\Delta y_t = a_1 + a_2 t + \delta y_{t-1} + \sum_{i=1}^{h} b_i y_{t-i} + e_t$$

Where, $\Delta y_t = y_t - y_{t-1}$, $a_1, a_2$ and $b_i$ are the parameters of regression model, $h$ is length of the lag, $\delta = \rho - 1$ and $-1 \le \rho \le 1$.

In ADF test, whether the time series having a unit root or not, which means whether the time series under consideration is stationary or not has been tested.

$H_0$ : $\delta = 0$, Time series is non-stationary

$H_1$: $\delta \neq 0$, Time series is stationary

Based on p-value of the test, we decide whether the data is stationary or not.

Phillips-Perron test (PP test) is a non-parametric test to check the stationarity of the time series data with null hypothesis of a unit root that explicitly allows for weak dependence and heterogeneity of the error process.

### GENERALISED AUTOREGRESSIVE CONDITIONAL HETEROSCEDASTICITY (GARCH)

The conditional variance in the GARCH model is also a linear function of its own lags. This model, like ARCH, is a weighted average of previously squared residuals, but unlike ARCH, it contains decreasing weights that never reach zero.

In our study, Lagrange multiplier test has been applied to detect the presence of autoregressive conditional heteroscedastic (ARCH) effect before going to fit the GARCH model.

The ARCH (q) model for the series ($\varepsilon_t$) is given by

$$\varepsilon_t \mid \Psi_{t-1} \sim \mathrm{N}(0, h_t) \tag{1}$$

here $\Psi_{t-1}$ denotes information available up to time $t-1$ and

$$h_t = a_0 + \sum_{i=1}^{q} a_i \varepsilon_{t-i}^2 \tag{2}$$

where $a_0 > 0$, $a_i \ge 0$ for all i and $\sum_{i=1}^{q} a_i < 1$ are required to be satisfied to ensure non-negativity and finite unconditional variance of stationary $\{\varepsilon_t\}$ series

Bollerslev (1986) proposed the Generalized ARCH (GARCH) model in which conditional variance is also a linear function of its own lags and has the following form

$$h_t = a_0 + \sum_{i=1}^{q} a_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} b_j h_{t-j} \tag{3}$$

A sufficient condition for the conditional variance to be positive is $a_0 > 0,\ a_i \ge 0\, i = 1,2,\ldots,q; b_j \ge 0\ , j = 1,2,\ldots,p$

The GARCH (p,q) process is weakly stationary if and only if $\sum_{i=1}^{q} a_i + \sum_{j=1}^{p} b_j < 1$

To express GARCH model in terms of ARMA model, denote $\eta_t = \varepsilon_t^2 - h_t$.

Then from eq. (3)

$$\varepsilon_t^2 = a_0 + \sum_{i=1}^{Max\,(p,q)} (a_i + b_i)\, \varepsilon_{t-i}^2 + \eta_t + \sum_{j=1}^{p} b_j \eta_{t-j} \tag{4}$$

Thus a GARCH model can be regarded as an extension of the ARMA approach to squared series $\{\varepsilon_t^2\}$.

### TIME DELAY NEURAL NETWORKS (TDNN)

The main disadvantage of the linear model like ARIMA is that it does not capture the non-linear component of time series data, which is bounded by residuals in the case of non-linear time series. Machine learning techniques will be more appropriate in dealing with such a situation. Time Delay Neural Networks (TDNN) are an efficient alternative forecasting model for non-linear time series data (Zhang *et al.*, 1998). The term “Neural Network” models comes from their ability to mimic the behaviour of the human brain’s central nervous system. They are a type of nonlinear, nonparametric, and data-driven statistical method.

The design consists of an input layer that accepts outside data, one or more hidden layers that provide non-linearity to the model, and an output layer that outputs the desired result. The ability of TDNN to model nonlinear systems and its high forecasting accuracy increased their appeal for time series forecasting significantly. A neural network with one hidden layer can approximate any non-linear function given enough hidden nodes and training data points.

## ARCHITECTURE OF TDNN

A TDNN model is made up of numerous interconnected neurons. A standard TDNN model consists of one input layer, one output layer, and one or more hidden layers. There are nodes in each layer, such as input nodes, output nodes, and hidden nodes. Each layer is distinguished by the fact that its output is the weighted sum of its inputs.

$$u_i = \sum_j w_{ij} * output_j + v_i$$

where $u_i$ is the value of net input of i$^{th}$ node, $w_{ij}$ is the weights connecting j$^{th}$ to i$^{th}$ neuron, $v_i$ denotes bias for i$^{th}$ node. Many non-linear functions are available in literature to be used as an activation functions by researchers. The two mostly widely used activation functions are identity function and sigmoid function.

$$\varnothing(netinput) = \begin{cases} 1, netinput \geq 0 \\ 0, otherwise \end{cases}$$

$$g(netinput) = \frac{1}{1 + e^{-netinput}}$$

where $\varnothing$ and $g$ represents identity and logistic activation function.

In this study, feed-forward time-delay neural network (TDNN) with single hidden layer will be employed as a multi-scale learning tool for fitting the training data. The general expression for a TDNN with single hidden layer is given by (Jha and Sinha, 2014)

$$y_{t+1} = g\left(\sum_{j=0}^{q} \alpha_j f\left(\sum_{i=0}^{p} \beta_{ij} y_{t-i}\right)\right)$$

where $y_{t+1}$ is the predicted value for $y_t$ at time t, $\alpha_j\,(j = 0,1,2, \ldots, q)$ and $\beta_{ij}\,(i = 0,1,2, \ldots, p; j = 1,2, \ldots, q)$ are the model parameters, $p$ is number of input layer nodes, $q$ is the number of hidden layer nodes, $f$ and $g$ denote the activation function at hidden and output layer respectively and $y_{t-i}$ is the i$^{th}$ input (lag) of the model. There is no theoretical idea for determining number of layers and nodes. These parameters are actually identified by performing experiments using given data i.e. trial and error method.

Numerous academic studies have concluded that a neural network model with a single hidden layer is adequate for accurately simulating any complex non-linear function. Because selecting an activation function is the primary focus when using a neural network model, it is also a difficult process. To predict future values using previously recorded values, this work employs a single hidden layer and a single layer. The logistic sigmoid transfer function is the most commonly used activation function for adding nonlinearity to a model.

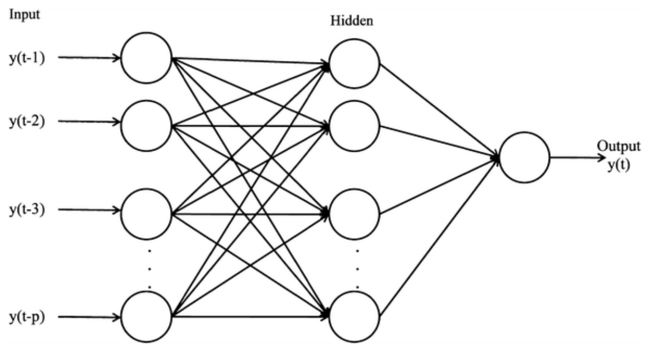

**Fig. 2.** Time Delay Neural Network (TDNN) with one hidden layer

## SUPPORT VECTOR MACHINE (SVM)

The support vector machine (SVM) originated from Vapnik's statistical learning theory (Vapnik,1995, 1997), which has the design of a feedforward network with an input layer, a single hidden layer of non-linear units and an output layer, and formulates the regression problem as a quadratic programming (QP) problem. SVM estimates a function by nonlinearly mapping the input space into a high-dimensional hidden space and then running the linear regression in the output space. Thus, the linear regression in the output space corresponds to a non-linear regression in the low-dimensional input space. The theory denotes that if the dimensions of feature space (or hidden space) are high enough, SVM may approximate any non-linear mapping relations. As the name implies, the design of the SVM hinges upon the extraction of a subset of the training data that serves as support vectors, which represent a stable characteristic of the data.

