Statistics and Applications {ISSN 2454-7395 (online)}
Volume 22, No. 2, 2024 (New Series), pp 63–90
http://www.ssca.org.in/journal

Price Forecasting of TOP (Tomato, Onion and Potato) Commodities using Hidden Markov-based Deep Learning Approach

G. Avinash[1], Ramasubramanian V.[2], Ranjit Kumar Paul[3], Mrinmoy Ray[3], Shashi Dahiya[3], Mir Asif Iquebal[3], Samarth Godara[3] and B. Manjunatha[1]
[1]*The Graduate School, ICAR-Indian Agricultural Research Institute, New Delhi*
[2]*ICAR-National Academy of Agricultural Research Management, Hyderabad*
[3]*ICAR-Indian Agricultural Statistics Research Institute, New Delhi*

Received: 08 October 2023; Revised: 24 November 2023; Accepted: 28 November 2023

Abstract

Accurate prediction of agricultural prices is crucial due to their complex and nonlinear nature. Due to the perishable nature of TOP (Tomato, Onion and Potato) vegetable produce, price fluctuates based on supply and demand. It is necessary to forecast harvest period TOP prices, so growers can make informed production decisions and also farmers can plan their market situation to enhance their profits. This research introduces novel Deep Learning (DL) models based on hidden states to enhance the precision of TOP price forecasting. The Hidden Markov Model (HMM) is employed to identify hidden states and uncover underlying patterns in TOP price data. The hidden states identified by HMM serve as a feature extraction technique and are utilized in four DL models, *viz*., Multilayer Perceptron (MLP), Recurrent Neural Networks (RNN), Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM). The integration of HMM with DL aims to improve forecasting accuracy compared to HMM and traditional DL models. The models are evaluated using a real dataset from Azadpur Mandi in Delhi, providing practical insights into forecasting accuracy. The performance of the models is evaluated using standard metrics such as Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). Additionally, the Diebold-Mariano (DM) test has been conducted to compare the accuracy of the proposed approach with baseline DL models. The findings demonstrate that the hybrid approach of Hidden Markov (HM) combined with DL models yields superior forecasting performance compared to existing models.

Key words: Diebold mariano; Gated recurrent units; Long short-term memory; Multilayer perceptron; Recurrent neural networks; Vegetable price.

Corresponding Author: Ramasubramanian V.
Email: ram.vaidhyanathan@gmail.com

1. Introduction

Forecasting of prices for any commodity or product needs hardly be emphasized. Effective planning and strategic decision-making are facilitated by precise and timely price information, coupled with accurate forecasting. However, analyzing agricultural commodity prices presents unique challenges when compared to non-farm goods and services due to their vulnerability owing to unforeseen events like droughts, floods, disease and pest outbreaks, as well as factors such as seasonality, demand fluctuations, climate variability, market imperfections, globalization and speculative trading, see Yin *et al.* (2020) and Manogna and Mishra (2021). Moreover, the nonlinear and nonstationary characteristics of price data further complicate the process of price forecasting, see Xiong (2018).

Vegetable-growing farmers in India are not in a comfortable situation despite the significant increase in the production of tomatoes, onions, and potatoes, collectively known as TOP vegetables. While India holds the position of the world's second-largest producer of overall vegetables, with a total production of 137.99 million metric tonnes (MMT) compared to China's 600.01 MMT, recent statistics show that Tomato, Onion and Potato production reached 21.18, 26.64 and 56.17 MMT respectively in 2021–22 (source; `https://www.statista.com`). Unfortunately, the farmers are currently facing various challenges due to overproduction which resulted in distress sales, crop burning, and the unfortunate practice of discarding their produce on the roads, especially during periods of bearish market conditions Guresen *et al.* (2011)Consequently, it becomes essential to address these issues and stabilize prices by providing storage facilities for farmers during bearish times, offering guidance on selling vegetables during inflationary periods, and imparting knowledge on the supply value chain for increasing the value of vegetables for the betterment of the farmers income. In this study, a real dataset from Azadpur mandi in Delhi has been utilized to shed light on these aspects.

This work has been undertaken on the premise that hybridizing Hidden Markov Models (HMMs) with DL models may offer many advantages over classical DL models. HMMs excel at modeling sequential data, capturing temporal dependencies and leveraging limited labelled data while providing interpretability and handling noisy or incomplete data. Incorporating HMM-based hidden states in DL models provides intermediate representations within these states, facilitating training by providing forecasts within each state and at the same time proceeding with forecasting successively by using the information from previous states. Thus, this explicit modeling of sequential dependencies by HMMs offers a structured framework for DL model training which enhances the predictive capabilities. In addition, while DL approaches offer advantages such as manual feature extraction and resource availability, their effectiveness heavily relies on large datasets. This distinguishes DL techniques from traditional machine learning methods. However, there is still uncertainty regarding the specialization and generalization capabilities of DL models compared to conventional methodologies as the former are computationally intensive, demanding speed and high-end computing resources. DL models are often regarded as black-box models, lacking interpretability and transparency in their decision-making processes. Furthermore, DL models are prone to overfitting, especially when dealing with noisy datasets Singh *et al.* (2023) Hence DL models can be challenging to train and fine-tune, requiring expertise in hyperparameter optimization and architectural design. To address this, our proposed approach combines the strengths of both methodologies by combining an HMM with DL to analyze underlying

patterns in the data series with the aim of overcoming challenges such as overfitting and circumventing local minima traps. By combining the strengths of HMMs and DL models, improved data efficiency, better sequential modeling, interpretability and robustness can be achieved.

The rest of the paper is structured as follows. In Section 2, a review of the literature has been studied. In Section 3, related work along with the background knowledge is discussed. In Section 4, an empirical study has been conducted using a real dataset focusing on the TOP price series with results and discussion. Finally, Section 5 concludes the paper by summarizing the findings with suitable remarks and discussing future prospects following a list of references.

2. Review of literature

Extensive research has focused on improving price forecasting through the utilization of diverse and advanced time series models, see Wang *et al.* (2020). These modeling approaches, designed to enhance price forecasting, can be broadly categorized into two main groups: statistical models and artificial intelligence (AI) based models, see Yu *et al.* (2017).

The ARIMA model, introduced by Box and Jenkins in 1970, is widely used in time series analysis, particularly in forecasting financial data, see [Kocak (2017); Adebiyi *et al.* (2014); Ariyo *et al.* (2014); Jarque and Bera (2011); Avinash *et al.* (2022)]. However, its capabilities are limited when it comes to modeling nonlinear data. To overcome this, alternative nonlinear time series models have emerged, including regime-switching models like SETAR model (Mehdizadeh *et al.* (2019)),STAR model [Athanasopoulos and De Silva (2012)] and GARCH model [Lin (2018)].These models capture nonlinearity but often require specific relationships in the data and lack generalization ability, as highlighted by Weron (2014).

To address the challenges posed by complex dependencies and nonlinear relationships in time series data, HMMs were developed on the basis of pioneering work by Baum and colleagues [Baum and Petrie (1966); Baum and Sell (1968); Baum (1972)]and its first application in the formulation of a statistical method of representing speech was made by Rabiner (1989). HMMs assume that the observed data is generated by a Markov process with hidden states, enabling them to capture nonlinearity and temporal dependencies in the data. By modeling these hidden states, HMMs can effectively uncover latent variables and extract essential features such as trend, seasonality, and volatility. However, it is important to note that the applicability of HMMs may vary depending on the characteristics of the time series data. Chaotic patterns with long-range dependencies may not align well with the assumptions of HMMs [Awad *et al.* (2015); Abdollahi and Ebrahimi (2020)]. Additionally, training an HMM model requires a substantial amount of data, which can be challenging in the context of price forecasting due to noisy data and external factors that may impact the model's performance and also they assume Markovian behavior, which may not hold in all scenarios, limiting their ability to capture complex dependencies. HMMs may struggle to model nonlinear relationships and can be sensitive to initial parameter values, affecting their performance. Determining the appropriate number of hidden states is challenging and HMMs lack transparency and interpretability. Additionally, handling continuous or high-dimensional data can be difficult for HMMs, requiring discretization or dimensionality reduction.

To overcome these challenges, Machine Learning (ML) models have gained prominence in financial time series forecasting due to their ability to learn from data, interpretability and lack of assumptions explained by Makridakis *et al.* (2018).Various ML models, including Artificial Neural Networks (ANNs)/ Multilayer Perceptron (MLP) [Haykin (2009)],Support Vector Regression (SVR) [Henrique *et al.* (2018)], Random Forest (RF) [Nti *et al.* (2019)],eXtreme Gradient Boosting (XGBoost) [Basak *et al.* (2019)] and ensemble models such as stacking [Jiang *et al.* (2020)]and bagging [Wang *et al.* (2009)] have been utilized in financial time series forecasting. ML models, being data-driven and adaptable, offer advantages over traditional model-based approaches. However, ANN has certain limitations such as slow convergence to the optimal solution and the risk of overfitting [Wang *et al.* (2016)]. In the absence of domain knowledge, DL excels at feature extraction, outperforming other methods, except for a few feature engineering techniques like the requirement of a substantial amount of labelled training data to achieve optimal performance.

Deep architectures, achieved by adding additional layers, leverage multiple levels of nonlinear processes by increasing model complexity. Deep Neural Networks (DNNs) with more layers can effectively handle complex functions using fewer parameters. These models include the Recurrent Neural Network (RNN), Gated Recurrent Units (GRUs) [Althelaya *et al.* (2018)] and Long Short-Term Memory (LSTM) [Nelson *et al.* (2017); Jaiswal *et al.* (2022); Zaheer *et al.* (2023); Heidarpanah *et al.* (2023); Latif *et al.* (2023)]

In recent research, several notable studies have explored the integration of Hidden Markov Models (HMMs) with various ML/ DL techniques to enhance the accuracy of time series forecasting across different domains. For instance, Chen *et al.* (2019) proposed a novel approach combining a Generative Adversarial Network (GAN) with an Iteratively Refined HMM for completely unsupervised speech recognition. Hassan (2009)combined hidden Markov and fuzzy model for stock market forecasting. Similarly, Hashish *et al.* (2019) developed a hybrid model that leveraged HMMs and optimized LSTM networks to predict Bitcoin prices. Yao and Cao (2020)introduced a neural network-enhanced HMM based structural time series model tailored explicitly for tourism demand forecasting. Building upon these advancements, Peng *et al.* (2021) devised an HMM-LSTM model for proactive traffic prediction in 6G wireless networks. Additionally, Khan *et al.* (2022) investigated the potential of an HM-BiLSTM-based system for event detection and classification, focusing specifically on food intake recognition. These studies collectively highlight the effectiveness of integrating HMMs with deep learning techniques to tackle complex time series forecasting challenges in diverse domains.

This highlights the need for further research in the area of the agriculture domain. In this study, an attempt has been made on the TOP price series from Azadpur Mandi (Delhi) by using HMM to extract relevant features that can be fed separately to MLP, RNN, GRU, and LSTM to improve the accuracy of forecasting. This approach can be beneficial when the underlying system is complex and difficult to model using traditional methods.

3. Material and methods

In this study, five baseline models *viz.*, HMM, MLP, RNN, GRU and LSTM models and the proposed HMM hybridized with the DL models *viz.* HM-MLP, HM-RNN, HM-GRU and HM-LSTM have been fitted. A brief description of the baseline models considered are

given subsequently followed by the proposed methodology of the hybrid models.

