PLANT SCIENCE TODAY
Vol x(x): xx–xx
https://doi.org/10.14719/pst.8643

eISSN 2348-1900

RESEARCH ARTICLE

Deep learning for cotton price prediction: Unveiling the impact of weather variables

Hari Priyaa A R A[1], Kalpana Muthuswamy[2]*, Venkatesa Palanichamy Narasimma Bharathi[3], M Vijayabhama[1] & R Parimalarangan[4]

[1]Department of Physical Sciences and Information Technology, Agricultural Engineering College and Research Institute, Tamil Nadu Agricultural University, Coimbatore 641 003, Tamil Nadu, India
[2]Office of the Dean (Agriculture), Tamil Nadu Agricultural University, Coimbatore 641 003, Tamil Nadu, India
[3]Agricultural College and Research Institute, Tamil Nadu Agricultural University, Coimbatore 641 003, Tamil Nadu, India
[4]Department of Agricultural Economics, CARDS, Tamil Nadu Agricultural University, Coimbatore 641 003, Tamil Nadu, India

*Correspondence email - kalpanam@tnau.ac.in

Received: 02 April 2025; Accepted: 01 June 2025; Available online: Version 1.0: 25 June 2025

Cite this article: Hari Priyaa ARA, Kalpana M, Venkatesa PNB, Vijayabhama M, Parimalarangan R. Deep learning for cotton price prediction: Unveiling the impact of weather variables. Plant Science Today (Early Access). https:/doi.org/10.14719/pst.8643

Abstract

Accurate forecasting of cotton prices is crucial for farmers and stakeholders in the agricultural sector to optimize crop selection, improve profitability and mitigate market risks. This study had developed a novel weather-based deep learning model to predict cotton prices in two major cotton-growing districts of Tamil Nadu, Perambalur and Salem. Despite the non-linear relationship between weather variables and price, advanced deep learning techniques were employed to uncover hidden patterns and enhance predictive accuracy. Since the price series was non-stationary, Seasonal-Trend-Residual decomposition using Loess decomposition was done to separate trend, seasonality and residual components and distinct models were fitted to each component. Four weather parameters-maximum temperature, minimum temperature, relative humidity and rainfall-were considered as exogenous variables. Feature selection was performed based on the mutual information score. Various deep learning architectures like STL-ANN, STL-TDNN, STL-GRU and STL-LSTM were explored to assess their effectiveness in forecasting prices for each decomposed component and finally ensembled together. The results demonstrated the potential of incorporating weather data into predictive models for cotton price forecasting, with the LSTM model outperforming other three models with MAPE of 3.68 % and 4.07 % in Salem and Perambalur districts respectively. The study highlights the potential of LSTM-based models in supporting informed decision-making and improved crop planning for cotton farmers in these regions.

Keywords: artificial neural networks (ANN); cotton price forecast; ensemble; gated recurrent units (GRU); long-short term memory (LSTM); STL decomposition; time delay neural networks (TDNN)

Introduction

Cotton, often referred to as the "King of Fibres" or "white gold," is a vital cash crop with global economic significance and serves as the primary raw material for the textile industry (1). According to Ministry of Textiles, Government of India, 2022, India stands first in the world in cotton acreage, with 124.69 lakh hectares under cotton cultivation which is approximately 39 % of the world area of 318.8 lakh hectares. In India, among commercial crops, cotton has crucial importance as it accounts for 23 % of global cotton production. The major cotton producing states in India are Gujarat, Maharashtra and Telangana accounting for 65 % of India's total production of cotton. Tamil Nadu, although not the top producer, significantly contributes to the textile industry by processing cotton into high-quality yarn and fabrics (2).

Agricultural commodity price fluctuations have an impact on a nation's Gross Domestic Product (GDP). Farmers suffer significant financial losses when prices drop after the crop is harvested. Price volatility in cotton can lead to financial uncertainty, affecting farmers' income and market stability. Reliable price predictions help stakeholders make informed decisions regarding sowing, storage and sales, ultimately improving profitability and reducing market risks. Weather is a key determinant of agricultural productivity, directly influencing crop yield, quality and price. Incorporating weather variables such as temperature, rainfall and humidity into price prediction models enhanced their accuracy by accounting for climate-driven fluctuations in supply. Weather-based forecasting could help in early identification of potential price surges or declines, allowing farmers and market participants to plan accordingly (3).

Unlike traditional models that rely solely on historical prices, weather-integrated models offer a dynamic and responsive approach aligned with real-time climatic variations. Deep learning has revolutionized predictive analytics by leveraging large datasets and extracting complex patterns that traditional models fail to capture (4). Recent research has applied deep learning techniques to agricultural price forecasting, aiming to support decision-making through enhanced pattern recognition and predictive performance. Unlike statistical and conventional machine learning models, deep learning automatically extracts relevant features, reducing manual effort and improving prediction accuracy. The growing availability of high-resolution weather and market data further strengthens the potential of deep learning in developing robust, data-driven forecasting models for agricultural commodities like cotton.

This study aims to develop a novel deep learning-based cotton price forecasting model by integrating weather parameters as key predictors. The model will be tailored for two major cotton-growing districts of Tamil Nadu, Perambalur and Salem, to enhance the accuracy of price predictions. By leveraging deep learning techniques like Artificial Neural Networks (ANN), Time Delay Neural Networks (TDNN), Gated Recurrent Units (GRU) and Long-Short Term Memory (LSTM), this research seeks to provide farmers with data-driven insights to optimize crop selection, improve market decision-making and ultimately enhance profitability.

Background

Accurate price forecasting of agricultural commodities is crucial for farmers and other industry stakeholders to make rational choices, ensure food security and increase farming profitability. Because of the growing amount of historical data on agricultural commodity prices and the necessity to accurately predict price movements, machine learning and deep learning have essentially replaced statistical methods (5).

Selecting an appropriate subset from historical data for forecasting remains a significant challenge. Implementing machine learning techniques in practice, often faces issues such as high dimensionality, nonlinearity and the difficulty of choosing model with optimal parameters. In a comparative study involving ARIMA, SVR, Prophet, XGBoost and LSTM on large historical datasets, the LSTM model outperformed the others in predicting prices of chicken, chilli and tomato (6).

Weather significantly impacts agricultural production, influencing crop yield, quality and market prices. Traditional price prediction models primarily relied on historical price trends, often neglecting crucial external factors like temperature, rainfall and humidity. However, integrating weather variables into forecasting models has been shown to enhance their predictive accuracy by accounting for climate-driven fluctuations in supply and demand (7).

Deep learning has emerged as a powerful tool in agricultural forecasting due to its ability to capture complex, non-linear relationships in large datasets. Unlike traditional statistical and machine learning models, deep learning methods would automatically extract relevant features, reducing manual preprocessing efforts and improving prediction accuracy. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) had been widely used in crop yield estimation, while advanced architectures like Long Short-Term Memory (LSTM) and Transformer-based models demonstrated superior performance in time series forecasting (8).

Deep learning models-NBEATSX and TransformerX were used for predicting agricultural commodity prices in three vegetable crops Tomato, Onion and Potato by incorporating weather data as exogenous factors. When compared with statistical (ARIMAX, MLR) and machine learning approaches (ANN, SVR, RFR, XGBoost), deep learning models achieved lower error metrics. To forecast realized price volatility in time series data, Heterogeneous Autoregression (HAR) model could be used with historical volatility data to predict future volatility levels (9).

TDNN has emerged as an effective model for time series forecasting due to their ability to capture temporal dependencies in sequential data. The TDNN model with error corrected term obtained by cointegration performed well for forecasting the monthly wholesale price index of fruits (10). Hidden Markov-based GRU model was found to be effective for short and medium-term forecasts (eight and twelve week ahead) of Potato prices with lower values of RMSE, MAE and MAPE than other considered deep learning models (11).