The SVM regression function is formulated as follows

$$y = w\,\phi(x) + b, \tag{5}$$

where $\phi(x)$ is called the feature which is non-linear mapped from the input space x.

The coefficients w and b are estimated by minimising

$$R(C) = C\frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(d_i, y_i) + \frac{1}{2}\|w\|^2 \quad (6)$$

$$= |d - y| - \varepsilon \; |d-y| \geq \varepsilon \; L_\varepsilon(d,y) \quad (7)$$

$= 0$ , otherwise

where both C and ε are prescribed parameters. The first term $L_\varepsilon(d, y)$ is called the ε-intensive loss function. The $d_i$ is the actual price in the ith period. This function indicates that error below ε are not penalized. The term C $\frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(d_i, y_i)$ is the empirical error. The second term $\frac{1}{2}\|w\|^2$ measures the flatness of the function. C evaluates the trade-off between the empirical risk and the flatness of the model.

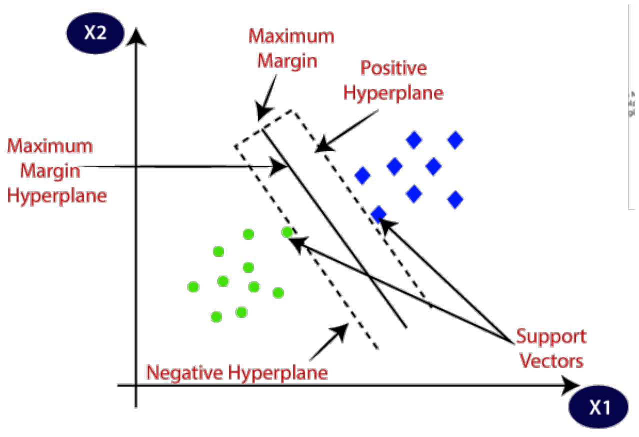

**Fig. 3.** Support Vector Machine diagrammatic representation

Introducing the positive slack variables ζ and ζ* which represent the distance from the actual values to the corresponding boundary values of ε-tube. Eq. (2) is transformed to the following constrained formation:

Minimize :

$$R(w, \zeta, \zeta^*) = \frac{1}{2}ww^T + C^*\left(\sum_{i=1}^{N}(\zeta_i + \zeta_i^*)\right) \quad (8)$$

Subjected to:

$$w\phi(x_i) + b_i - d_i \leq \varepsilon + \zeta_i^* \quad (9)$$

$$d_i - w\phi(x_i) - b_i \leq \varepsilon + \zeta_i \quad (10)$$

$$\zeta_i, \zeta_i^* \geq 0 \quad (11)$$

$i = 1, 2, 3, \ldots, N$

Finally, introducing Lagrangian multipliers and maximizing the dual function of Eq.(4) changes Eq. (4) to the following form:

$$R(\alpha_i - \alpha_i^*) = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i^*) - \varepsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*) \times (\alpha_i - \alpha_i^*) K(x_i, x_j) \quad (12)$$

with the constraints

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \quad (13)$$

$$0 \leq \alpha_i \leq C \quad (14)$$

$$0 \leq \alpha_i^* \leq C \quad (15)$$

and $i = 1, 2, 3, \ldots, N$

$\alpha_i$ *and* $\alpha_i^*$ are called Lagrangian multipliers. They satisfy the equalities ,

$\alpha_i \times \alpha_i^* = 0$ ,

$$f(x, \alpha, \alpha^*) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) K(x, x_i) + b \quad (16)$$

Here, $K(x, x_i)$ is called the kernel function. The value of the kernel is equal to the inner product of two vectors $x_i$ and $x_j$ in the feature space $\phi(x_i)$ and $\phi(x_j)$, such that

$$K(x_i, x_j) = \phi(x_i) * \phi(x_j)$$

Radial basis function is specified in this study.

## Long Short Term Memory (LSTM)

For the prediction of time series values where there is a significant gap between the data, traditional RNN (Recurrent Neural Network) is not favoured. Therefore, we opt to use the LSTM networks, a more advanced and fascinating neural network model. The acronym LSTM is used to refer to long-short term memory. "Long-way dependability" can be handled by these networks. They are naturally good for long-term memory retention. One cell state and three gates make up the LSTM unit's structure. The LSTM cell can add or delete information from the cell using this gating mechanism. Information can be passed via the network using gates, which are optional. The three gates—input gate, forget gate, and output gate—interact to calculate the cell state.

$$O^t = sigma\left[(W^o * h_{t-1}) + (W^o * X_t) + b^o\right] \quad (17)$$

here Eq.(13) represents calculations by output gate.

$$I^t = sigma\left[(W^I * h_{t-1}) + (W^I * X_t) + b^I\right] \quad (18)$$

here Eq.(14) represents calculations by input gate.

$$F^t = sigma\left[(W^F * h_{t-1}) + (W^F * X_t) + b^F\right] \quad (19)$$

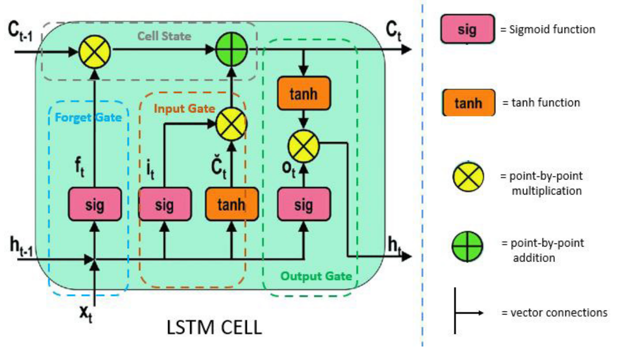

**Fig. 4.** LSTM unit cell structure

here Eq.(15) represents calculations by forget gate.

$$h_t = O^t * sigma_h(C_t) \tag{20}$$

$$C_t = F^t * C_{t-1} + I^t * sigma_C(W^C * X_t + b^C) \tag{21}$$

here Eq.(16) & Eq.(17) represents calculations of current hidden state and current cell state respectively. Where

$I^t$ denotes input gate,

$F^t$ denotes forget gate,

$O^t$ denotes output gate,

$h_t$ denotes current hidden state,

$C_t$ denotes current cell state,

$C_{t-1}$ denotes previous cell state,

$h_{t-1}$ denotes output of previous LSTM cell (time stamp t-1),

$X_t$ denotes current input at timestamp 't',

$b^x$ denotes bias for the respective gates(x)

## Stacked LSTM

A variation on the classic LSTM model, the stacked LSTM stacks multiple LSTM layers vertically on top of one another, with each LSTM layer containing multiple memory cells. Making the LSTM model denser simply means that it will learn the features and data descriptions more precisely from the beginning. In certain studies, researchers discovered that in order to simulate prediction abilities, network depth was more significant than the amount of memory cells in a given layer. When looking for a model that can adapt the hierarchical representation of time-sequence data, the stacked LSTM model is empirically found to be more beneficial. The phases of our model are as follows.

- The dataset is partitioned into the train and test dataset.
- Implement 2 layer stacked LSTM model.
- Choose the right number of LSTM hyper-parameters.
- Fit the model using train dataset
- Check the model performance using validation data.
- Calculate error in prediction results.

## Bidirectional LSTM

The two layers of LSTM structure that make up the bidirectional LSTM neural network are utilised to calculate the hidden vectors from the front to the back and from the back to the front, respectively. These two layers decide the output of the bidirectional LSTM neural network.