3.1. Hidden markov model (HMM)

HMMs are probabilistic models that generate a series of observations (Y) based on a series of underlying hidden states (S). HMMs are commonly employed to model time-dependent data and have found practical use in diverse fields including speech recognition, molecular biology and computer vision, see Ghahramani (2001).

HMMs are built upon two fundamental assumptions. Firstly, HMM assumes that an observation at a particular time t, denoted as Y_t, is generated by an underlying process where the corresponding state, S_t, remains hidden from the observer. Secondly, it assumes that this hidden state S_t follows a first-order Markov property, meaning that the current state S_t, given the previous state S_{t-1}, is independent of all states prior to $t-1$. Likewise, the output of an HMM also adheres to the Markov property. Consequently, the joint distribution of a sequence of hidden states and observations can be factorized as presented by equation (1).

$$P(S_{1:T}, Y_{1:T}) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t) \tag{1}$$

where $P(S_{1:T}, Y_{1:T})$ represents the joint distribution of the sequence of hidden states ($S_{1:T}$) and observations ($Y_{1:T}$). $P(S_1)$ is the initial probability distribution of the first hidden state S_1. $P(Y_1|S_1)$ is the probability of observing Y_1 given the state S_1. $P(S_t|S_{t-1})$ is the transition probability from state S_{t-1} to state S_t. $P(Y_t|S_t)$ is the probability of observing Y_t given the state S_t. Overall, the equation describes how the joint distribution of hidden states and observations in an HMM can be factorized based on the initial state probability, observation probabilities given the states, and transition probabilities between states.

HMM is defined by three key components: A, B, and π, while implicitly determined by the number of observations (N) and the number of hidden states (M). Where A represents the state transition probability $M \times M$ matrix, B represents the probability of the observations $M \times N$ matrix, and π is the initial state distribution. Thus, HMM can be defined as $\lambda = (A, B, \pi)$.

Hidden Markov Models (HMMs) are utilized to address three fundamental problems, which can be summarized as follows:

1. Problem 1: Given the model $\lambda = (A, B, \pi)$, along with a sequence of observations Y, determine the likelihood of the observed data with respect to the given model through the forward-backward or Expectation-Maximization algorithm.
2. Problem 2: Given the model $\lambda = (A, B, \pi)$, along with a sequence of observations Y, determine the optimal sequence of hidden states that underlie the Markov process through the Viterbi algorithm.
3. Problem 3: Given a sequence of observations Y, estimate the parameters of the model, namely A, B, and π, through the Baum-Welch algorithm.

In this study, our approach involves constructing an HMM based on a given sequence of observations. Subsequently, by calculating the likelihood of the data and determining

the optimal sequence of hidden states through the Viterbi algorithm, following the standard methodology, which can be found in, Giudici and Abu Hashish (2020).

3.2. Multilayer perceptron (MLP)

ANN is a mathematical model inspired by the human brain's information processing and analysis capabilities, used to solve a wide range of nonlinear problems. ANN offers advantages such as parallel processing, learning from experience (dataset) and the ability to approximate various functions with high accuracy. It finds applications in forecasting and classification tasks, with the Multilayer Perceptron (MLP) being the most well-known ANN model. MLP is particularly popular for time series forecasting [Aizenberg *et al.* (2016)].Typically, an MLP consists of an input layer, hidden layer(s), and an output layer, with neurons connected by weighted links. The mathematical equations describing the neural network are represented by equation (2).

$$\hat{y} = \sum_{j=0}^{s_0} W_{jk} \cdot \xi \left(\sum_{i=0}^{s_h} W_{ij} \cdot x \right) \tag{2}$$

$$\xi(\alpha, x) = \begin{cases} \alpha \cdot (e^x - 1), & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \tag{3}$$

where x and $\hat{y}$ are the input and output of the network, respectively. s_0 and s_h are the sizes of the output layer and hidden layers. W_{ij} are the weights of the connections between the input and hidden layers, and W_{jk} are the weights of the connections between the hidden and output layers. ξ is the Exponential Linear Unit (ELU) presented in equation (3) in its general form when $\alpha = 1$. It becomes the Rectified Linear Unit (ReLU) when $\alpha = 0$.

3.3. Recurrent neural networks (RNNs)

RNNs are a type of neural network that is well-suited for modeling time series data. RNNs use a series of interconnected neurons to model the functional relationship between input features in the recent past and a target variable in the future. By repeatedly learning from a training set of historical data, RNNs can capture the transitions of an internal (hidden) state over time and make more accurate predictions about future events as shown in Figure (1).

However, RNNs have a major limitation: they can suffer from the gradient vanishing problem, where the gradient becomes too small over time and the network is unable to retain information from long-term inputs. This can limit the accuracy of RNNs, particularly when modeling time series data with long-term dependencies. To overcome this problem, other variants of RNNs were developed, including the LSTM and the GRU network.

3.4. Long short term memory (LSTM)

Hochreiter and Schmidhuber (1997)recognized that traditional RNNs were unable to retain important historical information for extended periods of time. To address this issue, they developed the LSTM model, which introduced gate mechanisms to the RNN framework.

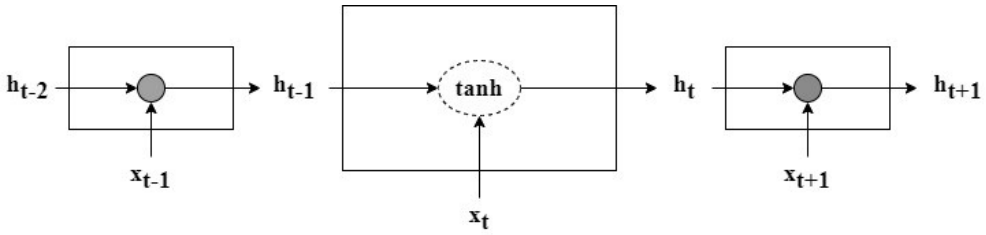

Figure 1: The architecture of RNN model

LSTM is an advanced form of RNN developed specifically for handling sequential data like texts and sentences [Alom *et al.* (2019)]. While a basic RNN is designed to retain and transfer information from one step to the next, it encounters the issue of a vanishing gradient, where long-term information cannot be effectively utilized. Consequently, significant amounts of previous information cannot be stored adequately, resulting in less accurate forecasting. The mathematical formulation of LSTM is represented by equations 4-9

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$
$$\tilde{c}_t = \gamma(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$
$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{7}$$
$$o_t = \sigma(W_0 x_t + U_0 h_{t-1} + b_o) \tag{8}$$
$$h_t = o_t \times \gamma(c_t) \tag{9}$$

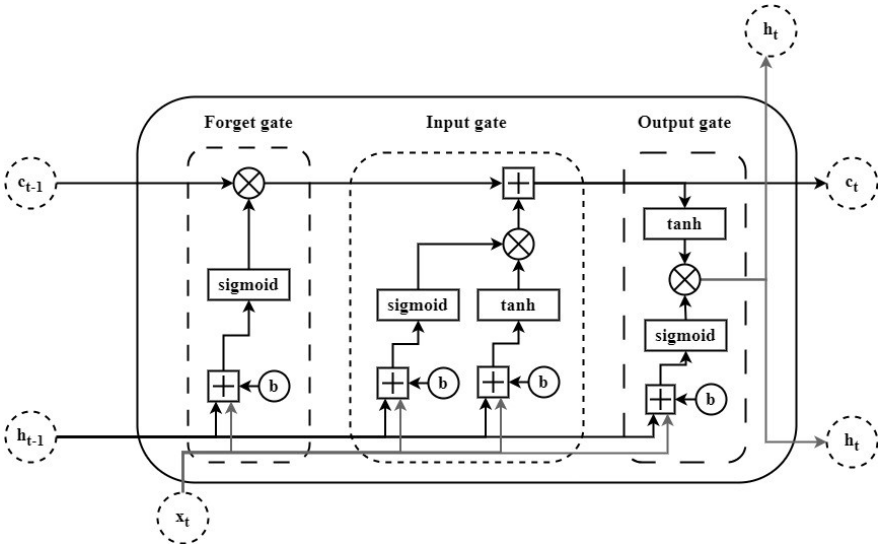

Figure 2: The architecture of LSTM cell

The LSTM mechanism is centred around a cell state, denoted as c_t, which serves as a storage unit for information. This information is regulated through three gates: the forget gate (f_t), the input gate (i_t), and the output gate (o_t). These gates determine whether incoming sequential data should be retained to preserve relevant information for subsequent stages. The forget gate, as indicated by equation 4,decides whether information should be added or omitted. If f_t is close to one (or zero), the information from the input and hidden

state will be preserved (or removed) accordingly. The input gate computes an update to the cell state, evaluating the importance of the input for the subsequent cell. Additionally, the output gate generates the output for the hidden states based on equation 9.Notably, the activation functions used in LSTM are the hyperbolic tangent function (tanh) and the sigmoid function (σ), employed respectively as the activation function as shown in Figure 2.

3.5. Gated recurrent units (GRUs)

GRUs share similarities with LSTM units as they possess a comparable design and can yield similar outcomes in certain cases. However, GRUs differ from LSTM units with the absence of an output gate. Instead, they employ an update gate and a reset gate to control the flow of information into and out of memory as shown in Figure 4. This gating mechanism allows the network to effectively retain information from long-term inputs, enabling more precise predictions in detail explanations by equations 10-13. GRUs offer a potent solution for addressing the vanishing gradient problem in RNNs and find extensive use in various applications including polyphonic music modeling, speech signal processing, handwriting recognition and time series data forecasting. They are particularly beneficial when working with smaller datasets compare to LSTM.

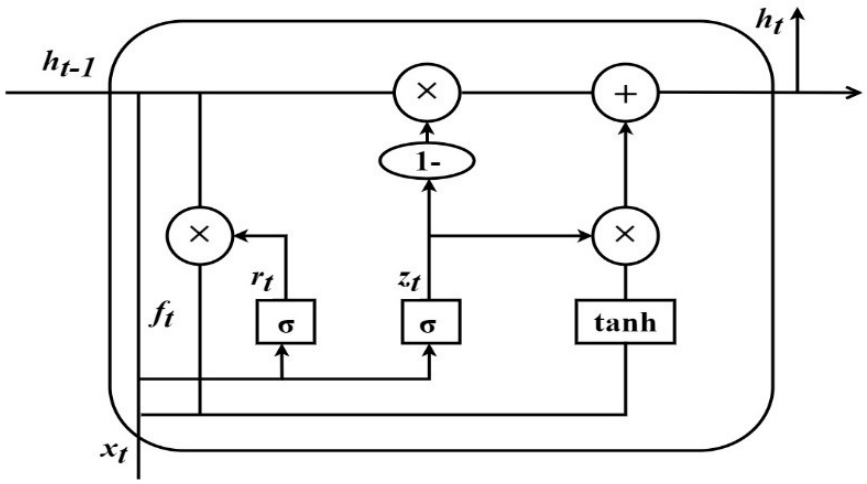

Figure 3: The architecture of GRU cell

The process can be described as:

$$Z_t = \sigma(x_t w^z + h_{t-1} U^z + b_z) \tag{10}$$

$$r_t = \sigma(x_t w^r + h_{t-1} U^r + b_r) \tag{11}$$

$$\tilde{h}_t = \tanh(r_t \cdot h_{t-1} U + x_t W + b) \tag{12}$$

$$h_t = (1 - Z_t) \cdot \tilde{h}_t + Z_t \cdot h_{t-1} \tag{13}$$

where w^z, w^r, and W denote the weight matrices for the corresponding connected input vectors. U^z, U^r, and U represent the weight matrices of the previous time step and b_z, b_r, and b are biases. The σ denotes the sigmoid function, r_t denotes the reset gate, z_t denotes the update gate and $\tilde{h}_t$ denotes the candidate hidden layer.