Artificial Neural Network was used for day-ahead electricity price forecasting using a 119-day training set of 119 days with no preprocessing to examine the realistic paradigm of ANN based neural network models. Increasing the number of epochs found to increase the quality of results but with the demerit of higher training period. A reasonable selection of model was achieved after 500 training epochs, but in many cases not maximum number of epochs were reached to get more accuracy (12).

For non-stationary and non-linear characteristics, Chinese hog price was decomposed using the Seasonal and Trend decomposition using the Loess (STL) model and trained using LSTM and SARIMA models. After STL decomposition, Factor analysis for multivariate influence factors was done and GRU model was trained, which produced more accurate forecast of hog prices (13).

Building on insights from prior studies, this paper proposes a deep learning-based model for forecasting the weekly price of cotton in the Perambalur and Salem districts of Tamil Nadu. The model incorporates four key weather parameters-minimum temperature (Tmin), maximum temperature (Tmax), relative humidity (RH) and rainfall (RF)-as exogenous variables to improve predictive accuracy.

Methodology

Data collection

The data used in this study included the weekly market price data of cotton collected from reference markets and AGMARKNET website and the weather data in the corresponding region. It included Tmin, Tmax, RH, RF data obtained from the web source NASA Power data. Area of data collection comprises two districts - Salem in North Western Zone and Perambalur in Western Zone of Tamil Nadu from January 2010 to December 2024 (15 years). These districts were selected due to their significant contribution to cotton production in Tamil Nadu during the year 2022 to 2023 with production of 28509 bales in Perambalur and 20641 bales in Salem. The weekly price of Cotton in Perambalur and Salem districts is illustrated in Fig. 1.

Removal of outliers

In agriculture and commodity pricing, sudden price spikes may indicate supply chain issues, seasonal effects, or external shocks which are termed as outliers. Outliers in price data can distort analysis and affect forecasting models and may skew the mean and affect statistical measures (for e.g., standard deviation). Therefore, outlier detection is crucial for price forecasting, economic modelling and decision-making. As the data is skewed strongly, Inter-Quartile Range (IQR) method is used to clean the price data. The distribution of data after cleaning of outliers is given in Fig. 2.

Empirical tests on data, data preprocessing and data splitting

Stationarity tests and non-linearity tests : A stationary series maintains a constant mean and variance over time, and the covariance depends on the lag distance. To accurately fit the models, it is important to check the stationarity and linearity of both series. The Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test are commonly used for this purpose, but they differ in their hypotheses: the ADF test has a null hypothesis of non-stationarity (presence of a unit root), while the KPSS test assumes stationarity as the null hypothesis. Using both tests in conjunction provides a more comprehensive view of the data's stationarity characteristics (14, 15). In addition to stationarity, checking for non-linearity is essential, as many real-world time series contain complex, non-linear relationships, structural breaks, or chaotic patterns that linear models may fail to capture. The Brock-Dechert-Scheinkman (BDS) test is a popular method for detecting non-linearity. It tests whether a time series is independently and identically distributed (i.i.d.) and helps determine if a linear model is sufficient or if a non-linear model might be more appropriate (16).

Data preprocessing: STL decomposition is particularly useful for agricultural commodity price analysis, breaking down a time series into three key components - trend, seasonality and residual. It identifies the underlying patterns influenced by harvest seasons, market cycles and policy changes (17). Trend Component shows long-term movement or direction of the price series. Seasonality Component captures the repeating patterns or cycles at fixed intervals and the Residual Component represents noise, random fluctuations, or irregular variations. It uses Locally Estimated Scatterplot Smoothing (LOESS) function to adaptively capture complex patterns in the data. STL decomposition for price data with changing seasonal patterns, makes it ideal for price fluctuations influenced by market conditions, weather, or policy shifts (18). The STL decomposition of the price series was performed using the STL function from the stats models library in Python, with the period parameter set to 26 (Fig. 3 and 4). By decomposing the data, the trend and seasonality component can be separately fed for model training and hyperparameter tuning for better forecast.

Both Perambalur and Salem price series were identified as trend-stationary, indicating that the fluctuations occur around a deterministic trend and do not contain a unit root.

Data splitting: The dataset was divided into three subsets: training data, validation data and test data, to ensure effective model training, hyperparameter tuning and performance evaluation. The general practice is to allocate more data for model building and selection. The length of entire data set is 783 data points for which training, validation and testing dataset are split in the ratio 80-10-10 as shown in Table 1. The training set is used to enable the model to learn underlying patterns, trends and dependencies (19). The validation set helps assess the model's generalization ability and prevents overfitting by optimizing key parameters and used for hyperparameter tuning and model selection. Test dataset serves as an independent dataset to measure the model's predictive accuracy on unseen data, ensuring its reliability in real-world forecasting

Table 1. Data split in- and out-of-sample for model fit

Training data	Validation data	Test data
2010-01-01 to 2022-01-02	2022-01-09 to 2023-07-02	2023-07-09 to 2024-12-27
627 datapoints	78 datapoints	78 datapoints

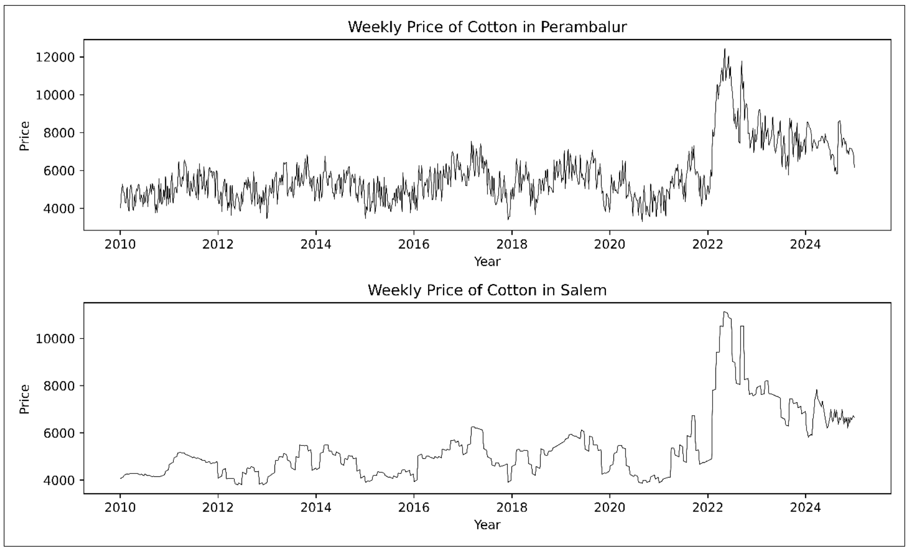

Fig. 1. Weekly market price of cotton in Perambalur and Salem districts.