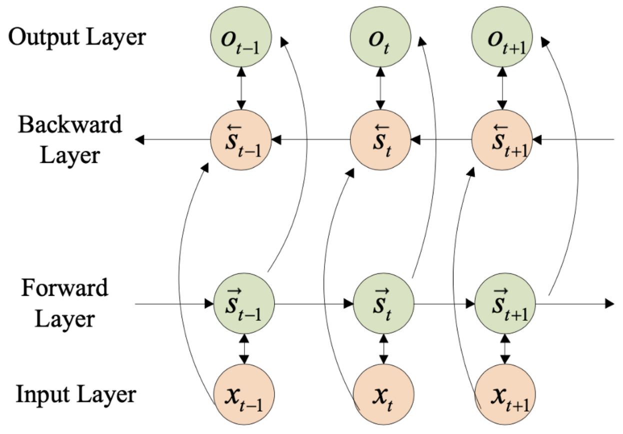

**Fig. 5.** Basic structure of Bi-directional LSTM

The typical feed-forward mechanism neural network is distinct from the bidirectional LSTM neural network. In a bidirectional LSTM, the internal nodes in each layer are not connected to one another. In order to increase the association of single pieces of information in various time series, a directional loop is added in the connection of hidden layers, foregoing information; outcomes are learned and stored in the memory unit. Combining the prior output with the current input yields the neural network's current output. However, due to a lack of delay window width, there will be gradient disappearance and gradient explosion issues as the amount of input data in the time series increases.

The bidirectional LSTM neural network, which is based on the conventional LSTM model, will effectively take into account the front and back correlation of the load data in time series and enhance the model's performance, particularly for the sequence classification problem. The forward layer's input data sequence serves as training data during the training phase, and the backward layer serves as the reverse copy of the input data sequence. In order to prevent the forgetting of the order information, the outcomes of bidirectional structure prediction are affected by the prior input and the subsequent input, increasing the reliance between the training data. Figure 5 demonstrates how the forward layer computes the forward direction from 1 to t while saving the forward hidden layer's output at each instant. The reverse time series is computed by the backward layer, which also records the output of the backward hidden layer at each instant. By merging the respective output values of the forward layer and backward layer at each time point, the output of the bidirectional LSTM neural network is then determined. You may write the bidirectional LSTM neural network as follows:

$$s_t = f\left(Ux_t + Ws_{t-1}\right) \tag{22}$$

$$s'_t = f\left(U'x_t + W's'_{t+1}\right) \tag{23}$$

$$o_t = g\left(Vs_t + V's_t\right) \tag{24}$$

where $s_t$ is the state variable of the hidden layer at time t, $o_t$ is the state variable of the output layer at time t, $s'_t$ is the state variable of the reverse hidden layer at time t, $x_t$ is the input vector, g and f are activation functions, V, W, and U are the weight matrix from the hidden layer to the output layer, the hidden layer, and the input layer to the hidden layer, and V′, W′, and U′ are the corresponding reverse weight matrix. The state weight matrix of the forward layer and the backward layer is not shared information between the two. The forward layer and backward layer are calculated in turn and give the result of each time. The final output $o_t$ depends on the sum of the forward calculation result $s_t$ and the reverse calculation result $s'_t$.

### Accuracy Measures:

Mean Absolute Percentage Error:

$$MAPE = \frac{1}{h}\sum_{s=1}^{h}\left|z_s\right| / f_s \times 100$$

where $f_s$ is the time series, *h* is the forecast horizon, $z_s$ is the residual of the time series and $z_s = f_s - \hat{f}_s$ where $\hat{f}_s$ is the predicted value for time s.

Root Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{h}\sum_{s=1}^{h}\left(z_s\right)^2}$$

where $z_s$ denotes the residual and h is the forecast horizon.

## 4. RESULTS AND DISCUSSION

The descriptive statistics of our datasets have been highlighted in Table 1. Initially, datasets were non-stationary and first differencing has been applied to make them stationary. The Table 2 given below shows that the two tests for stationarity of data give significant results for the first differenced level of the two onion price series used. Teraesvirta neural network test and White's neural network test have been used to detect non-linearity in data. Null hypothesis in both the test assumes that the series is linear. As we find out that the results are significant, therefore both the time series data in our study are non-linear (Table 3). The datasets have been decomposed into its components to check seasonality. Decompose() function of preinstalled "stats" package and stl() function from "forecast" package in R software indicated that data sets are non-seasonal in nature.

**Table 1.** Descriptive Statistics of weekly price series of Onion (2010-2018)

| Statistics | Azadpur market | Kolhapur market |
|---|---|---|
| Observations | 469 | 412 |
| Mean (Rs.) | 1296.90 | 1192.50 |
| Median (Rs.) | 1063.00 | 908.30 |
| Maximum (Rs.) | 5355.00 | 4750.00 |
| Minimum (Rs.) | 440.20 | 400.00 |
| Standard Deviation (Rs.) | 836.89 | 799.62 |
| Skewness | 2.03 | 2.05 |
| Kurtosis | 4.36 | 4.60 |

*Note: Onion price (Rs./quintal)*

**Table 2.** Stationarity test for Onion weekly price series

| Data | Series | ADF | | PP | |
|---|---|---|---|---|---|
| | | t-statistic | Prob. | t-statistic | Prob. |
| Onion | Azadpur | -4.44 | 0.01 | -24.46 | 0.03 |
| | Kolhapur | -3.99 | 0.01 | -33.46 | 0.01 |

**Table 3.** Non-linearity test for Onion weekly price series

| Data | Series | Teraesvirta neural network test | | White's neural network test | |
|---|---|---|---|---|---|
| | | Chi square | P value | Chi square | P value |
| Onion | Azadpur | 34.25 | <0.01 | 14.41 | <0.01 |
| | Kolhapur | 13.24 | <0.01 | 16.63 | <0.01 |

## GARCH

Fitting auto ARIMA Model in R to our data series gave ARIMA (3,0,2) as fit to the Azadpur onion weekly price series and ARIMA(1,0,1) for Kolhapur onion weekly price series. We cross checked the ARIMA parameters by checking AIC values for different combinations of parameter p and q and found out the mentioned ARIMA models to be the best fit (Table 4).

**Table 4.** Identifying the ARIMA parameters based on AIC values for Azadpur and Kolhapur (in brackets) dataset

| Parameters | q=0 | q=1 | q=2 | q=3 |
|---|---|---|---|---|
| **p=0** | 7460 (6495) | 6925 (6164) | 6671(5996) | 6472 (5925) |
| **p=1** | 6219 (5750) | 6192 (**5749**) | 6191 (5751) | 6183 (5750) |
| **p=2** | 6196 (5749) | 6183 (5751) | 6184 (5753) | 6174 (5749) |
| **p=3** | 6196 (5751) | 6183 (5753) | **6171**(5752) | 6178 (5750) |

Fitting these models to our data set and subsequently forecasting resulted in high RMSE and MAPE values, confirming that the ARIMA cannot model and forecast volatile data efficiently.

Thus, the need of modelling these series with nonlinear models like GARCH was felt. Thus, the ARCH – Lagrange multiplier (LM) test was carried out on the residuals obtained after fitting the ARIMA model on the two series to test whether residuals do in fact remain constant. The results of the test revealed the presence of ARCH effect for both the series (Table 5). The higher Chi-square value for Azadpur series claims that its conditional volatility is more than that of Kolhapur. It can also be inferred that the level of heteroscedasticity present in the data of Azadpur is high.

**Table 5.** ARCH-LM test

| Data | Series | ARCH-LM test | |
|---|---|---|---|
| | | Chi squared | P value |
| Onion | Azadpur | 138.69 | <0.01 |
| | Kolhapur | 26.40 | 0.01 |

### Fitting of GARCH Model

The GARCH model was fitted to both the price series and then forecasting was done. For the Azadpur market price series, the AR(3)-GARCH(1,1) model was identified to be the best model on the basis of in-sample performance and AR(1)-GARCH(1,1) in case of Kolhapur price series. The estimates of the parameters of the GARCH model along with their standard errors in brackets for both series are given in Table 6. The results have revealed that onion price series exhibit a persisting volatility as the sum of alpha and beta are close to one.