3.6. Proposed hidden markov based deep learning modeling

The entire analysis was conducted using Python software, employing the "GaussianHMM" and "TensorFlow" libraries (see Appendix). These provided a user-friendly interface for constructing and training DL models. The experiments were conducted on a system equipped with an AMD Ryzen 7 5700U processor and 8 GB of RAM, which proved sufficient for training and evaluating each DL model. The processing time for each model ranged from 25 to 30 minutes while employing the grid search validation technique.

The proposed methodology is represented schematically in Figure 4. To start with, pre-processing is done on the time series data. For this, normalization is employed to rescale the values of the series between 0 and 1 while preserving their shape for the modal price series. The normalization equation 14 used as follows:

$$Y_t = \frac{X_t - X_{\min}}{X_{\max} - X_{\min}} \tag{14}$$

where $X_{\min}$, $X_{\max}$, and X_t are the minimum, maximum and observation at time t, respectively and Y_t is the rescaled value. In Python, the 'Min-Max Scaler' function of the "Scikit-learn" package is used for this purpose. Thereafter, the data is split into, say, 90% training and 10% testing data subsets. The training dataset is then used for training classical Hidden Markov Models (HMM) and baseline Deep Learning (DL) models, such as Multilayer Perceptron (MLP), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), with optimized hyperparameters obtained through grid search. In addition, the training data is used to fit an HMM and extract hidden states using the Viterbi algorithm, employing grid search cross-validation. These hidden states are then utilized to train the proposed hybrid models, namely HM-MLP, HM-GRU, HM-RNN, and HM-LSTM. Finally, the performance of different models on the time series is evaluated using metrics such as Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). Additionally, the Diebold-Mariano (DM) test is conducted to compare the accuracy of the proposed approaches vis-à-vis baseline DL models and also among themselves.

4. Results and discussion

In the present study, the weekly TOP (Tomato, Onion, and Potato) prices (in Rs. /Quintal) from 01 Jan 2006 to 16 June 2023 (obtained from the Agmarknet; `https://agmarknet.gov.in`) of Azadpur market, Delhi were used, whose time plots are depicted in Figure (5, 6 and 7) for TOP commodity price series. This market situated within the Indo-Gangetic plains is characterized by the latitude and longitude coordinates of approximately 28.7078° N and 77.1676° E. This market holds immense significance as one of Asia's largest wholesale fruit and vegetable markets, serving as a crucial link in the agricultural supply chain. Commodities from various regions across the Indo-Gangetic plains converge at Azadpur Mandi, further highlighting its importance as a major hub for agricultural trade. Its strategic location, extensive infrastructure, and role as a price benchmark contribute significantly to its economic importance.

The summary statistics of the datasets are presented in Table 1. Additionally, the Jarque-Bera test [Jarque and Bera (1987)] and Shapiro-Wilk's test [Shapiro and Wilk (1965)]

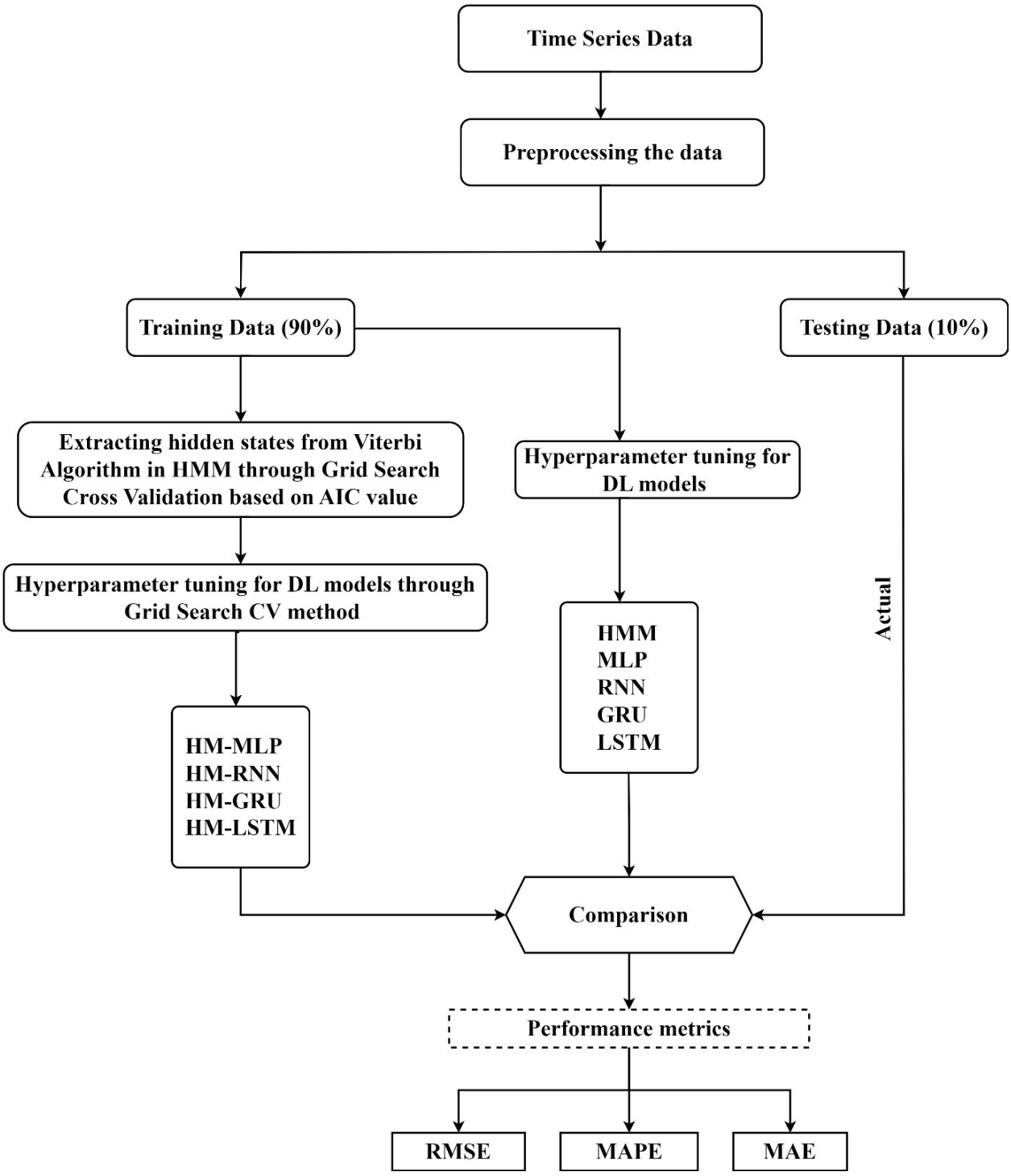

Figure 4: Proposed Hidden Markov (HM) based Deep Learning (DL) modeling

were used to assess the normality of the TOP price series. The tests were significant indicating that all the series are non-normal. Furthermore, the datasets displayed positive skewness but mesokurtic for tomato and potato, while exhibiting leptokurtosis in the case of the Onion price series.

In addition, tests were conducted for the presence of stationarity. The results of the tests (Table 2) reveal weak stationarity under the Augmented Dickey-Fuller (ADF) and Phillips Perron (PP) tests. Subsequently, the nonlinearity of the data series was assessed using the Brock- Dechert-Scheinkman (BDS) test (Table 3). The results highlighted that the weekly TOP price series of all three commodities considered exhibited nonlinear patterns.

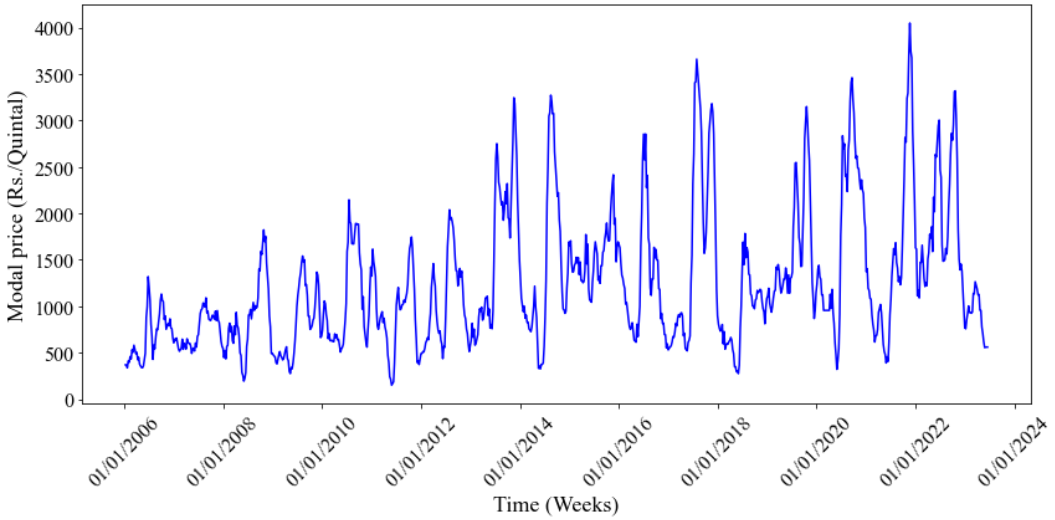

Figure 5: Time plot of weekly Tomato price of Azadpur market

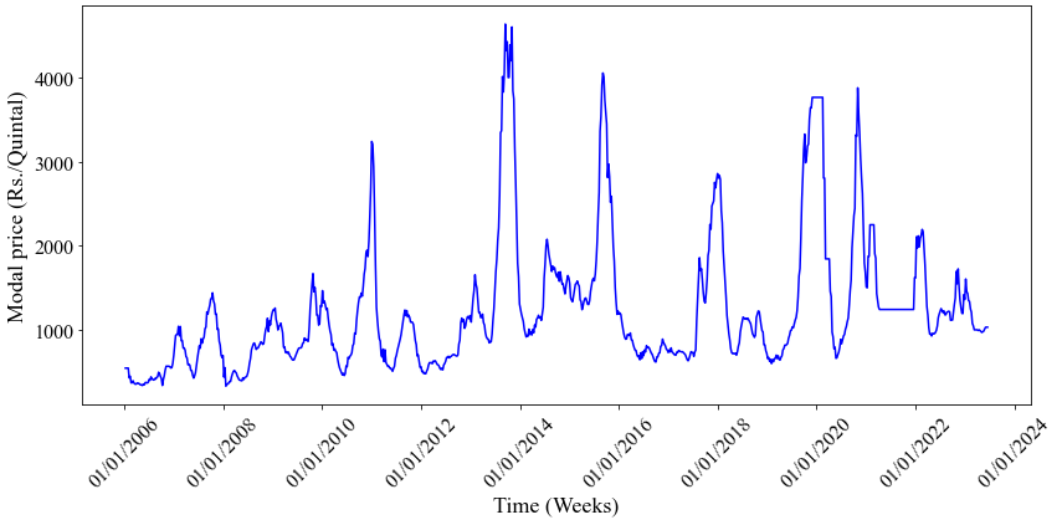

Figure 6: Time plot of weekly Onion price of Azadpur market

The TOP price series comprised 911 observations, which were split into training (90%; 822 data points) and subsequent data points as testing (10%; 89 data points) sets. There were a few missing values in the data series; hence, imputation was done by taking the average of preceding and succeeding observations in the weekly data series. Initially, HMMs were fitted by employing the Viterbi algorithm with grid search cross-validation (2-12 hidden states) to determine the optimal number of hidden states. Results revealed that six hidden states were found for Tomato, and eight hidden states for both Onion and Potato price series,