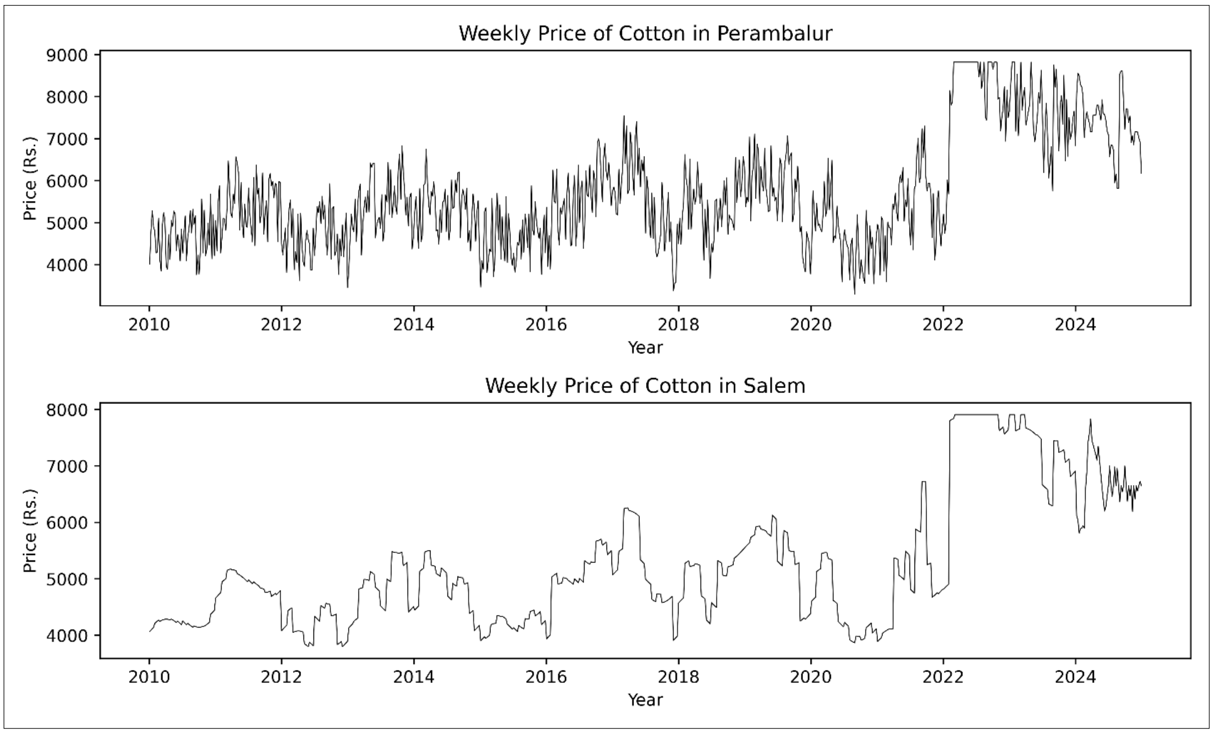

Fig. 2. Weekly price of cotton after removal of outliers.

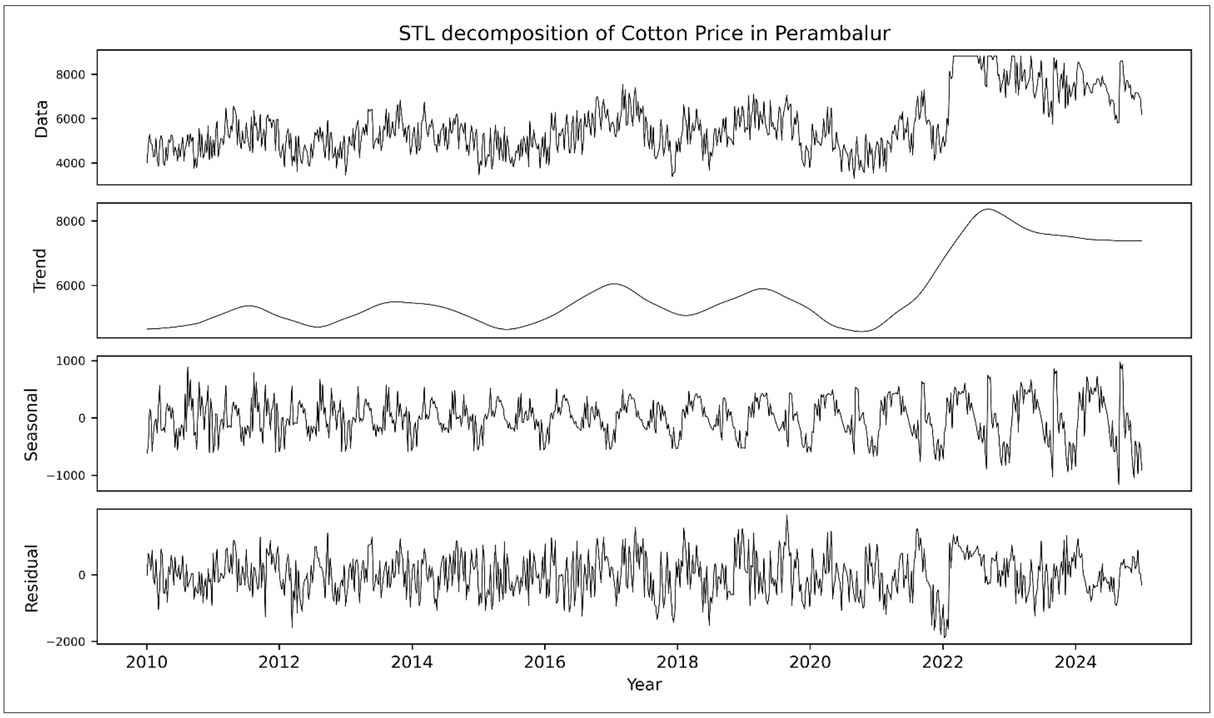

Fig. 3. STL decomposed cotton price data of Perambalur.

Feature selection

Mutual information is a fundamental concept from information theory that measures the mutual dependence between two variables. Unlike correlation, which captures only linear relationships, MI detects both linear and nonlinear dependencies between variables (20).

Mathematically, MI is defined in Equation 1:

$$(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

(Eqn. 1)

where *p(x,y)* is the joint probability distribution of X and Y;

p(x) p(y) are the marginal probability distributions of X and Y.

Using an appropriate bin size improves the accuracy of MI estimation, making it a valuable tool for feature selection and dependency analysis in machine learning and statistics. The Freedman-Diaconis Rule which helps determine the optimal number of bins for MI computation to balance bias and variance (21) is given in the Equation 2:

$$h = 2 \times IQR(X) \times n^{-1/3} \qquad \text{(Eqn. 2)}$$

where, *h* is the optimal bin width;

IQR(X) is the interquartile range of variable X;

n is the number of observations.