**Table 6.** Estimates of the parameters of the GARCH model

| Series | alpha | beta | AIC |
|---|---|---|---|
| Azadpur | 0.39 (0.06) | 0.61 (0.04) | 12.72 |
| Kolhapur | 0.63 (0.07) | 0.37 (0.04) | 13.08 |

Performance of the model in forecasting the prices for last 12 weeks of the year 2018 for both the datasets have been depicted in Figs. 6 and 7.

## TDNN

The issue of finding a parsimonious model is taken into account while selecting the best model for each price series. The parsimonious models not only have the recognition ability but also have the more important generalization ability. Ten hidden nodes in case of Azadpur dataset and six hidden nodes in case of Kolhapur have been used. Sigmoid activation function has been applied to both the data series. After fitting the training data set, out of sample forecasting has been done for twelve weeks. According to the observed MAPE and RMSE values, this model performs slightly better than GARCH. Performance of the model is represented by graphs in figures 6 and 7.

## SVM

Two free parameters (ε and C) and two kernel coefficients (d and $\sigma^2$) have to be selected by users before running the SVM procedure. Tenfold cross validation technique has been employed in our study. The motivation for using cross-validation here is to validate the model on a dataset different from the one used for parameter estimation. In this way we may use the training set to assess the performance of various values of parameters, and thereby choose the best one.

For the SVMs models, three parameters: σ, ε, and C were adjusted based on the validation sets. The parameter sets with the lowest values of MSE were selected for use in the best fitted model. Improper selection of parameters can cause either over-fitting or under-fitting of the training data. ε = 0.1, C=16 have been specified in our SVM model and Radial basis kernel function is employed. Forecast performance of the model has been depicted by graphs in figures 6 and 7.

## LSTM

Different forms of LSTM model are tuned by conducting a set of experiments on different structures. Different number of memory cells and number of epochs are used to train the models to find the best trained structure. The same operations are performed on the different architectures to measure the performance of each designed model. Finally, the parameters of the models with more accuracy have been chosen to proceed in our study.

In our study, the final output is produced by a linear activation function of one dense layer. The various hyper parameters considered for the training of LSTM models are the number of input nodes, number of hidden nodes, number of hidden layers, batch size and the number of epochs. These hyper parameters not

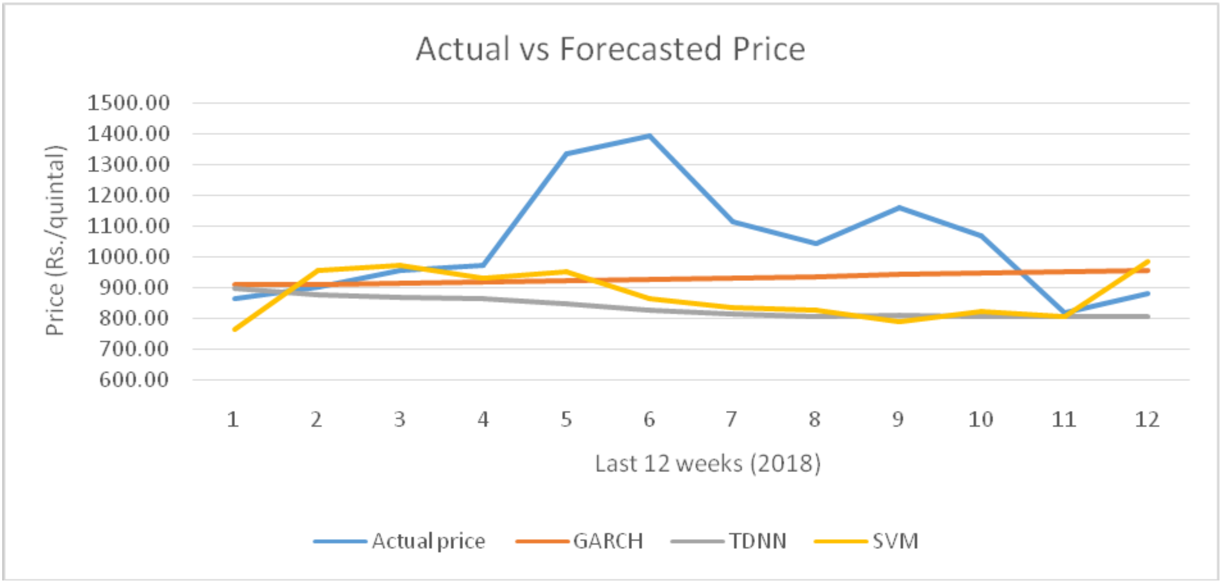

**Fig. 6.** Comparison of actual price and forecasted price by GARCH, TDNN and SVM for Azadpur onion price series (last 12 weeks of 2018)

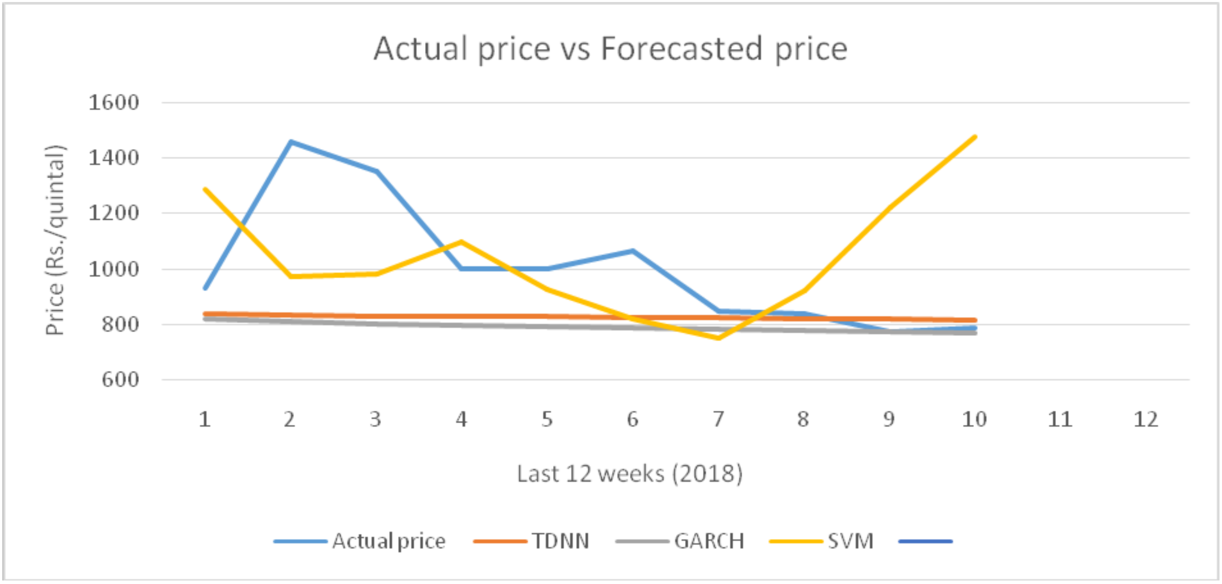

**Fig. 7.** Comparison of actual and forecasted price by GARCH,TDNN and SVM for Kolhapur onion price series (last 12 weeks of 2018)

only regulate the design (architecture and topology) of the model, but also influence the primary parameters of a model such as weights are optimized. Thus, determining the optimized set of different hyper parameters is of great importance for yielding the best model for a given data set. Initial value of training learning rate is 0.001. Further, Rectified Linear Unit (RELU) activation function has been used for all the three forms of LSTM employed in our study.