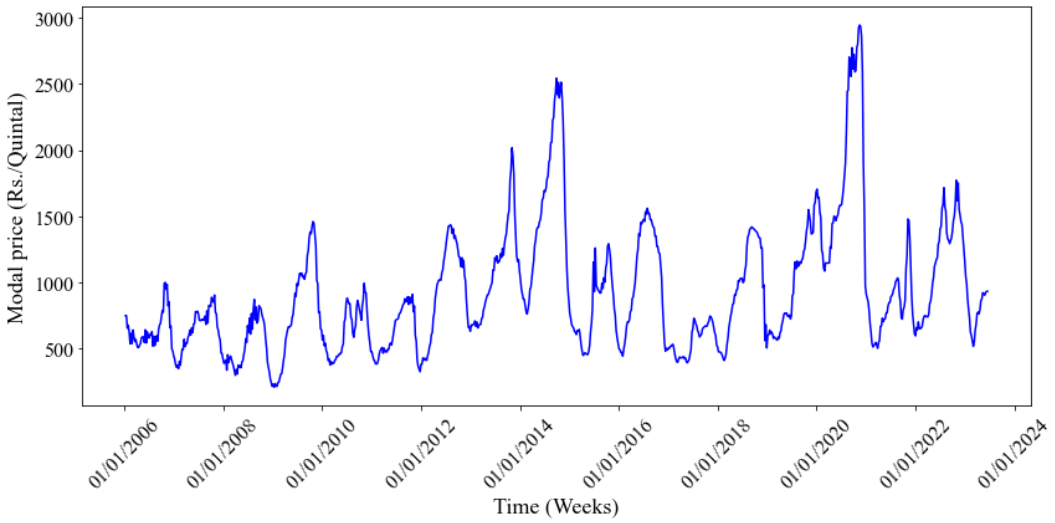

Figure 7: Time plot of weekly Potato price of Azadpur market

Table 1: Descriptive statistics and normality tests of the weekly data series of TOP commodities

Descriptive Statistics	Tomato price series	Onion price series	Potato price series
Mean (Rs. /Quintal)	1244.95	1236.51	916.14
Median (Rs. /Quintal)	1035.06	1031.00	775.92
Maximum (Rs. /Quintal)	4049.75	4638.50	2946.33
Minimum (Rs. /Quintal)	150.08	329.33	207.81
Std. Dev. (Rs. /Quintal)	733.60	799.19	480.28
CV (%)	58.89	64.59	52.39
Skewness	1.19	1.89	1.52
Kurtosis	0.99	3.61	2.91
Jarque-Bera test statistic	253.12^{**}	1039.41^{**}	673.43^{**}
Shapiro-Wilk test statistic	0.89^{**}	0.79^{**}	0.87^{**}

Table 2: Stationarity test of the weekly price series for TOP commodities

Commodities	ADF test		PP test		Conclusion
	Test Statistic	p value	Test Statistic	p value	
Tomato	5.48	< 0.001	4.95	< 0.001	Stationary
Onion	4.56	< 0.001	4.57	< 0.001	Stationary
Potato	4.30	< 0.001	4.37	< 0.001	Stationary

enabling the capturing of complex dynamics and trends to enhance feature engineering in subsequent DL models to be trained (as shown in Figures (8, 9 and 10).

Following the confirmation of stationarity, nonlinearity, feature extraction from HMM, and normalization of the modal price data for TOP, Classical HMM was fitted based on the hidden states obtained by the Viterbi algorithm, and the forecasts were obtained for testing data sets and are shown in Figures (11, 12 and 13). Thereafter, the DL models *viz.*, MLP,

Table 3: Nonlinearity BDS test results with different embedding dimensions for TOP commodities at 0.5, 1, 1.5 and 2 σ respectively

Commodities	Embedding dimension 2		Embedding dimension 3		Conclusion
	Statistics	p value	Statistics	p value	
Tomato	1442.11	<0.001	613663.83	<0.001	Nonlinearity
	1249.30	<0.001	303035.76	<0.001	
	527.43	<0.001	70621.75	<0.001	
	486.41	<0.001	35310.50	<0.001	
Onion	440.29	<0.001	43244.29	<0.001	Nonlinearity
	453.90	<0.001	41327.66	<0.001	
	465.56	<0.001	39211.17	<0.001	
	479.76	<0.001	36843.34	<0.001	
Potato	1248.17	<0.001	326417.51	<0.001	Nonlinearity
	1008.86	<0.001	72434.55	<0.001	
	689.48	<0.001	22456.71	<0.001	
	568.79	<0.001	9869.33	<0.001	

Table 4: Optimal hyperparameters for the various DL models

Model	Batch Size			No. of Epochs			No. of HL			No. of units / HL		
	T	O	P	T	O	P	T	O	P	T	O	P
MLP	64	32	32	57	68	89	2	1	1	32,64	64	128
RNN	128	64	32	76	72	45	1	1	1	32	32	64
GRU	32	64	64	78	56	73	1	1	1	64	64	32
LSTM	64	32	32	87	52	85	1	1	1	128	128	64
HM-MLP	64	16	64	176	147	67	1	1	1	32	32	32
HM-RNN	32	64	32	168	128	112	1	1	1	32	32	8
HM-GRU	16	64	32	52	64	64	1	1	1	64	32	16
HM-LSTM	32	32	32	100	106	65	1	1	1	32	64	16

RNN, LSTM and GRU were also trained. The primary objective of this study is to assess the performance of Hidden Markov hybridized DL (HM-DL) models in forecasting price series.

For training the DL and HM-DL models, hyperparameters play a crucial role as they significantly impact the performance of forecast accuracy to overcome the local minima trap. The batch size in DL models determines how many samples are processed before updating the model's weights. Larger batch sizes can provide more stable gradients but may require more computational resources. The number of epochs specifies how many times the model is trained on the entire dataset. Increasing the number of epochs can potentially improve model performance, but it also increases the risk of overfitting. To mitigate overfitting, early stopping criteria based on mean square error have been applied to select the best weights during training. The number of input units in the model determines the number of variables the model takes as inputs. Having a larger number of input units allows the model to capture more complex relationships in the data but may increase computational costs. A range of hyperparameter values were used in grid search cross-validation on DL models viz., MLP, RNN, GRU, LSTM, and their hybrid HMM cum DL versions, *i.e.*, HM-MLP, HM-RNN,