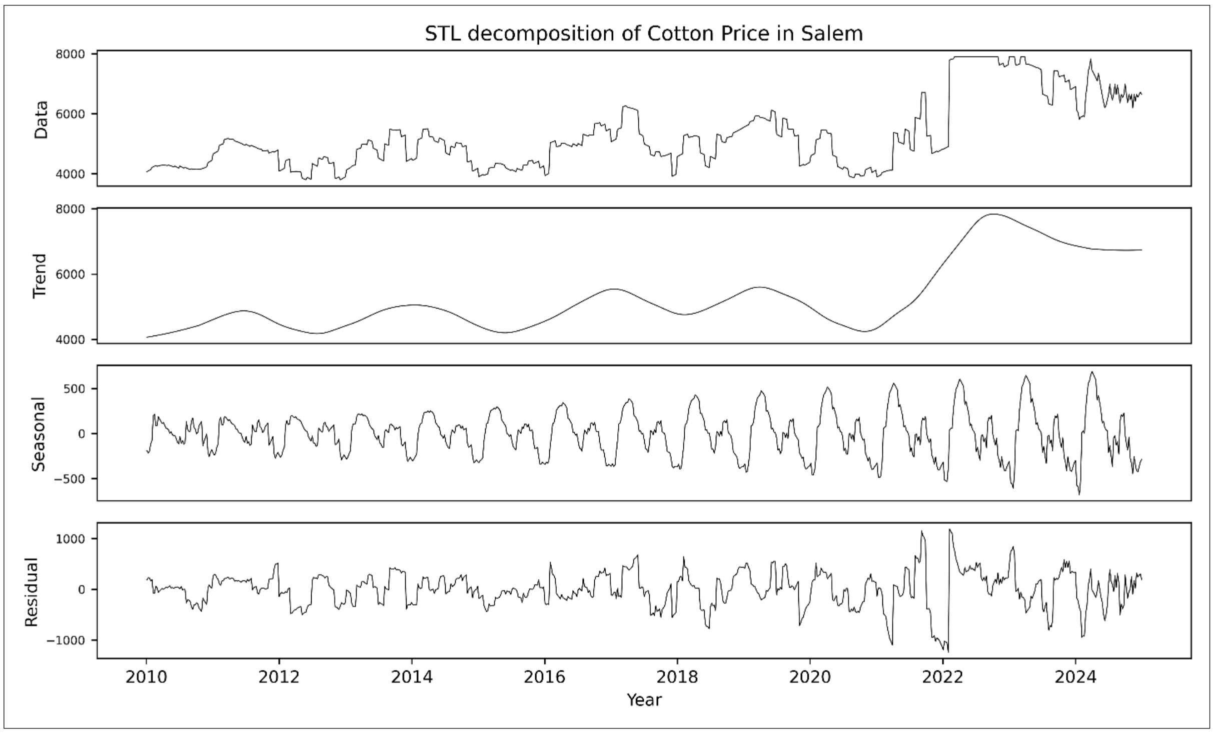

Fig. 4. STL decomposed cotton price data of Salem.

Then the number of bins is then calculated as given in Equation 3:

$$\text{Number of bins} = \frac{\max(x) - \min(x)}{h} \qquad \text{(Eqn. 3)}$$

This rule adapts to the distribution of the data, ensuring that the bin width is neither too large thus, losing detail nor too small leading to more noise.

Deep learning and model architecture description

Deep learning

Deep learning had gained significant attention consisting of CNNs, RNNs autoencoder neural networks and deep belief networks. Besides commonly used ANN and TDNN, RNNs such as LSTM and GRU are used. As RNNs are neural networks with recurrent loops, they are considered to be most appropriate for time series data. RNNs have memory that holds the knowledge pertaining to the observed data, which is why they are termed as recurrent networks. Each layer's output is dependent on the computations of its preceding levels (22).

The input features for the neural network include four weather parameters-maximum temperature (Tmax), minimum temperature (Tmin), rainfall (RF) and relative humidity (RH)-along with the target variable, cotton price. The training process for a deep learning model typically involves four key stages:

1) **Data preparation:** The effectiveness of model training improves with larger datasets. This stage focuses on refining the data by removing incomplete records and unwanted variations while also incorporating data augmentation or simulation to enhance its quality.
2) **Choosing an optimal network architecture:** The appropriate number of neurons, layers and network type is determined through experimentation or trial-and-error to achieve the best performance.
3) **Training the network:** During this stage, the model is trained over several epochs using optimization algorithms such as stochastic gradient descent or Adam. Training continues until convergence criteria are met or until the model achieves acceptable performance on the validation dataset.
4) **Enhancing training efficiency:** To improve training stability and prevent overfitting, advanced techniques such as batch normalization, dropout regularization and transfer learning are utilized. These methods enhance convergence speed, improve generalization to unseen data and increase model robustness (22).

The hidden layer in an RNN changes according to the Equation 4:

$$a_h^t = \sum_{i=1}^{I} \omega_{ih} X_i^t + \sum_{h'=1}^{H} \omega_{h'h} b_{h'}^{t-1} \qquad \text{(Eqn. 4)}$$

where, ω_{ih} is the weight between the ith input and hth neuron from the hidden layer;

X_i^t is the ith input of network in moment *t*;

$\omega_{h'h}$ is the weight between neurons of the hidden layer.

The Back Propagation (BP) training changes according to the Equation 5:

$$\delta_h^t = \theta'\,(a_h^t)\Big(\sum_{k=1}^{K} \delta_k^t\,\omega_{hk} + \sum_{h'=1}^{H} \delta_{h'}^{t+1}\,\omega_{hh'}\ \Big) \qquad \text{(Eqn. 5)}$$

Activation function: The activation function introduces non-linearity into neural networks, allowing them to model complex patterns. ReLU and Leaky ReLU are commonly used in regression models due to their ability to handle non-linear relationships. The Sigmoid function and tanh are often used in binary classification. Rectified Linear Unit (ReLU) is widely used due to its simplicity and effectiveness in deep networks, but it suffers from the dying ReLU problem (23). Leaky ReLU addresses this issue by allowing small negative gradients (24). The choice of activation function impacts training efficiency and model performance.

Weight initialization: Proper weight initialization is crucial for stable and efficient training. The commonly used initialization techniques are Random Initialization, Xavier/Glorot Initialization and He Initialization. Xavier initialization is suitable for activation functions like Sigmoid and Tanh, as it ensures that activations remain in a reasonable range. He initialization is ideal for ReLU-based networks as it scales weights according to the number of incoming neurons, preventing vanishing or exploding gradients. Poor weight initialization can lead to slow convergence or divergence during training (25).

Regularization: Regularization is a technique used to reduce overfitting by incorporating a penalty term into the loss function. The two most common types are L1 (Lasso) and L2 (Ridge) regularization. By setting some weights to zero, L1 regularization promotes sparsity and improves the model's interpretability. By punishing big weights' squared values, L2 regularization (also known as weight decay) discourages them and produces a more generic model (26). Combining L1 and L2 regularization results in ElasticNet. By preventing the model from becoming over complicated and memorizing the training data, these strategies improve generalization to new data (27).

Dropout is another regularization technique used to reduce overfitting in neural networks. It works by randomly deactivating a fraction of neurons during each training iteration, forcing the network to learn redundant representations. The dropout rate controls how many neurons are dropped per layer. A higher dropout rate may lead to underfitting, while a very low dropout rate might not sufficiently prevent overfitting. Dropout is often applied in fully connected layers and sometimes in recurrent layers in models like LSTMs and GRUs (28).

ANN

McCulloch and Pitts in 1943 introduced the idea of ANN, which were then widely adopted by researchers for experimental modelling of nonlinear phenomena. McClelland, Rumelhart and Hinton in 1986 successfully developed a multilayer feedforward neural network (MLP) by introducing the Back Propagation algorithm. Neural networks have several uses in the financial and investing domains, including financial planning, decision-making and bankruptcy prediction (22).

Units, or artificial neurons, are the building blocks of ANN. These include an input layer that receives external data, one or more hidden layers that transform inputs through weighted connections and activation functions and an output layer that produces the final predictions (29). The connections between neurons are represented by weighted links, where each weight reflects the strength of influence from one unit to the next. To enhance model performance, these weights are changed throughout training. The weights assigned to each of these relationships specify how one unit affects another. The neural network learns more about the data as it moves from one unit to another and the output layer eventually receives the results (30).

A fundamental process in an ANN is determining the weighted sum of inputs (z_i) and then applying an activation function (a_i). The commonly used activation functions are the sigmoid function, hyperbolic tangent (tanh) and rectified linear unit (ReLU), as well as Leaky ReLU. These functions add non-linearities to the input, helping ANNs to grasp deep relationships and improve their ability to learn complex patterns (31). The optimization approach uses the gradient descent technique to quantify prediction errors. These neural networks have the capacity to simulate intricate nonlinear relationships without making presumptions about their nature that makes them a valuable tool in regression analysis.

Hyperparameters such as activation functions, regularization methods, dropout rates and initialization techniques were optimized using search algorithms. The activation function was chosen from ReLU or, Leaky ReLU. The weight initialization was selected from HeNormal or Random, ensuring stable gradient propagation. Regularization methods were explored in the range of L1 (0.0001–0.01). Dropout rates between 0.1 and 0.5 were tested to find the best balance between underfitting and overfitting. These selections ensured a well-regularized, stable and efficient ANN model, for both price series and used accordingly.