The training of a LSTM model consists of several forward and backward passes of information for the optimization of loss function to get the optimized weights. A proper choice of optimization algorithm is very important as it affects the overall training results in terms of both efficiency and time. Adaptive moment estimation (Adam) has been used as optimizer which uses different learning rates for different parameters which can also adapt automatically to optimize the loss function and thus performs better than others. The algorithms of the Adam optimizer maintain the moving average of the first and second moments of the gradient to normalize the updates of each parameter (or weight) of the neural network. In addition, in case if the gradients over many iterations are similar, the optimizer gives a push to the parameter updates to get momentum in a certain direction. Further, the optimizer

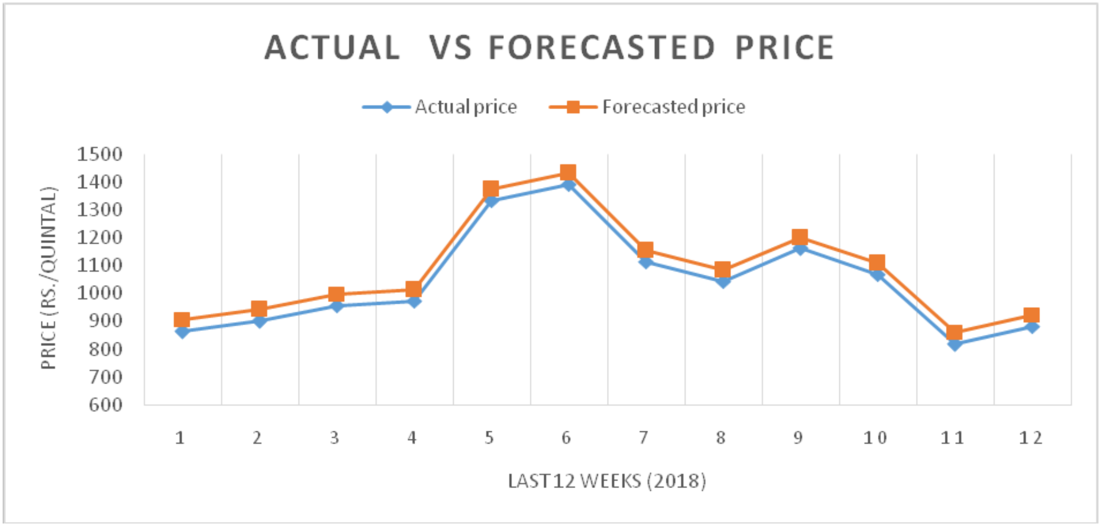

**Fig. 8.** Comparison of actual price and forecasted price by LSTM for Azadpur onion price series (last 12 weeks of 2018)

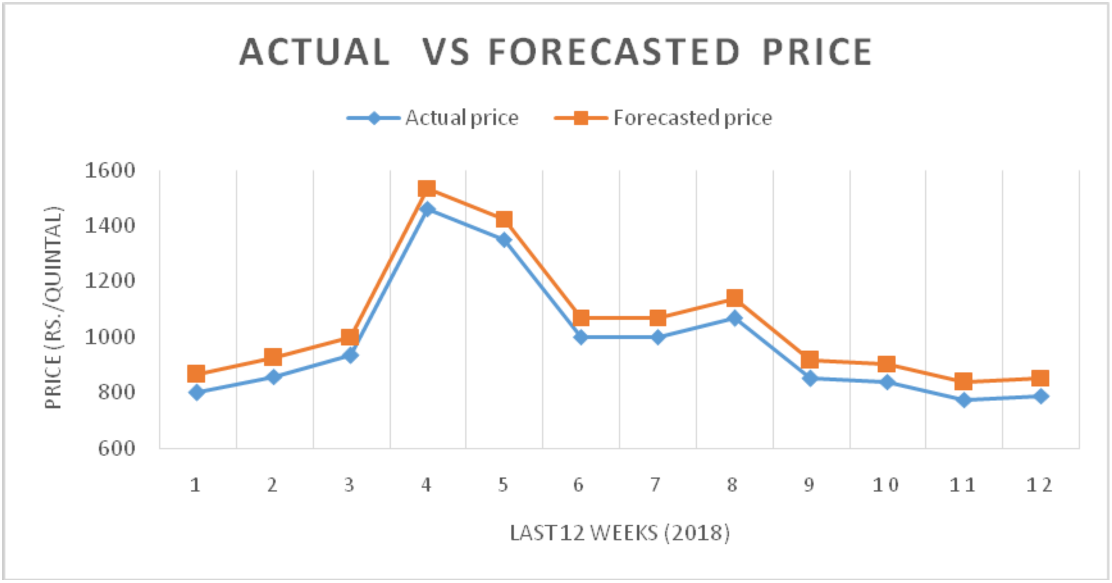

**Fig. 9.** Comparison of actual price and forecasted price by LSTM for Kolhapur onion price series (last 12 weeks of 2018)

has also the ability to correct the bias value besides the weights.

Forecast performance of the model is depicted by graphs in figures 8 and 9.

**Table 7.** Specifications used for LSTM model

| Specifications | Azadpur | Kolhapur |
|---|---|---|
| Number of epochs | 100 | 150 |
| Batch size | 64 | 32 |
| Loss Function | MSE | MSE |

## Stacked LSTM

Two or more layers can be used to build stacked architectures to model more sophisticated data patterns. The architecture of the unidirectional stacked layers does not see information in the future, its hidden states can only learn and process data inputs from the past. Each memory unit receives the output state of its corresponding preceding memory unit and redirects its output state to the next memory unit. Forecast performance of the model has been represented by graphs in Figs. 10 and 11.

**Table 8.** Specifications used for Stacked LSTM model

| Specifications | Azadpur | Kolhapur |
|---|---|---|
| Number of epochs | 150 | 100 |
| Batch size | 32 | 16 |
| Loss Function | MSE | MSE |

## Bi-LSTM

Bidirectional architecture makes use of data in both directions. It consists of two layers; each layer processes the data following different flow direction, from past to future and from future to past. Both layers combine forward and backward contextual information

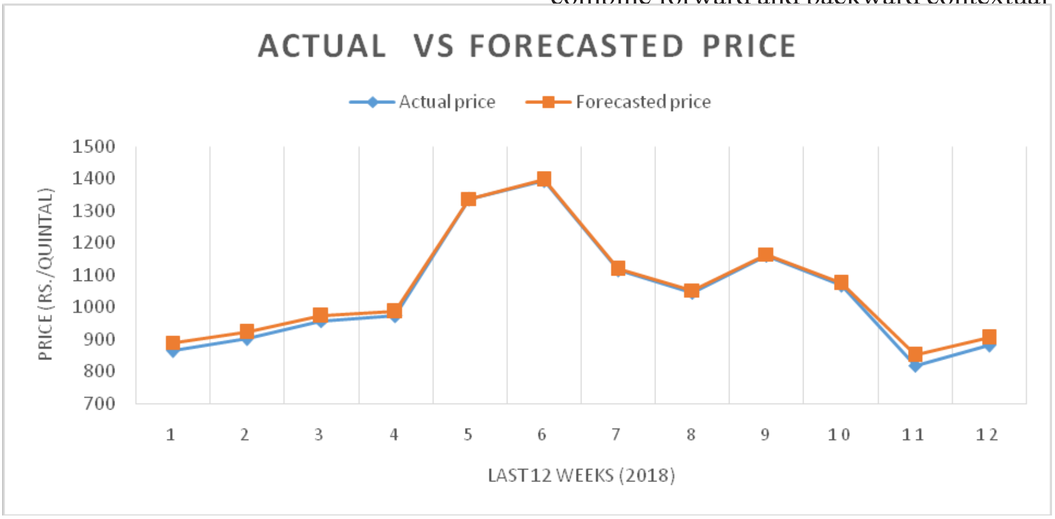

**Fig. 10.** Comparison of actual price and forecasted price by Stacked LSTM model for Azadpur onion price series (last 12 weeks of 2018)