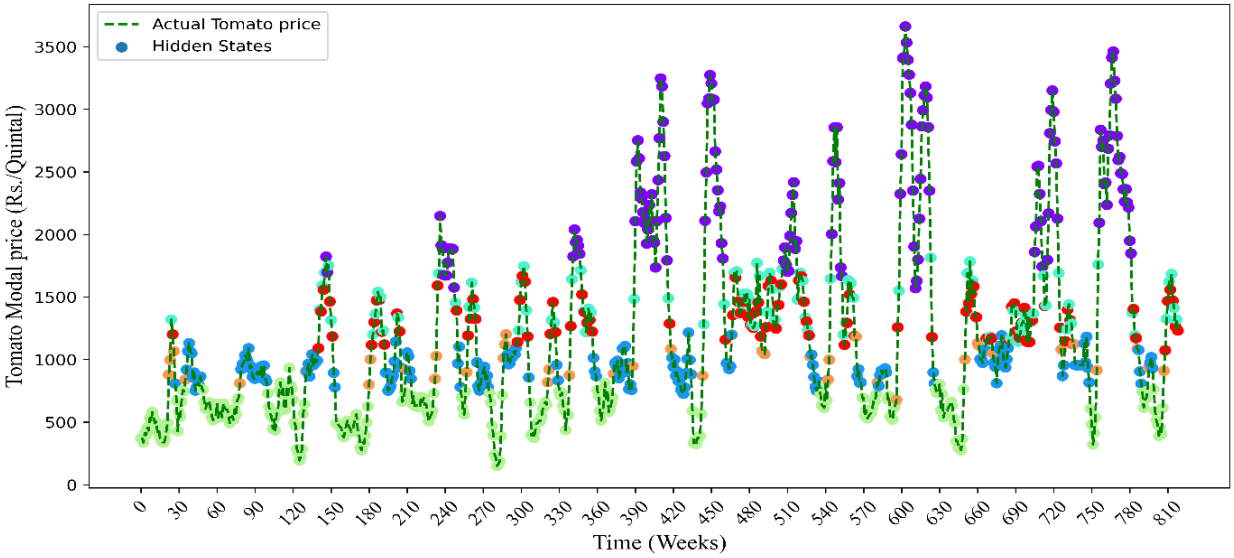

Figure 8: Hidden states obtained from HMM on Tomato price series

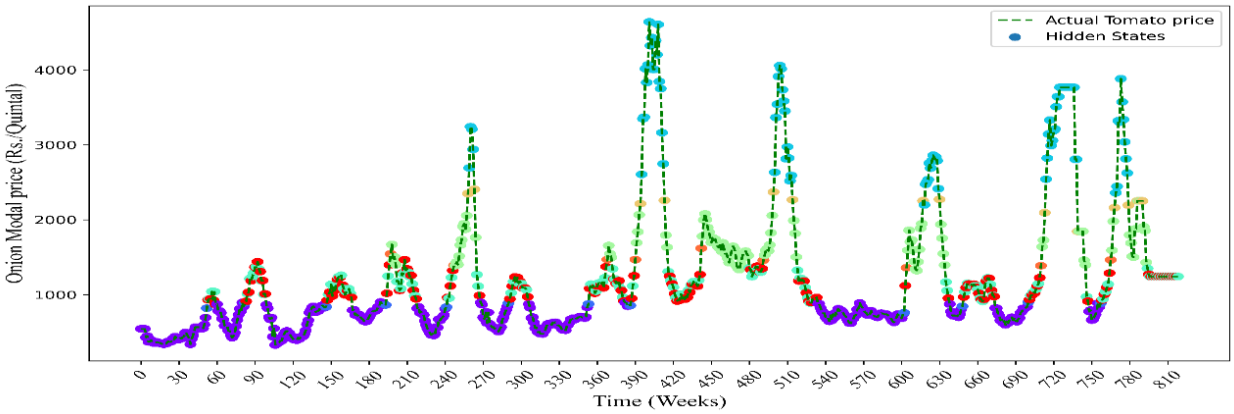

Figure 9: Hidden states obtained from HMM on Onion price series

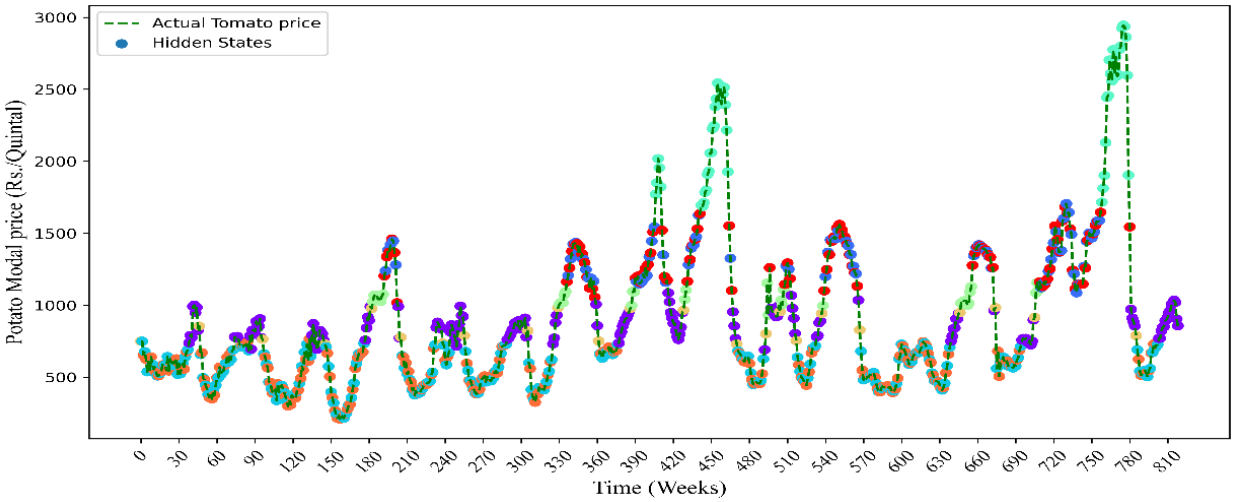

Figure 10: Hidden states obtained from HMM on Potato price series

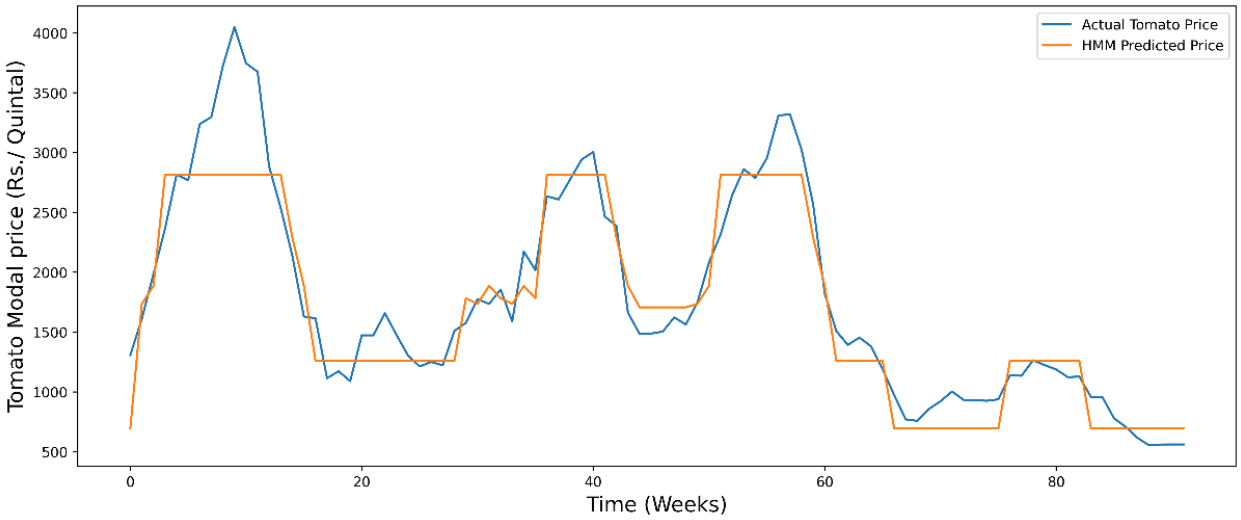

Figure 11: Predicted price by HMM on Tomato price series

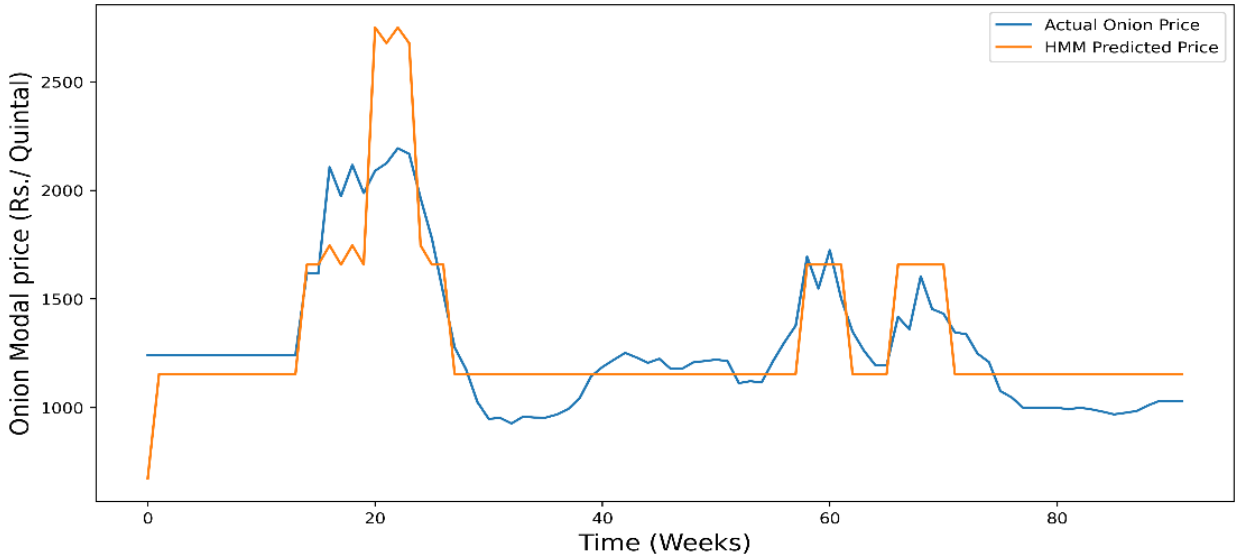

Figure 12: Predicted price by HMM on Onion price series

HM-LSTM, and HM-GRU as follows: the number of lags - fixed as 24 weekly data points for the TOP crop price series; batch size - 8, 16, 32, 64, 128, 256; the number of epochs - 200 with early stopping criteria; the number of hidden layers (HL) - 1, 2, 3; and the number of hidden units - 8, 16, 32, 64, 128, 256, 512, which led to 126 combinations of candidate models for each DL model. For training, each DL model took around 25-30 minutes on average as computing time. The optimal combination of hyperparameters determined is shown in Table (4).

Using these optimal hyper-parameters, DL models trained were utilized for forecasting prices for the test data period. The performance of the models based on RMSE, MAPE and MAE revealed that hybridized HM-DL models perform well as compared to all other models for forecasting of TOP price are shown in Table 5 and Figures (14, 15 and 16).

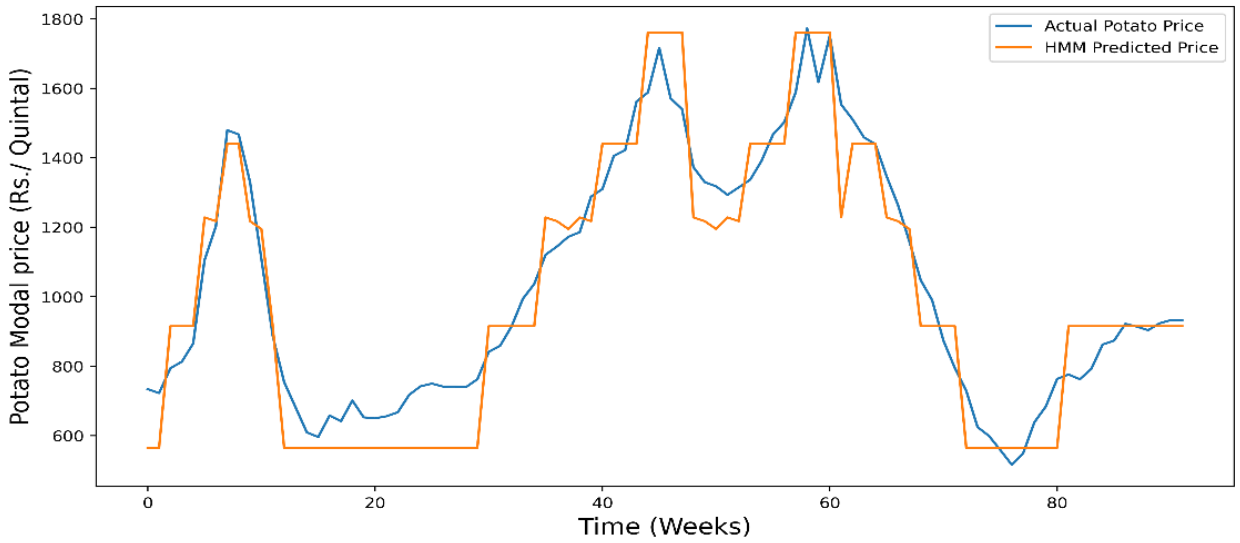

Figure 13: Predicted price by HMM on Potato price series

Table 5 revealed that HMM seems to perform better for forecasting the Potato price series compared to the other two series when considering the results of the testing datasets. Comparison of MAPE values for the baseline DL models with that of the HM-DL models clearly showed that the HM-DL models perform better. By and large, the RMSE of the forecasts for the proposed HM-DL models, namely HM-MLP, HM-RNN, HM-GRU, HM-LSTM, were lower than their corresponding baseline models, namely MLP, RNN, GRU, LSTM, by 9.77–17.50%, 15.02–44.39%, and 7.94–32.60% respectively for the tomato, onion, and potato prices, except for two cases where the baseline DL models seem to be better.

Table 5: Performance of various models on TOP price series data of Azadpur mandi, Delhi

Price series		Tomato			Onion			Potato		
Evaluation measures		RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE
Training set (90%)	HMM	221.19	20.29	180.67	274.55	13.75	159.24	165.38	17.31	127.01
	MLP	190.49	10.72	147.63	215.25	16.40	175.39	77.90	7.44	57.21
	RNN	150.18	10.72	111.24	211.20	12.01	144.80	99.66	8.84	70.43
	GRU	154.46	12.48	118.03	135.32	6.86	85.77	91.29	8.22	63.52
	LSTM	152.32	12.73	117.73	133.73	7.16	87.13	69.89	6.51	50.05
	HM-MLP	201.14	17.10	156.20	146.06	9.62	103.23	101.84	10.00	73.85
	HM-RNN	**128.16**	**9.82**	**95.92**	**106.64**	**5.10**	**63.95**	**64.87**	**5.70**	**44.99**
	HM-GRU	147.05	12.64	116.31	116.98	7.23	79.96	73.95	8.00	58.35
	HM-LSTM	129.85	10.06	97.92	115.57	6.20	73.53	71.56	6.48	49.73
Testing set (10%)	HMM	263.76	11.63	189.88	214.58	13.79	169.36	146.32	10.03	101.63
	MLP	294.16	12.90	220.85	168.42	9.44	123.58	111.66	9.04	86.15
	RNN	216.24	9.00	162.60	159.50	9.94	128.64	110.07	8.52	83.85
	GRU	220.87	9.40	163.11	116.69	6.16	84.78	122.18	9.03	90.95
	LSTM	226.01	10.38	176.56	121.50	6.26	85.58	96.66	7.02	68.33
	HM-MLP	265.43	12.66	207.15	122.24	6.33	86.61	113.09	8.44	83.45
	HM-RNN	240.59	12.55	176.23	**88.69**	**4.62**	**63.21**	**79.58**	**5.80**	**58.86**
	HM-GRU	193.94	8.41	135.98	99.16	5.91	78.68	82.35	6.56	64.22
	HM-LSTM	**186.45**	**8.02**	**133.40**	95.57	4.91	68.04	88.99	**5.79**	60.79