The output layer with single a node was used for regression to represent the predicted continuous value. The output $\hat{y}_i$ of the ith node in the network was computed as in Equation 6:

$$\hat{Y}_i = a_i(z_i) \quad \text{(Eqn. 6)}$$

$a_i(.)$ is the activation function applied at the ith node;

z_i is the weighted input to the node;

$\hat{y}_i$ output of the ith node in the network.

TDNN

Conventional ANNs do not inherently capture time dependencies, making them less effective for sequential or time series data unless specifically adapted. Neural networks can be used in two ways to model time series. A recurrent neural network is used in the first and a short-term memory is established at the input layer of the network in the second. One example of the latter is a TDNN, which uses the time delays to capture the temporal dimension of the series and build a short-term memory (also called hetero-associative memory) in its network (32). The past data are mapped to the future value via a nonlinear functional mapping as in Equation 7:

$$X_t = f(X_{t-1}, X_{t-2}, \ldots, X_{t-p}, w) + e_t \quad \text{(Eqn. 7)}$$

where f is a function defined by the network structure and connection weights and w is a vector containing all parameters (33).

TDNN is similar to FFNN, except that the input weight includes a delay element.

If $x = [x(t) \cdot x(t-1), \ldots, x(t-q)]$ is the input signal, the net input signal can be expressed as in Equation 8:

$$net_j(t) = \sum_{i=0}^{p}(w_{ji}y(t-i) + b_j) \quad \text{(Eqn. 8)}$$

$net_j(t)$= Net input signal to neuron *j* at time *t*;

w_{ji}= Weight associated with the connection from input *y(t-i) to* neuron *j*;

y(t - i)= Input signal at time *t - i* (delayed input);

p = the Number of past time steps considered (time delay);

b_j= Bias term for neuron j.

The final output value y_{t+1} in TDNN model is shown in the following Equation 9:

$$y_{t+1} = g[\sum_{j=0}^{q} \propto_j f(\sum_{i=0}^{p} \beta_{ij} y_{t-i})] \quad \text{(Eqn. 9)}$$

where, *f* and *g* are the activation functions at the hidden and output layers;

p is number of input nodes;

q is number of hidden nodes;

β_{ij} is weight attached to the connection between ith input node and jth node of hidden layer;

α_j is weight attached to the connection from the jth hidden node to the output node;

Y_{t-i} is the ith lag input of the model (34).

TDNN require careful selection of hyperparameters to optimize temporal feature extraction. The tuning of these hyperparameters was done in a problem-specific way and decided by grid search technique on the available data. The context window size was searched in the range of {3, 5, 7, 9}, determining how many time steps contribute to each prediction. Activation functions like ReLU and Leaky ReLU were tested. Weight initialization was selected from Glorot, HeNormal and Random, ensuring stable training. Regularization techniques such as L2 regularization (λ = 0.0001-0.1) and dropout rates (0.2-0.5) were used to help prevent overfitting. Besides these, number of epochs (50-100) and batch size (16-128) were also taken for grid search. Optimizing these parameters would ensure that TDNN captured long-range dependencies effectively while maintaining generalization. Only one output node was used since recursive forecasting method was adopted (17).

LSTM

LSTM is a type of RNN model designed for sequential data, introduced and improved by Gers, Schmidhuber and Cummins in 2000. Long-term memory refers to learned weights and short-term memory refers to internal states of cells (22). Neural Networks often notice difficulties in learning long-term dependencies, when information from distant time steps is critical for making correct predictions about current state. This issue is referred to as the vanishing or exploding gradient problem. When training a model over time, the gradients can diminish as they progress through several steps. This makes it difficult for the model to understand long -term patterns because previous data becomes almost useless. Sometimes, gradients can become too big and cause instability. This makes it difficult for the model to learn correctly because the updates are irregular and unexpected. Both of these limitations make it difficult for ordinary RNNs to accurately capture long-term dependencies in sequential data (35).

LSTM is capable of capturing long-term dependencies. Its architecture consists of recurrent subnetworks known as memory blocks. Each block contains one or more autoregressive memory cells along with three key components: input, output and forget gates, which regulate information flow. The input gate is responsible for incorporating inputs such as Tmax, Tmin, RH, RF and Price into the cell. The output gate o_t specifies the output of the cell and the forget gate f_t is responsible for indicating any prior values that might be needed in the future and retains them (36). The calculation of f_t, i_t, o_t and h_t are carried out as in Equations (10-13).

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f) \quad \text{(Eqn. 10)}$$

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \quad \text{(Eqn. 11)}$$

$$O_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \quad \text{(Eqn. 12)}$$

$$h_t = O_t \tanh(C_t) \quad \text{(Eqn. 13)}$$

where, W_i, W_f, W_o, are the weight matrices;

b_i, b_f and b_o are the bias vectors;

h_t is value of the memory cell at time *t*;

f_t is the value of the forget gate layer;

C_t is the current cell state;

i_t is the value of the input gate; and

o_t is the output gate layer.

For LSTM model, hyperparameters were optimized to enhance sequential learning. The number of LSTM units was searched in {32, 64, 128, 256 }, balancing model complexity and computational efficiency. Recurrent dropout rates between 0.1 and 0.5 were tested to regularize recurrent connections. Weight initialization was optimized using HeNormal, ensuring stability. L2 regularization values in the range 0.0001-0.1 would help prevent overfitting. Gradient clipping in the range 0.1-1.0 was also tuned to control exploding gradients, ensuring stable training.

GRU

GRU Neural network is one of the most successful variants of LSTM network (37). However, LSTMs are extremely complicated structures with huge computational costs. To address this, GRU was invented, which exploits the LSTM architecture by combining its gating techniques to provide a more efficient solution for numerous sequential tasks without giving up performance (38).

The fundamental principle behind GRUs is to use gating mechanisms which selectively updates the hidden state at each time step, which allows them to retain long-term and crucial information while discarding irrelevant details, enabling more accurate predictions. GRUs seek to simplify the LSTM design by combining some of its components and focusing on only two major gates-the update gate and the reset gate-resulting in fewer parameters and reduced computational complexity. They exhibit more efficient nonlinear fitting ability and are better to handle smaller datasets in comparison with LSTM models. At each timestamp t, it receives an input X_t which contains Tmax, Tmin, RF, RH, Price all together and the hidden state H_{t-1} is from the previous timestamp $t-1$. Later, it returns a new hidden state H_t, which is then sent to the following timestamp. The Reset Gate controls the network's short-term memory, i.e. the hidden state (H_t) and the Update Gate controls long-term memory (39).

The process can be described in the equations (14-17) as follows:

$$Z_t = \sigma(x_t w^z + h_{t-1}U^Z + b_z) \quad \text{(Eqn. 14)}$$

$$r_t = \sigma(x_t w^r + h_{t-1}U^r + b_r) \quad \text{(Eqn. 15)}$$

$$\tilde{h}_t = \tanh(r_t \,.\, h_{t-1}U + x_t W + b) \quad \text{(Eqn. 16)}$$

$$h_t = (1 - Z_t)\,.\,\tilde{h}_t + Z_t \,.\, h_{t-1} \quad \text{(Eqn. 17)}$$

Where, W^z, W^r and W are weight matrices of corresponding connected input vectors.

U^z, U^r and U are weight matrices of the previous time step;

b_z, b_r and b are the biases;

σ denotes sigmoid function;

r_t, Z_t and $\tilde{h}_t$ denotes reset gate, update gate and the candidate hidden layer.