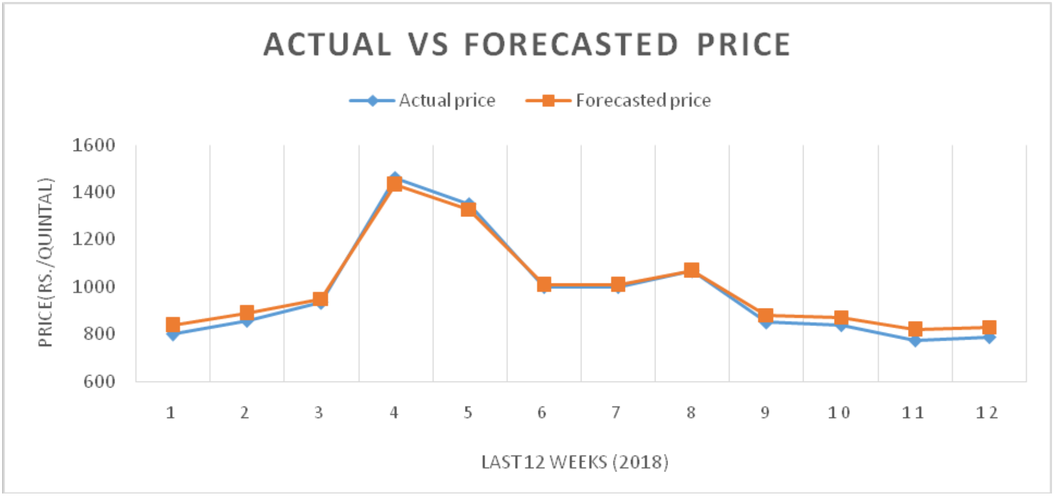

**Fig. 11.** Comparison of actual price and forecasted price by Stacked LSTM model for Kolhapur onion price series (last 12 weeks of 2018)

of data sequence to perform forecasting. Performance of the model is depicted by graphs in figures 12 and 13.

Table 9. Specifications used for Bi-LSTM model

| **Specifications** | **Azadpur** | **Kolhapur** |
|---|---|---|
| Number of epochs | 150 | 200 |
| Batch size | 16 | 8 |
| Loss Function | MSE | MSE |

Figures 14 to 17 shows the contrast between the actual values and fitted values by all the models for the training sets used in both the series. It is only after model fitting (in-sample forecasting) that we go on for out of sample forecasting.

The forecasted values of all the models for the last 12 weeks of the year 2018 have been summarised in Table 10 and 12 for both the datasets and their accuracy measures are presented in Tables 11 and 13**.**

**Table 11.** Comparison of different models by accuracy measures for Azadpur dataset

| **Models** | **Training** | | **Testing** | |
|---|---|---|---|---|
| | **MAPE** | **RMSE** | **MAPE** | **RMSE** |
| GARCH | 7.29 | 203.65 | 16.74 | 223.71 |
| TDNN | 6.41 | 116.79 | 15.63 | 239.37 |
| SVM | 35.01 | 301.97 | 17 | 253.7 |
| LSTM | 1.24 | 40.74 | 3.17 | 40.19 |
| S_LSTM | 4.05 | 69.12 | **1.39** | **15.93** |
| Bi-LSTM | 3.21 | 40.22 | 2.24 | 23.69 |

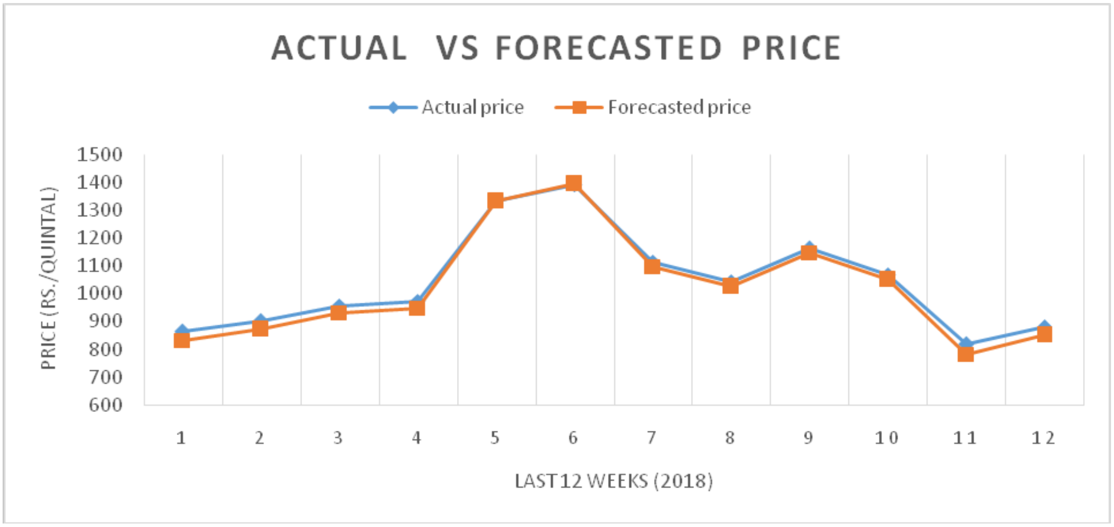

**Fig. 12.** Comparison of actual price and forecasted price by Bi- LSTM model for Azadpur onion price series (last 12 weeks of 2018)

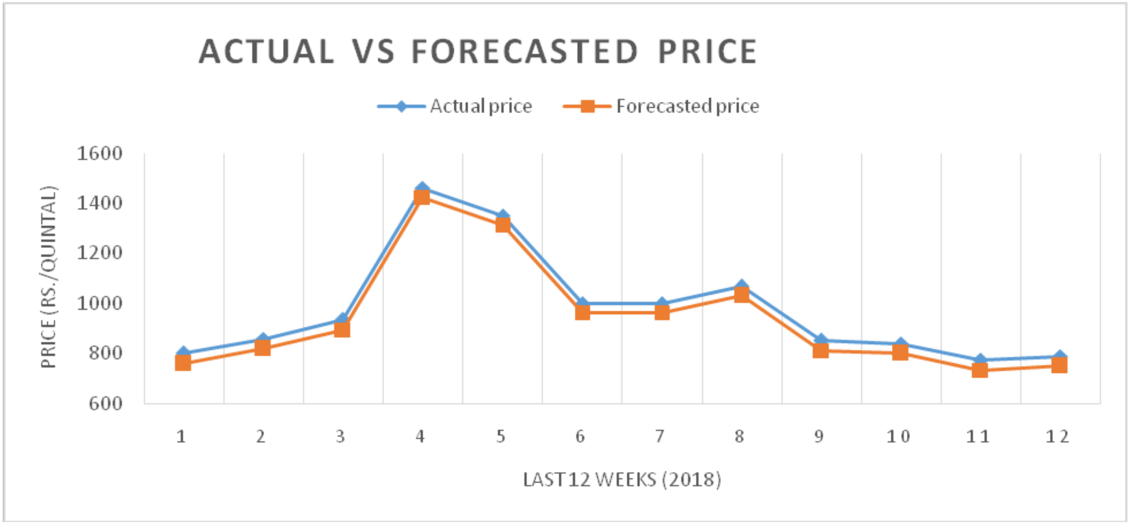

**Fig. 13.** Comparison of actual price and forecasted price by Bi-LSTM model for Kolhapur onion price series (last 12 weeks of 2018)