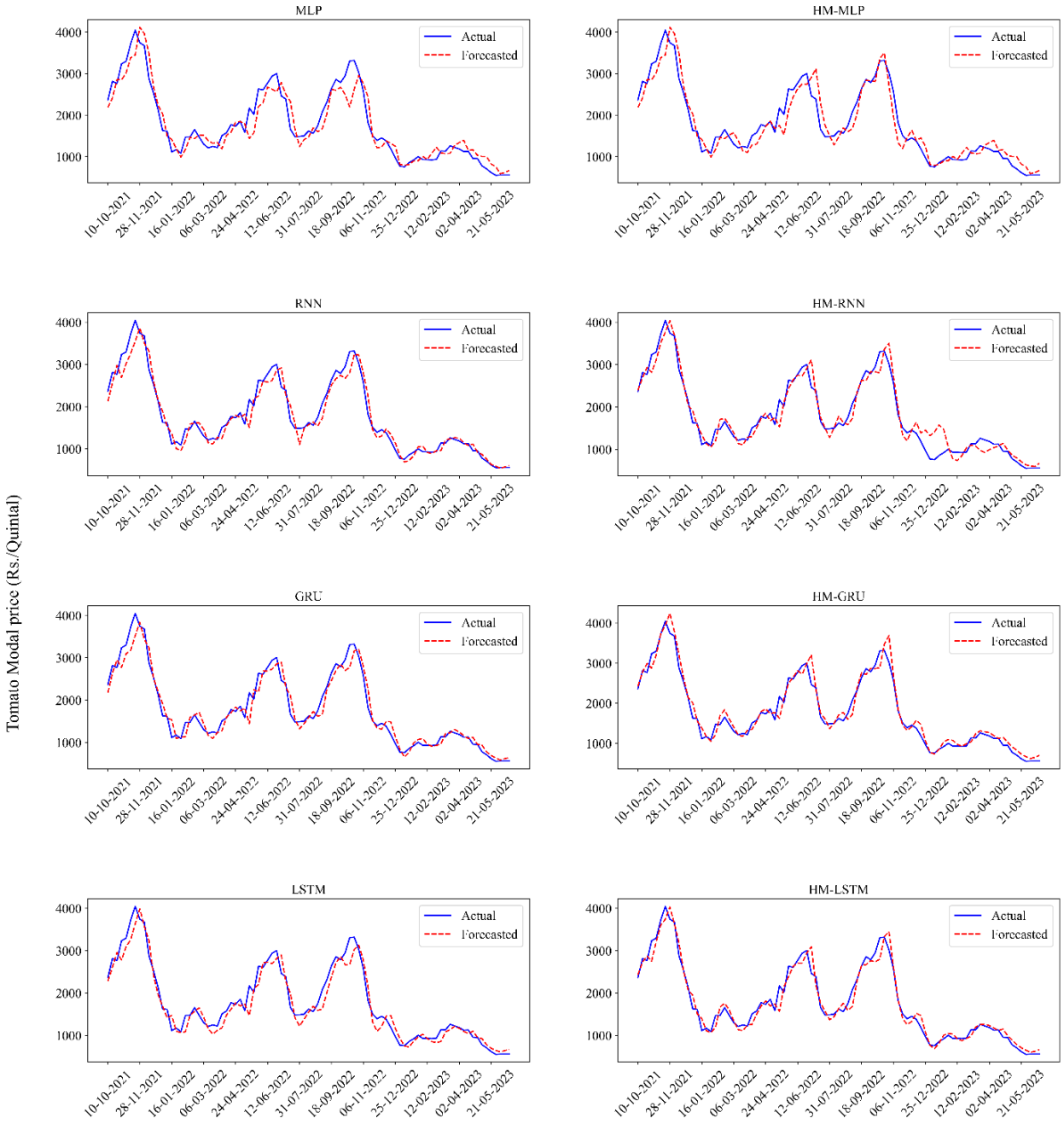

Figure 14: Forecasted Tomato price series obtained from DL and HM-DL models

The proposed HM-RNN model is the best among the HM-DL models proposed. The MAPE of the forecasts for the proposed HM-DL models, namely HM-MLP, HM-RNN, HM-GRU, HM-LSTM, were lower than their corresponding baseline models, namely MLP, RNN, GRU, LSTM, by 0.24–3.55%, 0.25–5.32%, and 0.60–2.72% respectively for the Tomtao, Onion and Potato prices. This reduction is more pronounced for the proposed HM-RNN model with a reduction as high as 5.32%.

It is also emphasized here that the proposed HM-DL models took almost the same computational time while training them when compared to the baseline DL models. The

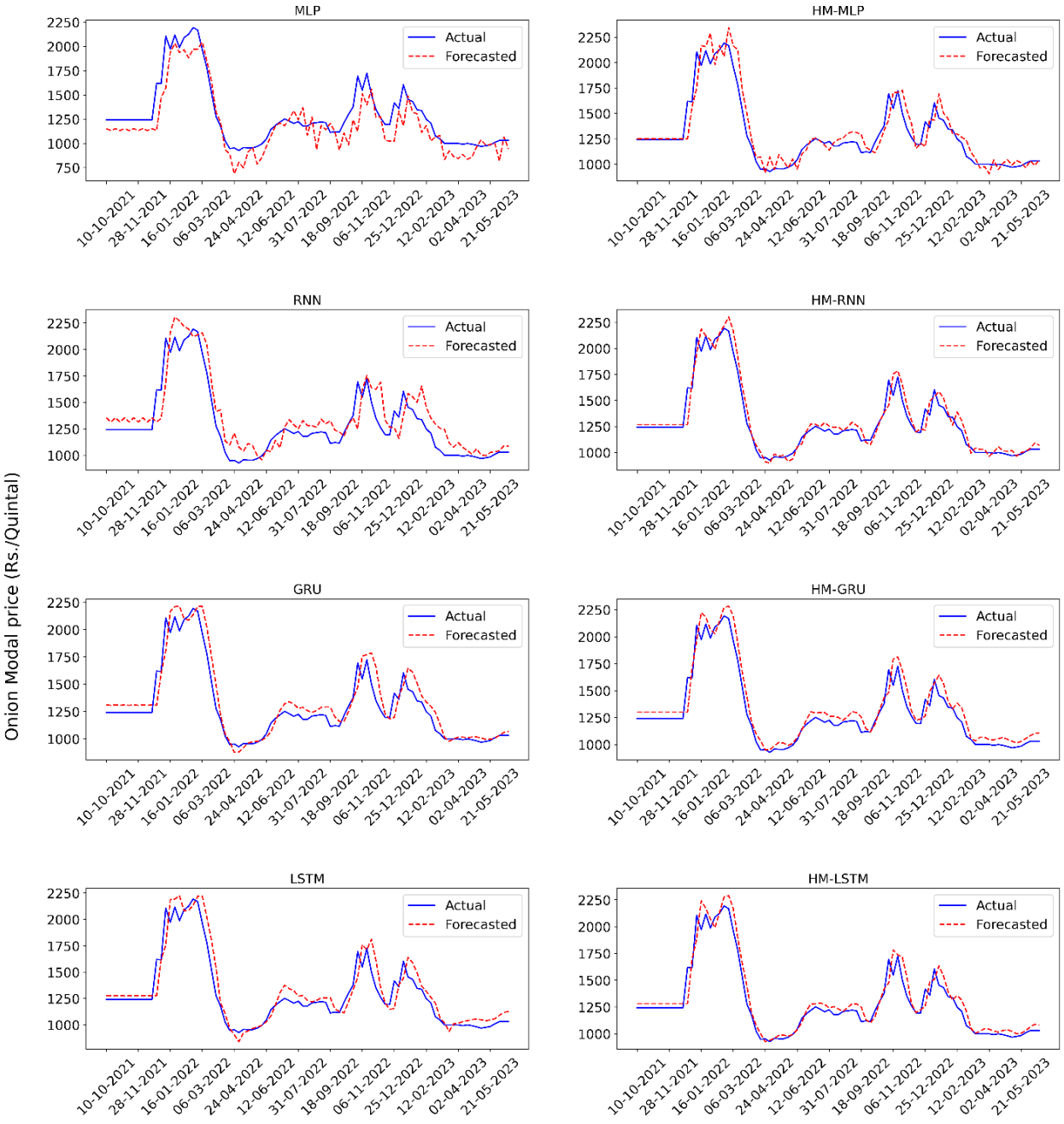

Figure 15: Forecasted Onion price series obtained from DL and HM-DL models

HMM models, when fitted in isolation on the three data series considered, seem to perform inferior as compared to other models, and hence HMM does not capture well the peaks and chaotic patterns present in the price series, even though it could capture the overall trend and latent structure of the data. Moreover, HMM cannot be used for long-term predictions, as the Markovian property assumes a simple conditional dependence of the present on the recent past.

For all the training datasets of the three TOP commodities, consistently HMM-RNN model has been found to be best fitted. On the testing datasets, while for the Onion and