For Gated Recurrent Units (GRU), hyperparameter tuning was focused on optimizing memory efficiency and learning speed. The number of GRU units was selected from {32, 64, 128, 256}, balancing the model performance and training time. Weight initialization was explored between Glorot and HeNormal Initialization. Dropout and recurrent dropout rates were optimized in the range 0.1–0.5. L2 regularization between 0.0001 and 0.1 was applied to improve generalization. Bidirectionality was also considered, selecting between Unidirectional and bidirectional GRU to assess its impact on performance. The number of dense layers, epochs and batch size were also considered for grid search with mean absolute error as loss function for model compilation with each component after STL decomposition of both price series.

Model evaluation

To quantitatively assess the performance of each forecasting model across different time horizons, three widely used accuracy metrics were employed: Root Mean Square Error (RMSE), Mean Absolute Prediction Error (MAPE) and Mean Absolute Error (MAE). RMSE is the standard deviation of the residuals of the model. MAE is the average difference of residuals of the model and MAPE is the percentage of average absolute error which give a way to compare the performance of the different models (36). The formulae for RMSE, MAE AND MAPE are given in Equations (18 - 20)

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2} \quad \text{(Eqn. 18)}$$

$$MAE = \frac{1}{n}\sum_{T=1}^{N}|Y_t - \hat{Y}_t| \quad \text{(Eqn. 19)}$$

$$MAPE = \left[\frac{1}{n}\left(\sum_{t=1}^{n}\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right|\right)\right] * 100 \quad \text{(Eqn. 20)}$$

Where, Y_t and $\hat{Y}_t$ are the actual and predicted values of price.

Furthermore, to determine the best-performing model, a post hoc analysis was conducted using the Diebold-Mariano test, enabling an effective comparison of forecast accuracies (11). The null hypothesis of Diebold-Mariano test is that the models have equal predictive accuracy and the alternative hypothesis is that one model is significantly better than the other. The Diebold-Mariano test statistic is given in Equation 21:

$$DM = \frac{\bar{d}}{\sqrt{\frac{2\pi \hat{f}_d(0)}{T}}} \quad \text{(Eqn. 21)}$$

Where, $\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t$ is the mean difference of the loss function between two models;

d_t is the difference between the loss functions of the two models at time t;

$\hat{f}_d(0)$ is an estimate of the spectral density of d_t at frequency zero, often estimated using the Newey-West estimator to account for autocorrelation;

T is the total number of observations.

Results and Discussion

Descriptive statistics

Table 2 presents the descriptive statistics of the weekly cotton price series for Salem and Perambalur districts used in this study.

Perambalur price series exhibits a higher mean value than Salem, indicating that the central tendency of the observed values is greater in Perambalur. The standard deviation for Perambalur is slightly higher than that of Salem, suggesting that Perambalur has a marginally greater spread

Table 2. Descriptive statistic measures of the weekly market prices of cotton in Perambalur and Salem districts

District	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Perambalur	3298	8824	5700.05	1308.83	0.7683	-0.0586
Salem	3799	7909.5	5267.66	1151.08	1.0020	0.0073

in its data distribution. The skewness values for Perambalur and Salem indicate a right-skewed distribution, meaning that both districts experience occasional price spikes. However, Salem exhibits a higher degree of positive skewness, suggesting that extreme price increases are more frequent in Salem than in Perambalur, which can potentially influence their crop planning and market strategies based on the anticipation of occasional sharp price spikes. The kurtosis values for Perambalur and Salem are close to zero, indicating a mesokurtic distribution. This suggests that price distributions in both districts are relatively stable and resemble a normal distribution, with moderate occurrences of extreme price variations. This insight can help farmers and market analysts anticipate price trends and manage risk accordingly.

Tests for stationarity

The stationarity of the time series data for Perambalur and Salem was assessed using the Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests. For the ADF test, the null hypothesis (H_0) states that the series is non-stationary. The test statistic for Perambalur was -3.8000, with a p-value of 0.0166, leading to the rejection of the null hypothesis, indicating that the series is stationary. Similarly, for Salem, the test statistic was -3.5265, with a p-value of 0.0366, also supporting stationarity. The KPSS test, which has a null hypothesis that the series is stationary, yielded test statistics of 1.8830 and 2.0227 for Perambalur and Salem, respectively, both with p-values of 0.0100. While these values suggest non-stationarity based on conventional interpretation, further stationarity analysis of the period from 2010 to 2022 indicated that the series exhibits stationarity within this timeframe. Therefore, STL decomposition was performed to analyse the trend, seasonality and residual components of the data in greater detail.

The results of BDS test presented in Table 3 reveal strong rejection of linearity for both price series.

The rejection of the null hypothesis suggests that the data exhibits nonlinear dependence. This means the series is not purely random; there is some underlying pattern or structure. Thus, nonlinear models are suggested for cotton price

Table 3. Results of BDS test for linearity

District	BDS Test Statistic	P value	Conclusion
Perambalur	14.3047	<0.00	Non-linear dependence
Salem	39.0563	0.00	Non-linear dependence