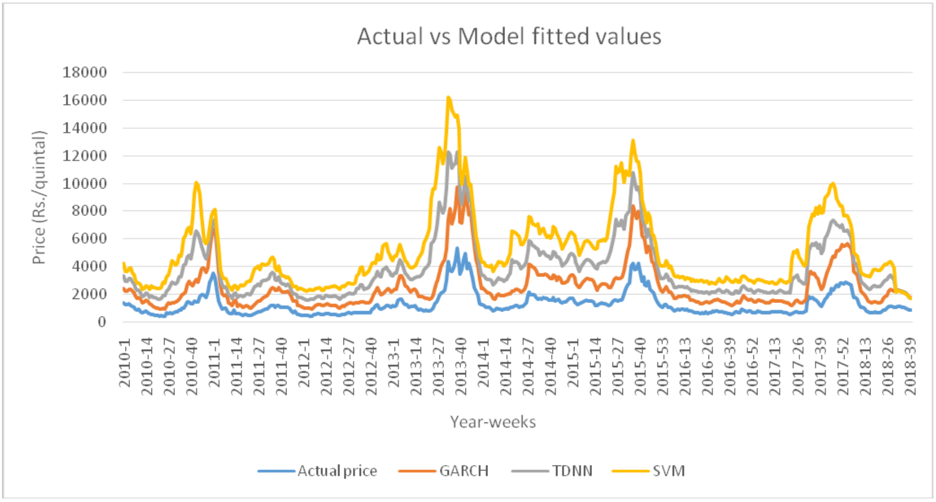

**Fig. 14.** Graphical plot of actual price of Azadpur dataset and fitted values by GARCH, TDNN and SVM

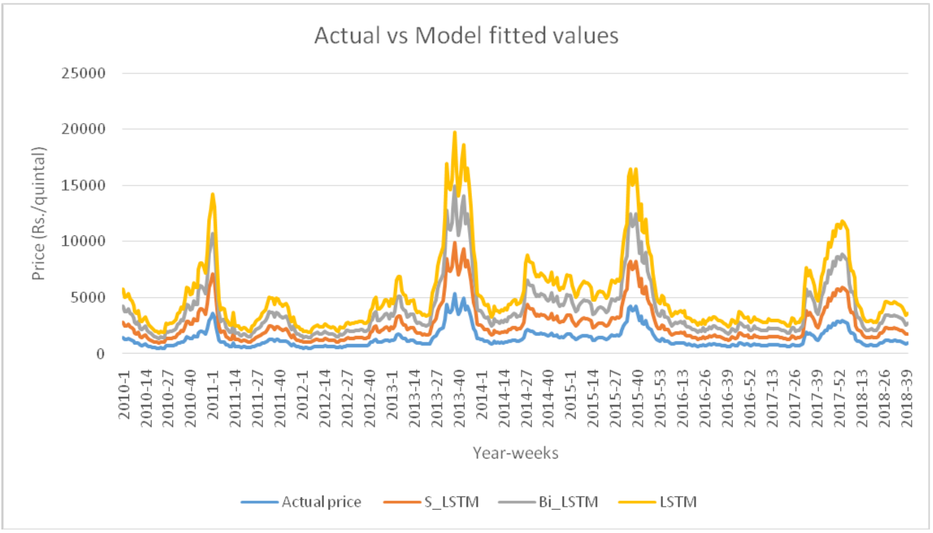

**Fig. 15.** Graphical plot of actual price of Azadpur dataset and fitted values by LSTM, Stacked LSTM and Bi-LSTM

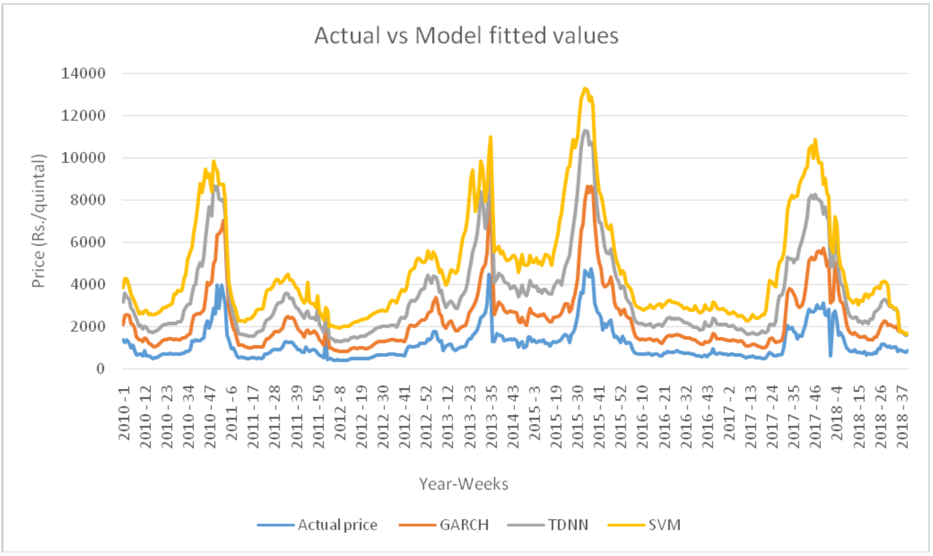

**Fig. 16.** Graphical plot of actual price of Kolhapur dataset and fitted values by GARCH, TDNN and SVM

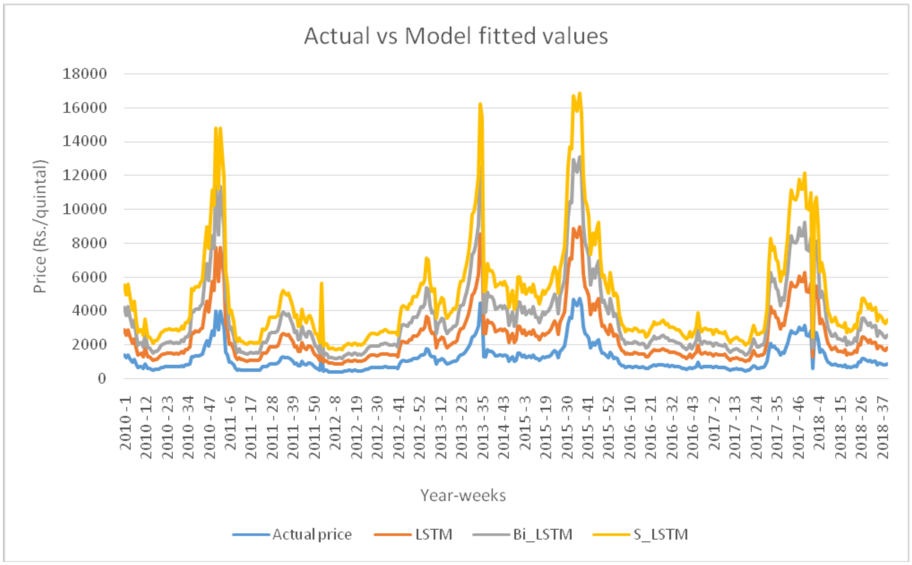

**Fig. 17.** Graphical plot of actual price of Kolhapur dataset and fitted values by LSTM, Stacked LSTM and Bi-LSTM

**Table 10.** Predicted values of all the models for Azadpur data set (last 12 weeks, 2018)

| **Week** | **Actual** | **Stacked LSTM** | **Bi LSTM** | **LSTM** | **GARCH** | **TDNN** | **SVM** |
|---|---|---|---|---|---|---|---|
| **1** | 864.00 | 888.93 | 831.53 | 905.91 | 913.13 | 899.40 | 765.23 |
| **2** | 901.50 | 921.89 | 871.53 | 942.67 | 911.43 | 878.75 | 956.78 |
| **3** | 957.00 | 971.59 | 930.72 | 997.27 | 915.07 | 869.22 | 973.75 |
| **4** | 973.00 | 986.12 | 947.78 | 1013.05 | 918.41 | 865.53 | 932.66 |
| **5** | 1334.33 | 1336.30 | 1332.09 | 1373.27 | 922.74 | 849.21 | 951.84 |
| **6** | 1391.67 | 1395.34 | 1392.81 | 1430.94 | 927.42 | 828.83 | 864.62 |
| **7** | 1113.80 | 1117.71 | 1097.79 | 1152.62 | 932.41 | 814.00 | 834.91 |
| **8** | 1044.33 | 1051.96 | 1023.81 | 1083.61 | 937.55 | 808.70 | 826.56 |
| **9** | 1160.80 | 1163.08 | 1147.81 | 1199.46 | 942.72 | 809.88 | 790.76 |
| **10** | 1067.80 | 1073.99 | 1048.81 | 1106.89 | 947.83 | 807.76 | 822.37 |
| **11** | 818.33 | 849.48 | 782.81 | 861.28 | 952.81 | 806.57 | 808.21 |
| **12** | 881.50 | 904.25 | 850.20 | 923.05 | 957.60 | 807.46 | 987.81 |