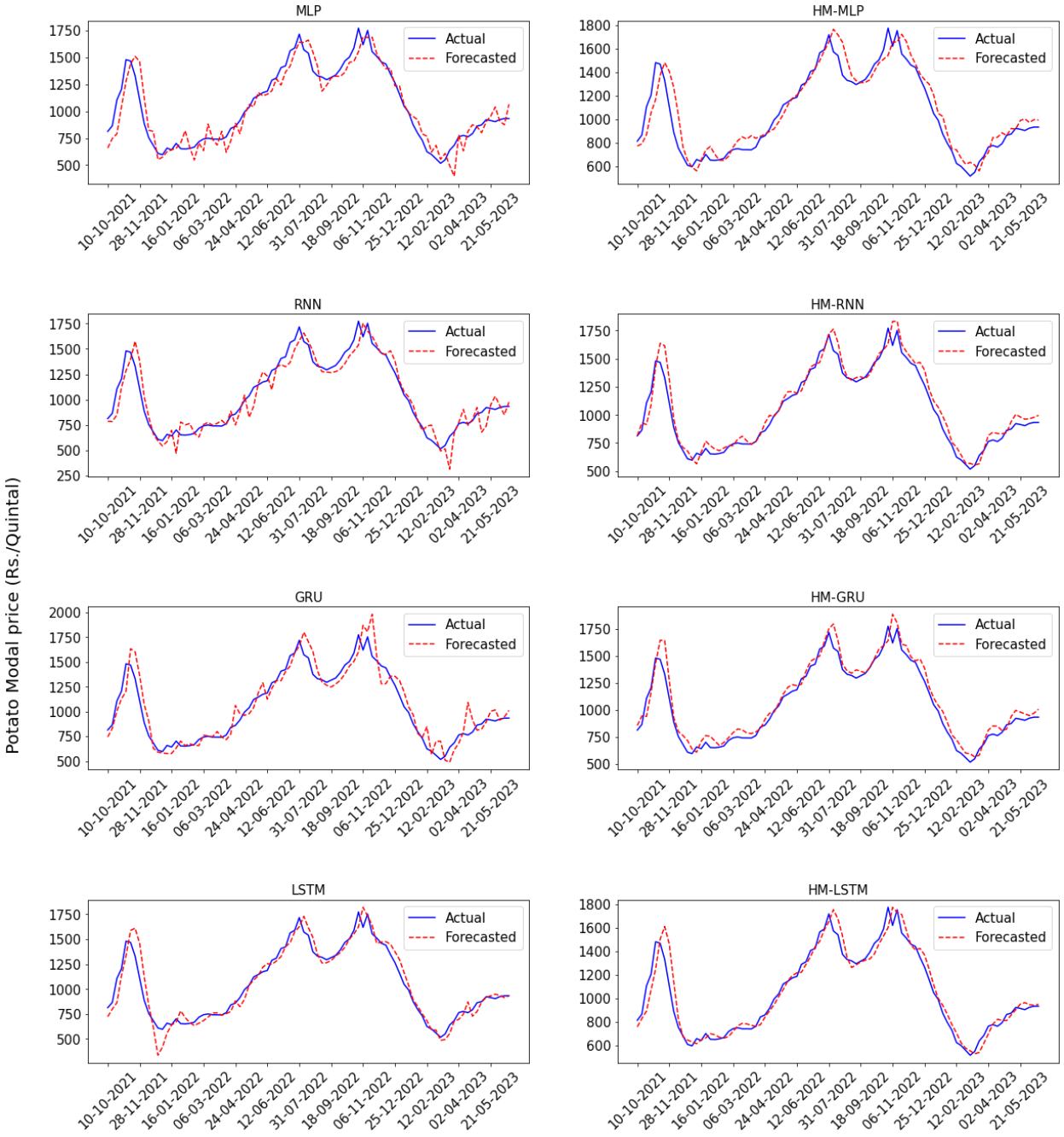

Figure 16: Forecasted Potato price series obtained from DL and HM-DL models

Potato prices, HMM-RNN model performed well as compared to the hybrid and conventional DL models, for the Tomato dataset, HMM-LSTM performed well as compared to HMM-RNN and other models. On further inspection, it has been found that, in the Tomato test dataset, three significant spikes were found with HMM-LSTM also capturing the long memory of the data quite well (as shown in figure 14)while in the other two test datasets (Onion and Potato), only one moderate spike each was present as shown in Figure (15 and 16).

Overall, from the Diebold-Mariano (DM) tests in Figures (17, 18 and 19), it can be

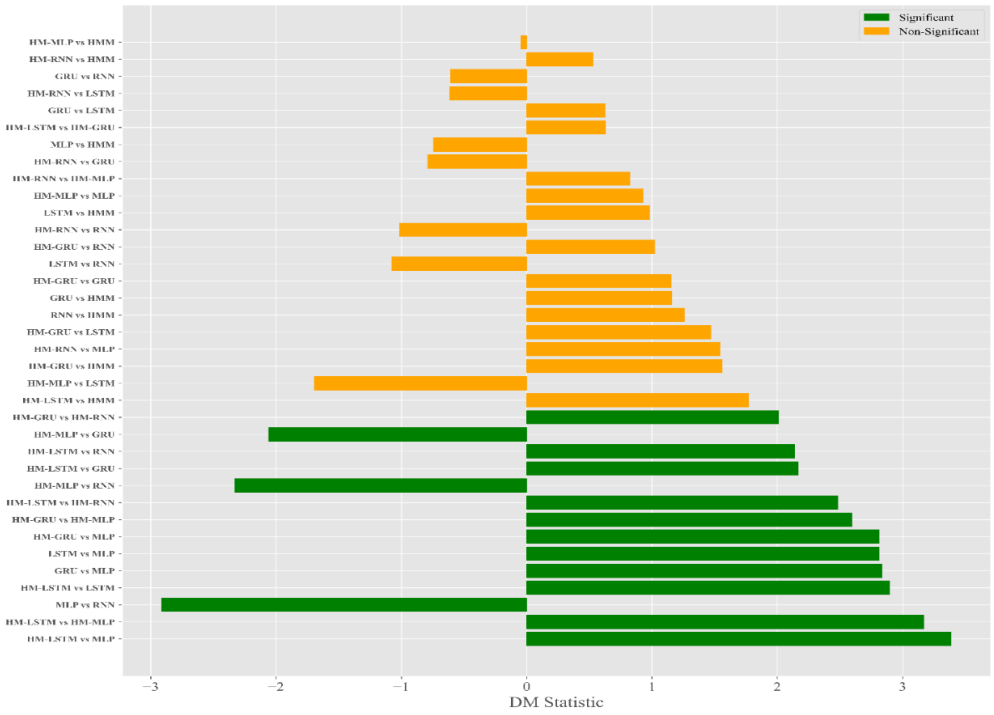

Figure 17: Various forecasting model comparison using DM test on Tomato price series

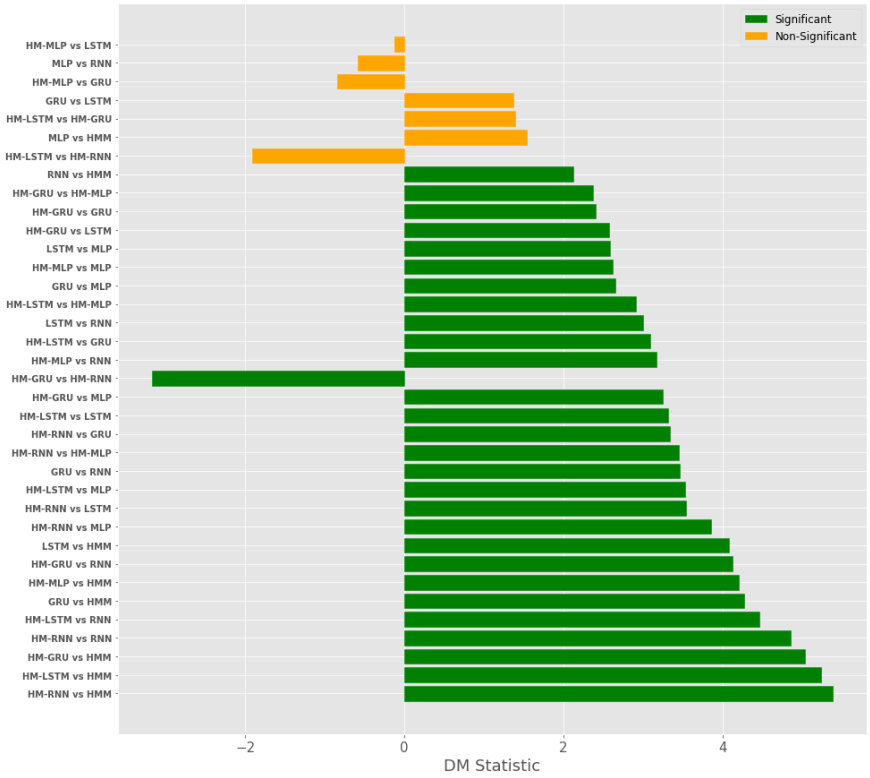

Figure 18: Various forecasting model comparison using DM test on Onion price series

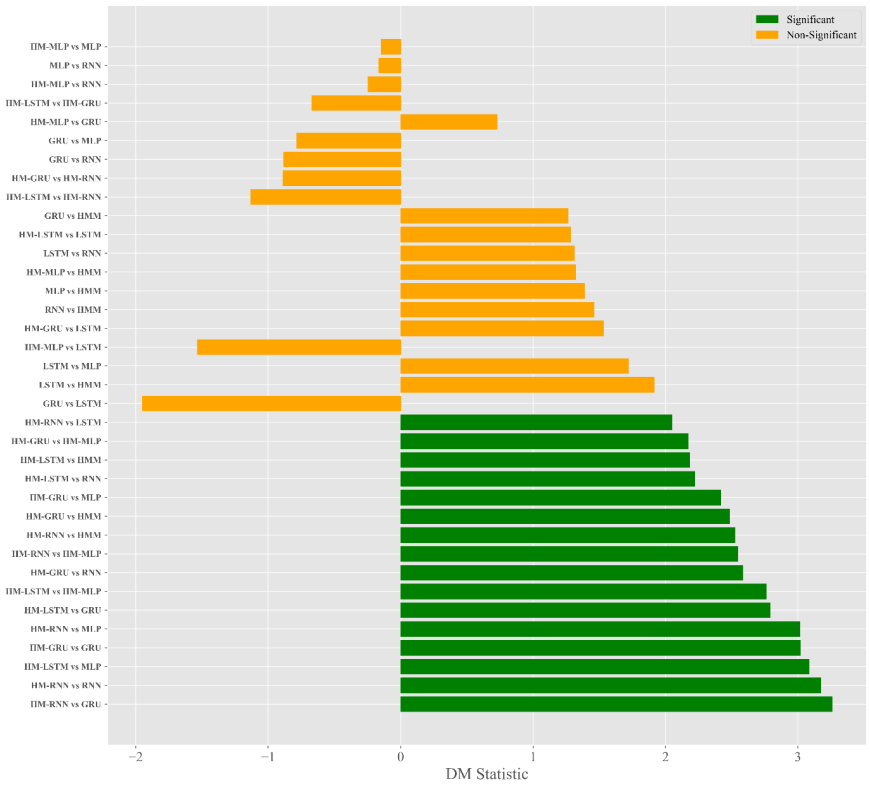

Figure 19: Various forecasting model comparison using DM test on Potato price series

inferred that the RMSEs of the HM-DL models *viz.*, HM-MLP, HM-RNN, HM-GRU, HM-LSTM were all significantly different (read lower) as compared to their DL counterparts for the Onion price series. For the Tomato series, HM-LSTM and for the Potato series, both HM-RNN and HM-GRU were found to be statistically significantly different.

HM based DL models have the advantage of adjusting to the pattern of the data within each of the hidden states found and hence the effect of non-stationarity of the data will be minimal. Meanwhile, HMM-DL models are able to handle data volatility, non-stationarity and non-normality better. However, in situations with datasets which are of relatively lesser size as compared to very large data sets, the application of HMM-DL models might underperform due to overfitting. In this study, careful hyperparameter tuning has been made to ensure model performance that avoided the overfitting issues.

To sum up, it can be concluded that Hidden Markov-Deep Learning (HMM-DL) approaches are more effective in forecasting TOP prices than traditional methods like HMM and baseline DL models. Thus, combining HMM with DL techniques seems to improve prediction accuracy even further, especially for long-term predictions like those required for agricultural commodities pricing.

This study demonstrates the effectiveness of HM-DL models for the accurate prediction of TOP vegetable prices. The proposed models provide farmers, traders, and Mandis with enhanced capabilities for reliable price forecasting and informed decision-making. Furthermore, the analysis enables farmers to optimize storage capacity by identifying periods of

low prices for storing vegetables and selling them during periods of higher prices, minimizing losses, as these can be readily seen by the stakeholders in the plots of forecasts. Overall, the hybrid HM-DL models offer a comprehensive understanding of market dynamics and provide valuable insights for optimizing decision-making in the agricultural sector.

5. Concluding remarks

This work proposed a novel Deep Learning (DL) approach based on hidden states to enhance the precision of TOP price forecasting. The hidden states identified by HMM serve as a feature extraction technique and were utilized in four DL models. The integration of HMM with DL and HMM models improved the forecasting accuracy compared to HMM and traditional DL models. The Diebold-Mariano (DM) tests, by and large, revealed that the RMSEs of the proposed HM-DL models were all significantly different (read lower) as compared to their DL counterparts for the onion price series. It is also emphasized here that the proposed HM-DL models took almost the same computational time while training them when compared to the baseline DL models. The findings demonstrate that the hybrid approach of Hidden Markov (HM) combined with DL models yields superior forecasting performance compared to existing models. Future research directions can extend the current study's univariate analysis of vegetable price series by incorporating multiple related variables, including weather conditions, market demand and economic indicators into the hybrid models. Moreover, other sectors such as finance, energy, and healthcare which involve complex time series data, can benefit from integrating HMM and DL models to improve forecasting accuracy and facilitate informed decision-making.