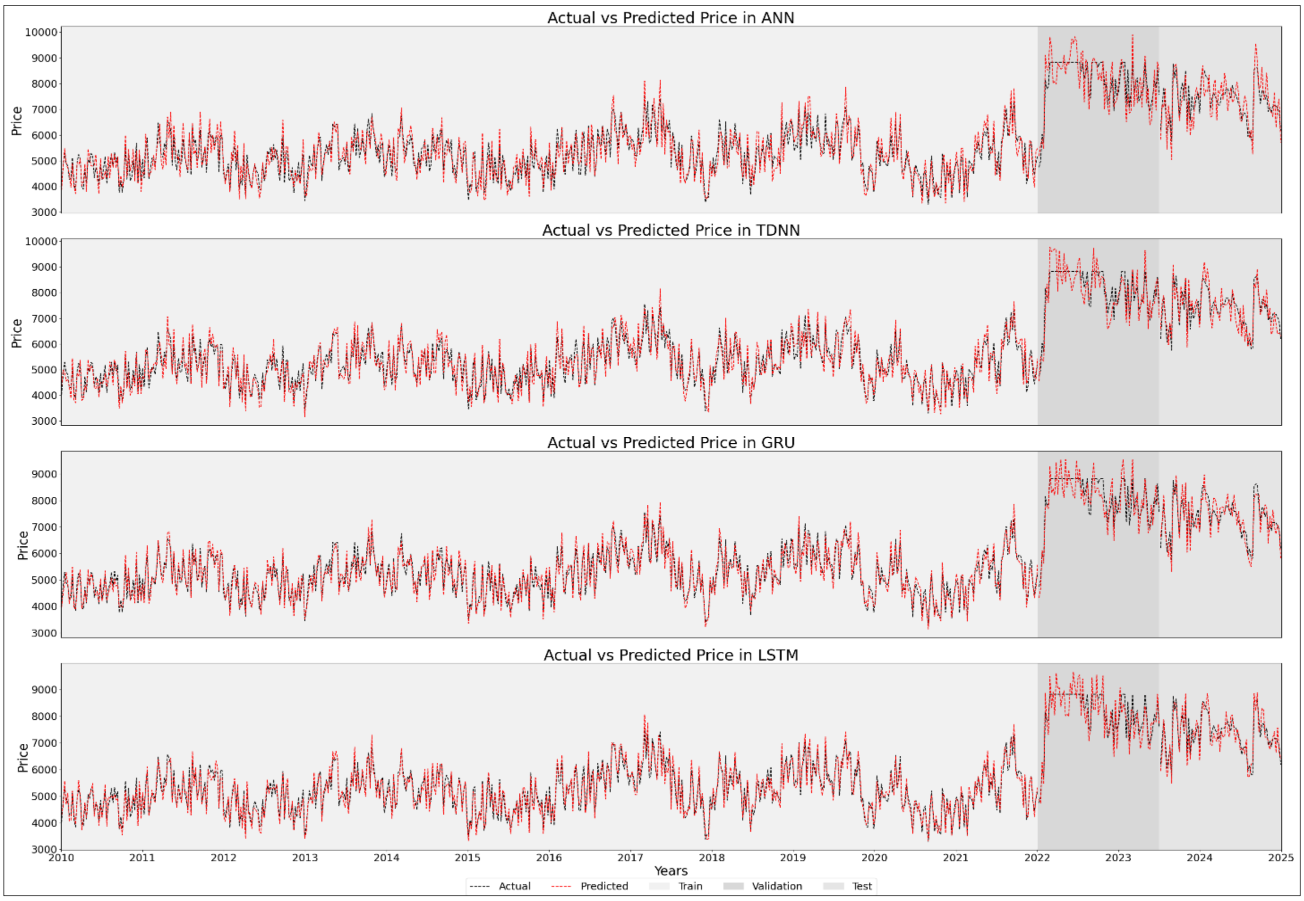

Fig. 5. Cotton price prediction of different models for Perambalur district.

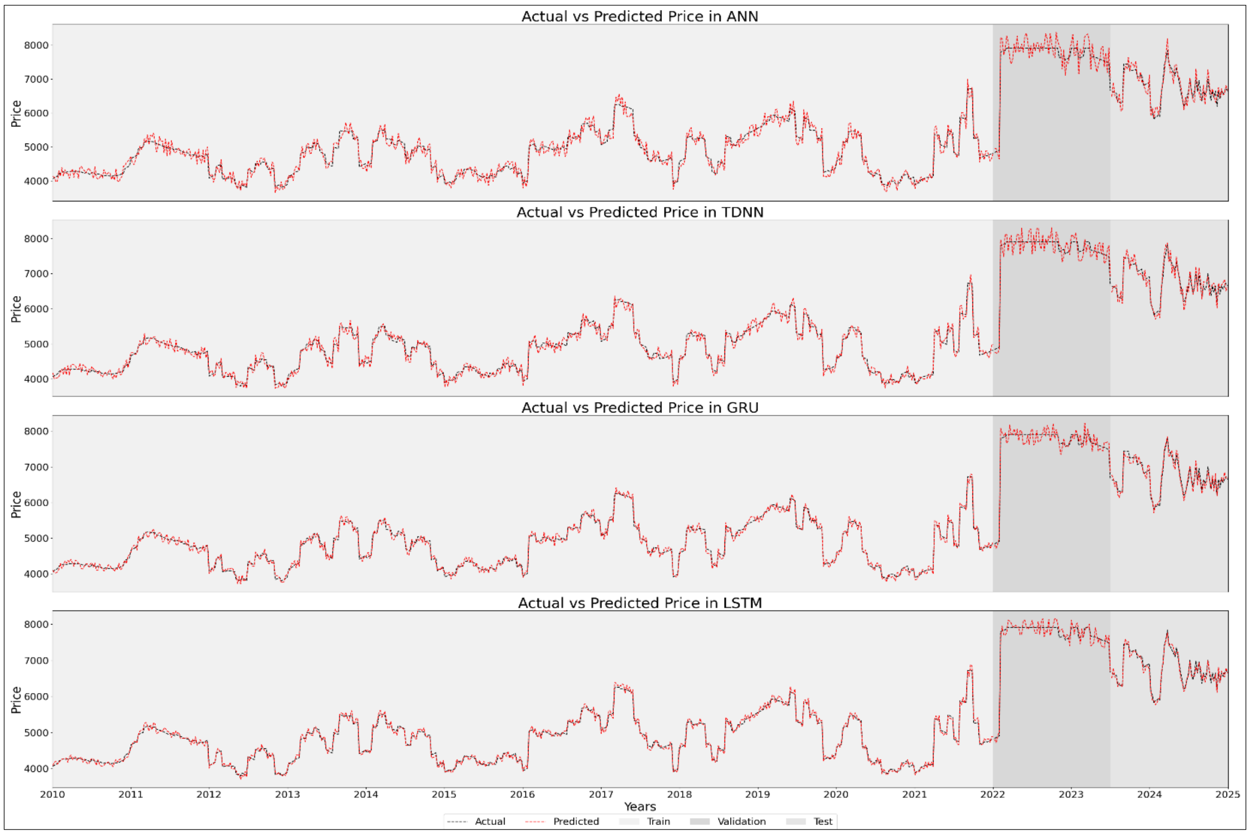

Fig. 6. Cotton price prediction of different models for Salem district.

predictions based on available facts.

Also, among the variables, there is non-linear relationship which cannot be explained in terms of correlation. A measure of non-linear dependency called Mutual Information is computed between the four weather variables and the Price. As stated earlier, Freedman-Diaconis Rule admitted that optimum number of bins was 15 to compute Mutual Information. Moderate MI values were observed: 0.236 for Tmin, 0.374 for Tmax, 0.256 for RF and 0.214 for RH. These findings suggest a moderate, non-linear association between weather parameters and cotton prices.

Following STL decomposition, the trend, seasonal and residual components of the price series were modelled independently. The final price prediction was obtained by summing the trend, seasonal and residual forecasts. The final reconstructed model fit in all the four models for Perambalur and Salem districts is depicted in Fig. 5 and 6.

To ensure a clear and comprehensive understanding of model performance, evaluation metrics have been presented separately for the training, validation and test datasets (Table 4).

Among the models, STL-LSTM consistently outperformed others, highlighting its superior predictive ability for cotton prices based on both historical and weather data.

To compare the predictive accuracy of all the different models, Diebold-Mariano test was also conducted for different combinations of all four models in both Perambalur and Salem price datasets. The p-values obtained for all model comparisons were <0.0001. Since the DM test evaluates whether the predictive errors of two models are significantly different, this result highlights that the predictive performance of the models differs substantially. Thus, it can be concluded that STL-LSTM outperforms other models in prediction of cotton prices in Salem and Perambalur districts based on weather variables and past fifteen years price data. Subsequently, the future prices of cotton in Perambalur and Salem districts are forecasted via STL-LSTM model for the next 26 weeks from January to June 2025 which is presented in Fig. 7 given below.

Conclusion

The prediction of crop prices is of significant importance for

Table 4. Quantitative evaluation metrics for prediction accuracy

Data Split	Models	Salem			Perambalur		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE
Training Data	STL-ANN	289.35	4.86	309.5861	303.49	5.64	359.4365
	STL-TDNN	267.25	3.91	324.5837	272.46	5.10	318.8218
	STL-GRU	204.12	3.06	237.2467	221.78	4.13	261.2952
	STL-LSTM	157.36	2.56	198.7826	184.51	3.45	216.3104
Validation Data	STL-ANN	387.24	5.07	436.8918	468.65	6.31	539.6506
	STL-TDNN	348.76	4.32	394.2873	438.43	5.88	496.3264
	STL-GRU	285.21	3.81	368.1927	338.78	4.61	392.4005
	STL-LSTM	248.26	3.17	296.5453	345.31	4.64	399.1131
Testing Data	STL-ANN	371.39	4.58	406.2548	422.06	5.82	498.1345
	STL-TDNN	340.35	4.02	390.5762	409.23	5.61	484.5776
	STL-GRU	285.79	3.95	319.5438	312.43	4.32	354.7807
	STL-LSTM	230.56	3.68	269.2368	295.22	4.07	337.7959