**Table 12.** Predicted values of all the models for Kolhapur data set (last 12 weeks, 2018)

| Week | Actual | Stacked LSTM | Bi LSTM | LSTM | TDNN | GARCH | SVM |
|---|---|---|---|---|---|---|---|
| **1** | 800.00 | 838.34 | 760.07 | 863.89 | 837.71 | 843.08 | 1085.72 |
| **2** | 858.33 | 886.93 | 819.29 | 923.69 | 832.37 | 831.51 | 1480.95 |
| **3** | 933.33 | 950.68 | 895.34 | 1000.53 | 839.38 | 821.40 | 1287.85 |
| **4** | 1460.00 | 1430.67 | 1423.67 | 1535.19 | 835.29 | 812.56 | 972.70 |
| **5** | 1350.00 | 1326.78 | 1314.40 | 1424.55 | 831.92 | 804.83 | 981.57 |
| **6** | 1000.00 | 1008.49 | 962.82 | 1068.74 | 831.05 | 798.08 | 1098.42 |
| **7** | 1000.00 | 1008.49 | 962.82 | 1068.74 | 828.55 | 792.18 | 927.53 |
| **8** | 1066.67 | 1067.31 | 1030.16 | 1136.85 | 826.08 | 787.02 | 823.06 |
| **9** | 850.00 | 879.93 | 810.84 | 915.15 | 824.17 | 782.51 | 753.43 |
| **10** | 840.00 | 871.56 | 800.69 | 904.90 | 822.13 | 778.57 | 923.64 |
| **11** | 775.00 | 817.79 | 734.67 | 838.25 | 820.17 | 775.13 | 1222.61 |
| **12** | 790.00 | 830.10 | 749.91 | 853.63 | 818.40 | 772.12 | 1476.53 |

**Table 13.** Comparison of different models by accuracy measures for Kolhapur dataset

| | Training | | Testing | |
|---|---|---|---|---|
| | **MAPE** | **RMSE** | **MAPE** | **RMSE** |
| GARCH | 11.9 | 325.71 | 17.13 | 270.12 |
| TDNN | 11.05 | 250.91 | 14.18 | 256.45 |
| SVM | 44.01 | 368.56 | 34.4 | 379.74 |
| LSTM | 7.18 | 89.01 | 7.12 | 67.68 |
| S_LSTM | 7.14 | 139.23 | **2.78** | **28.08** |
| Bi-LSTM | 5.14 | 95.27 | 4.1 | 38.25 |

## 5. CONCLUSION

Location wise two different datasets on the same commodity have been used in our study to get a comprehensive idea about performance of different time series models, machine learning techniques and deep learning models on different series. Surprisingly, GARCH, SVM and TDNN couldn't capture the volatility properly and do not work according to our expectations in forecasting the prices for the datasets under consideration. In contrary to this, LSTM, Stacked LSTM and Bi-LSTM being deep learning models show excellent performance as they capture the volatility in our data series perfectly and provide forecast with commendable accuracy. For Stacked LSTM, larger RMSE value in case of Kolhapur dataset was found out in contradiction with the other better performing deep learning models, viz., Bi-LSTM and LSTM but the low MAPE value of Stacked LSTM model re-assures us of good accuracy. Moreover, the RMSE of fitted values of GARCH, SVM and TDNN were found to be far more than this method for Kolhapur dataset. The graphical plots of actual price values against the model fitted values have been shown in Figs.14 to 17 which gives an idea about how good fit the deep learning models are than the other three models (GARCH,TDNN and SVM) used in our study. Through this study, it's quite clear that we cannot use any particular model as a universal one for volatility forecasting as forecasting techniques are data dependent approaches. For our two datasets, deep learning models have outperformed GARCH which is a very popular model in dealing with the volatile prices of commodities like onion. Similarly, TDNN and SVM are also reputable approaches for non-linear time series forecasting but haven't worked upto our expectations in dealing with these two datasets. It can also be said that the models didn't perform up to the mark in our case, might work good in case of other datasets because models are data dependent entities at the end. We can also add from our study that deep learning models like LSTM and its variants could be used further for forecasting volatile price series of different agricultural crops like potato and others.

## ACKNOWLEDGMENTS

The first author is grateful to The Graduate School, ICAR-Indian Agricultural Research Institute, New Delhi for providing the requisite facilities to carry out this study. The authors are grateful to the editors and the reviewers for their constructive suggestions/comments that helped to improve the quality of the paper.

## REFERENCES

Althelaya, K.A., El-Alfy, E.S.M. and Mohammed, S. (2018). Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU).*21st Saudi Computer Society National Computer Conference (NCC)* (pp. 1-7). IEEE.

Bhardwaj, S.P., Paul, R.K., Singh, D.R. and Singh, K.N. (2014). An empirical investigation of ARIMA and GARCH models in agricultural price forecasting. *Economic Affairs*, **59(3)**, 415.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, **31(3)**, 307-327.

Cai, C., Tao, Y., Zhu, T. and Deng, Z. (2021). Short-Term Load Forecasting Based on Deep Learning Bidirectional LSTM Neural Network. *Applied Sciences,* **11(17)***, 8129.*

Chen, S., Härdle, W.K. and Jeong, K. (2010). Forecasting volatility with support vector machine-based GARCH model. *Journal of Forecasting*, **29(4)**, 406-433.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, **20(3)**, 273-297.

Engle, R.F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, **50(4)**, 987-1007.

Engle, R.F. and Ng, V.K. (1993). Measuring and testing the impact of news on volatility. *The journal of finance*, **48(5)**, 1749-1778.

Franses, P.H. and Van Dijk, D. (1996). Forecasting stock market volatility using non-linear Garch models. *Journal of forecasting*, **15(3)**, 229-235.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9(8)**, 1735-1780.

Jaiswal, R., Jha, G.K., Kumar, R.R. and Choudhary, K. (2022). Deep long short-term memory based model for agricultural price forecasting. *Neural Computing and Applications*, **34(6)**, 4661-4676.

Jha, G.K. and Sinha, K. (2013). Agricultural price forecasting using neural network model: An innovative information delivery system. *Agricultural Economics Research Review*, **26**, 229-239.

Jha, G.K. and Sinha, K. (2014). Time-delay neural networks for time series prediction: an application to the monthly wholesale price of oilseeds in India. *Neural Computing and Applications*, **24(3)**, 563-571.

Lama, A., Jha, G.K. and Paul, R.K. (2015). Modelling and forecasting of price volatility: An application of GARCH and EGARCH models. *Agricultural Economics Research Review*, **28**, 73-82.

Lee, T.H., White, H. and Granger, C.W. (1993). Testing for neglected nonlinearity in time series models: A comparison of neural network methods and alternative tests. *Journal of Econometrics*, **56(3)**, 269-290.

Pai, P.F. and Lin, C.S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, **33(6)**, 497-505.

Panigrahi, S. and Behera, H.S. (2017). A hybrid ETS–ANN model for time series forecasting. *Engineering applications of artificial intelligence*, **66**, 49-59.

Teräsvirta, T., Lin, C.F. and Granger, C.W. (1993). Power of the neural network linearity test. *Journal of time series analysis*, **14(2)**, 209-220.

Vapnik, V.N. (1997). The support vector method. In *International Conference on Artificial Neural Networks* (pp. 261-271).

Wang, B., Liu, P., Chao, Z., Junmei, W., Chen, W., Cao, N. and Wen, F. (2018). Research on hybrid model of garlic short-term price forecasting based on big data. *CMC: Comput. Mater. Continua*, **57(2)**, 283-296.

Zhang, G., Patuwo, B E. and Hu, M.Y. (1998). Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, **14(1)**, 35-62.

Zhang, G.P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, **50**, 159-175.