Acknowledgements

The facilities provided by ICAR-Indian Agricultural Statistics Research Institute (IASRI), New Delhi and the funding granted to the first author by Indian Council of Agricultural Research in the form of an IASRI-SRF fellowship are duly acknowledged for carrying out this study, which is a part of his doctoral research being pursued at ICAR-IASRI. In addition, thanks are due to the Graduate School, ICAR-IARI, New Delhi for their support provided. The authors also thank the Chair Editor and reviewer for helpful comments which led to considerable improvement in the paper.

References

Abdollahi, H. and Ebrahimi, S. B. (2020). A new hybrid model for forecasting brent crude oil price. *Energy*, **200**, 117520.

Adebiyi, A. A., Adewumi, A. O., and Ayo, C. K. (2014). Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, **2014**, 1–7.

Aizenberg, L., Sheremetov, L., Villa-Vargas, J., and Martinez-Munoz (2016). Multilayer neural network with multi-valued neurons in time series forecasting of oil production. *Neurocomputing*, **175**, 980–989.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Essen, B. C., Awwal, A. A. S., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, **8**, 292.

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

Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *16th International Conference on Computer Modelling and Simulation*, pages 106–112.

Athanasopoulos, G. and De Silva, A. (2012). Multivariate exponential smoothing for forecasting tourist arrivals. *Journal of Travel Research*, **51**, 640–652.

Avinash, G., Ramasubramanian, V., and Gopalakrishnan, B. N. (2022). Heterogeneous autoregressive modeling based realised volatility forecasting. *Statistics and Applications*, **21**, 121–140.

Awad, M., Khanna, R., Awad, M., and Khanna, R. (2015). Hidden markov model. In *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, pages 81–104.

Basak, S., Kar, S., Saha, S., Khaidem, L., and Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance*, **47**, 552–567.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, **3**, 1–8.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, **37**, 1554–1563.

Baum, L. E. and Sell, G. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, **27**, 211–227.

Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, **15**, 9–42.

Giudici, P. and Abu Hashish, I. (2020). A hidden markov model to detect regime changes in cryptoasset markets. *Quality and Reliability Engineering International*, **36**, 2057–2065.

Guresen, E., Kayakutlu, G., and Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, **38**, 10389–10397.

Hashish, I. A., Forni, F., Andreotti, G., Facchinetti, T., and Darjani, S. (2019). A hybrid model for bitcoin price prediction using hidden markov models and optimized LSTM networks. In *2019 24th International Conference on Emerging Technologies and Factory Automation*, pages 721–728.

Hassan, M. R. (2009). A combination of hidden markov model and fuzzy model for stock market forecasting. *Neurocomputing*, **72**, 3439–3446.

Haykin, S. (2009). *Neural Networks and Learning Machines*. Pearson Education India.

Heidarpanah, M., Hooshyaripor, F., and Fazeli, M. (2023). Daily electricity price forecasting using artificial intelligence models in the iranian electricity market. *Energy*, **263**, 126011.

Henrique, B. M., Sobreiro, V. A., and Kimura, H. (2018). Stock price prediction using support vector regression on daily and up-to-the-minute prices. *The Journal of Finance and Data Science*, **4**, 183–201.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**, 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**, 4661–4676.

Jarque, C. M. and Bera, A. K. (1987). A test for normality of observations and regression residuals. *International Statistical Review/Revue Internationale de Statistique*, **55**, 163–172.

Jarque, C. M. and Bera, A. K. (2011). Arima modeling with intervention to forecast and analyze chinese stock prices. *International Journal of Engineering Business Management*, **3**, 53–58.

Jiang, M., Liu, J., Zhang, L., and Liu, C. (2020). An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. *Physica A: Statistical Mechanics and its Applications*, **541**, 122272.

Khan, M. I., Acharya, B., and Chaurasiya, R. K. (2022). Hybrid biLSTM-HMM based event detection and classification system for food intake recognition. In *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies*, pages 1–5.

Kocak, C. (2017). Arma (p, q) type high order fuzzy time series forecast method based on fuzzy logic relations. *Applied Soft Computing*, **58**, 92–103.

Latif, N., Selvam, J. D., Kapse, M., Sharma, V., and Mahajan, V. (2023). Comparative performance of LSTM and ARIMA for the short-term prediction of bitcoin prices. *Australasian Accounting, Business and Finance Journal*, **17**, 256–276.

Lin, Z. (2018). Modelling and forecasting the stock market volatility of sse composite index using garch models. *Future Generation Computer Systems*, **79**, 960–972.

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

Manogna, R. and Mishra, A. K. (2021). Forecasting spot prices of agricultural commodities in india: Application of deep-learning models. *Intelligent Systems in Accounting, Finance and Management*, **28**, 72–83.

Mehdizadeh, S., Fathian, F., and Adamowski, J. F. (2019). Hybrid artificial intelligence-time series models for monthly streamflow modeling. *Applied Soft Computing*, **80**, 873–887.

Nelson, D. M., Pereira, A. C., and De Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *2017 International Joint Conference on Neural Networks*, pages 1419–1426.

Nti, K. O., Adekoya, A., and Weyori, B. (2019). Random forest-based feature selection of macroeconomic variables for stock market prediction. *American Journal of Applied Sciences*, **16**, 200–212.

Peng, Y., Feng, T., Yang, C., Leng, C., Jiao, L., Zhu, X., and Li, R. (2021). HMM-LSTM for proactive traffic prediction in 6g wireless networks. In *21st International Conference on Communication Technology (ICCT)*, pages 544–548.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, **52**, 591–611.

Singh, K. N., Sharma, K., Avinash, G., Kumar, R. R., Ray, M., Ramasubramanian, V., Ray, M., Lama, A., and Lal, S. B. (2023). LSTM based stacked autoencoder approach for time series forecasting. *Journal of the Indian Society of Agricultural Statistics*, **77**, 71–78.

Wang, C., Yang, H., Bartz, C., and Meinel, C. (2016). Image captioning with deep bidirectional LSTMs. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 988–997.

Wang, H., Jiang, Y., and Wang, H. (2009). Stock return prediction based on bagging-decision tree. In *International Conference on Grey Systems and Intelligent Services*, pages 1575–1580.

Wang, L., Feng, J., Sui, X., Chu, X., and Mu, W. (2020). Agricultural product price forecasting methods: research advances and trend. *British Food Journal*, **122**, 2121–2138.

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting*, **30**, 1030–1081.

Xiong, T. L. (2018). Seasonal forecasting of agricultural commodity price using a hybrid stl and elm method: Evidence from the vegetable market in china. *Neurocomputing*, **275**, 2831–2844.

Yao, Y. and Cao, Y. (2020). A neural network enhanced hidden markov model for tourism demand forecasting. *Applied Soft Computing*, **94**, 106465.

Yin, H., Jin, D., Gu, Y. H., Park, C. J., Han, S. K., and Yoo, S. J. (2020). STL-ATTLSTM: vegetable price forecasting using STL and attention mechanism-based LSTM. *Agriculture*, **10**, 612–619.

Yu, L., Zhao, Y., and Tang, L. (2017). Ensemble forecasting for complex time series using sparse representation and neural networks. *Journal of Forecasting*, **36**, 122–138.

Zaheer, S., Anjum, N., Hussain, S., Algarni, A. D., Iqbal, J., Bourouis, S., and Ullah, S. S. (2023). A multi-parameter forecasting for stock time series data using LSTM and deep learning model. *Mathematics*, **11**, 590.

APPENDIX

The following Python code implements the Hidden Markov Deep Learning (HM-DL) models for time series prediction.

```
#HM-DL models implementation by Python software

import pandas as pd
from sklearn.model_selection import train_test_split

# Load your dataset
my_data = pd.read_csv('path_to_your_data.csv')

# Convert 'Date' column to datetime if necessary
my_data['Date'] = pd.to_datetime(my_data['Date'])

# Ensure the data is sorted by date
my_data.sort_values('Date', inplace=True)

# Split data into train and test sets
train_size = int(len(my_data) * 0.9)
train_data, test_data = my_data[:train_size], my_data[train_size:]

from hmmlearn import hmm

# Initialize Gaussian HMM
# This assumes you've decided on the number of components based on
   your data (AIC/BIC)
model = hmm.GaussianHMM(n_components=number_of_states)

# Train HMM on the prices from the training data
model.fit(train_data[['Price']].values)

# Find the Viterbi path
hidden_states = model.predict(train_data[['Price']].values)

# Append Viterbi path to the training data
train_data['ViterbiPath'] = hidden_states

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, GRU, LSTM, Dense
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, GRU, LSTM, Dense
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_squared_error
```

```
# Define a function to create a deep learning model of a specified
   type
def create_dl_model(input_shape, num_hidden_layers, num_hidden_units
   , dl_type):
    model = Sequential()

    if dl_type == 'HM-MLP':
        for i in range(num_hidden_layers):
            model.add(Dense(num_hidden_units, activation='relu',
               input_shape=input_shape if i == 0 else (
               num_hidden_units,)))
    elif dl_type == 'HM-GRU':
        for i in range(num_hidden_layers):
            return_sequences = i < (num_hidden_layers - 1)
            model.add(GRU(num_hidden_units, return_sequences=
               return_sequences, input_shape=input_shape if i == 0
               else (num_hidden_units,)))
    elif dl_type == 'HM-LSTM':
        for i in range(num_hidden_layers):
            return_sequences = i < (num_hidden_layers - 1)
            model.add(LSTM(num_hidden_units, return_sequences=
               return_sequences, input_shape=input_shape if i == 0
               else (num_hidden_units,)))
    elif dl_type == 'HM-RNN':
        for i in range(num_hidden_layers):
            return_sequences = i < (num_hidden_layers - 1)
            model.add(SimpleRNN(num_hidden_units, return_sequences=
               return_sequences, input_shape=input_shape if i == 0
               else (num_hidden_units,)))
    else:
        raise ValueError("Unsupported␣deep␣learning␣type")

    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer=Adam())
    return model

# Grid search over hyperparameters
best_rmse = float('inf')
best_model = None
best_params = {}

batch_sizes = [8,16,32,64,128,256]
num_hidden_layers = [1, 2,3]
num_hidden_units = [8,16,32,64,128,256,512]
num_epochs = 200  # Use early stopping criteria if needed.

for dl_type in ['HM-MLP', 'HM-GRU', 'HM-LSTM', 'HM-RNN']:
    for batch_size in batch_sizes:
        for num_layers in num_hidden_layers:
```

```
            for num_units in num_hidden_units:
                # Create model
                dl_model = create_dl_model(input_shape=(lags, 2),
                   num_hidden_layers=num_layers, num_hidden_units=
                   num_units, dl_type=dl_type)

                # Convert the series to a supervised learning
                   problem, split into input and output
                # X_train, y_train = ...

                # Train model
                dl_model.fit(X_train, y_train, epochs=num_epochs,
                   batch_size=batch_size, verbose=0)

                # Predict on training set and calculate RMSE
                train_predictions = dl_model.predict(X_train)
                train_rmse = mean_squared_error(y_train,
                   train_predictions, squared=False)

                # Update best model if RMSE improves
                if train_rmse < best_rmse:
                    best_rmse = train_rmse
                    best_model = dl_model
                    best_params = {'batch_size': batch_size, '
                       num_layers': num_layers, 'num_units':
                       num_units, 'dl_type': dl_type}
```