the

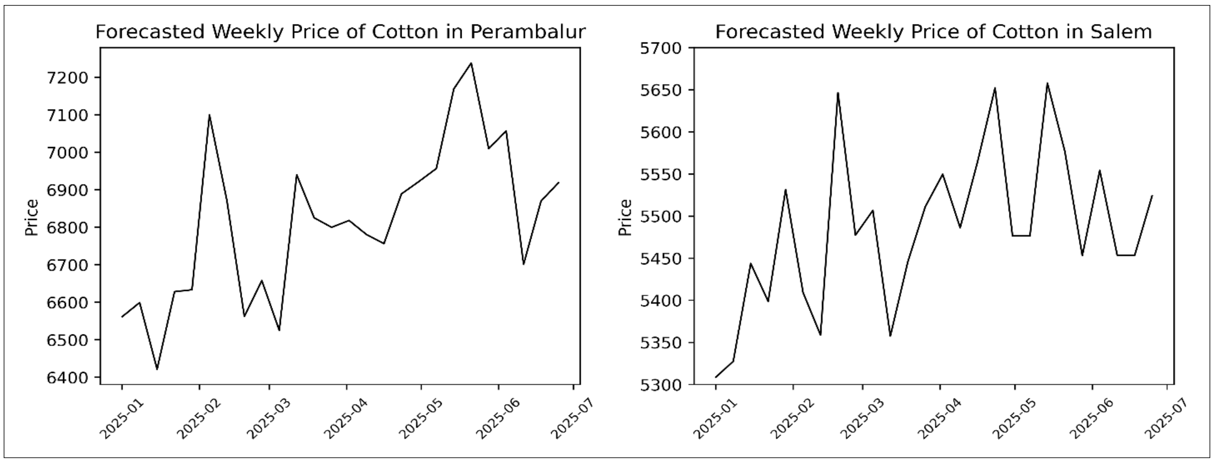

Fig. 7. Forecasted price of cotton in Perambalur and Salem.

farmers to make informed decisions and to plan crop pattern. This study presents a novel approach to cotton price forecasting in the major cotton-growing districts of Perambalur and Salem in Tamil Nadu, integrating weather variables using deep learning techniques. While the correlation between individual weather parameters Tmax, Tmin, RF and RH and cotton prices was weak, the application of Mutual Information measure effectively unveiled critical non-linear dependencies. Incorporating these exogenous variables improved the accuracy of price predictions. STL decomposition was used to uncover hidden cyclic patterns in the data, with separate models applied to the trend, seasonal and residual components. These were then ensembled to reconstruct the original price series. The forecasting model's efficacy highlights the capability of deep learning strategies to model intricate relationships within agricultural data, leading to enhanced price prediction accuracy. Furthermore, the statistical significance among the four models STL-ANN, STL-TDNN, STL-LSTM and STL-GRU was confirmed through the Diebold-Mariano test. The results of this research indicate that STL-LSTM model outperforms in predicting cotton price with MAPE 4.07 % and 3.68 % respectively, in Perambalur and Salem districts with the incorporation of weather features. Finally, the STL-LSTM model was used to forecast cotton prices for the next 26 weeks, from January 2025 to June 2025. This framework offers valuable decision-making support for farmers, helping them manage price volatility and contributing to greater economic stability in the agricultural sector. However, certain limitations exist in the model as the model was trained and validated using data from only two districts, which may limit its generalizability to other regions with different climatic or market conditions. Additionally, the dependence on historical weather and price data assumes stability in the underlying data-generating processes, which may not hold under climate change or abrupt policy shifts. Future research may explore on expanding these models by exploiting it for other regions and including additional variables such as soil conditions and broader market dynamics, which may lead to even more refined and robust predictive capabilities.

Acknowledgements

The authors are thankful to the Tamil Nadu Agricultural University for supporting their research work.

Authors' contributions

AHP and MK outlined the research idea; RP helped in collection of information in the regarding field; AHP and MV carried out analysis and brought out the interpretation; AHP prepared the manuscript draft; NVP and MK supervised the manuscript preparation and provided feedback and approval.

Compliance with ethical standards

Conflict of interest: The authors declare that they have no competing interests, financial or non-financial.

Ethical issues: None

References

1. Prashar K, Talwar R, Kant C, editors. CNN based on overlapping pooling method and multi-layered learning with SVM & KNN for American cotton leaf disease recognition. In: 2019 International Conference on Automation, Computational and Technology Management (ICACTM). IEEE; 2019. https://doi.org/10.1109/ICACTM.2019.8776730

2. India. Ministry of Textiles. Note on cotton sector [Internet]. New Delhi: The Ministry; [cited 2025 May 6]. Available from: https://texmin.gov.in/sites/default/files/Note%20on%20Cotton%20Sector.pdf.

3. Crane TA, Roncoli C, Paz J, Breuer N, Broad K, Ingram KT, et al. Forecast skill and farmers' skills: Seasonal climate forecasts and agricultural risk management in the southeastern United States. Weather, Climate and Society. 2010;2(1):44-59. https://doi.org/10.1175/2009WCAS1006.1

4. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-44. https://doi.org/10.1038/nature14539

5. Antad S. Kisan Dhan - crop price prediction using random forest. Communications on Applied Nonlinear Analysis. 2024;31:240-53. https://doi.org/10.52783/cana.v31.762

6. Chen Z, Goh HS, Sin KL, Lim K, Chung NKH, Liew XY. Automated agriculture commodity price prediction system with machine learning techniques. 2021. https://doi.org/10.48550/arXiv.2106.12747

7. Zhang D, Dai X, Wang Q, Lau CKM. Impacts of weather conditions on the US commodity markets systemic interdependence across multi-timescales. Energy Economics. 2023;123:106732. https://doi.org/10.1016/j.eneco.2023.106732

8. Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning. Cambridge: MIT Press; 2016. https://doi.org/10.4258/hir.2016.22.4.351

9. Krupesh S, Kalpana M, Venkatesa N, Shivakumar K, Vijayabhama M. Neural network framework for predicting groundnut price volatility in india. Plant Science Today. 2024;11:5849. https://doi.org/10.14719/pst.5849

10. Das P, Jha GK, Lama A. An improved cointegration based time delay neural network model for price forecasting. J Indian Soc Agric Stat. 2021;75(3):187-92. https://doi.org/10.32614/CRAN.package.ECTTDNN

11. Avinash G, Ramasubramanian V, Ray M, Paul RK, Godara S, Nayak GH, et al. Hidden Markov guided deep learning models for forecasting highly volatile agricultural commodity prices. Applied Soft Computing. 2024;158:111557. https://doi.org/10.1016/j.asoc.2024.111557

12. Panapakidis IP, Dagoumas AS. Day-ahead electricity price forecasting via the application of artificial neural network based models. Applied Energy. 2016;172:132-51. https://doi.org/10.1016/j.apenergy.2016.03.089

13. Guo Y, Tang D, Cai Q, Tang W, Wu J, Tang Q. Agricultural price prediction based on data mining and attention-based gated recurrent unit: a case study on China's hog. Journal of Intelligent & Fuzzy Systems. 2024;46(4):9923-43. http://dx.doi.org/10.3233/JIFS-235843

14. Guo Z. Research on the augmented Dickey-Fuller test for predicting stock prices and returns. Advances in Economics, Management and Political Sciences. 2023;44:101-6. https://doi.org/10.54254/2754-1169/44/20232198

15. Kagalwala A. kpsstest: A command that implements the Kwiatkowski, Phillips, Schmidt and Shin test with sample-specific critical values and reports p-values. The Stata Journal: Promoting communications on statistics and Stata. 2022;22(2):269-92. https://doi.org/10.1177/1536867X221106371

16. Olalude G. Detection of non-linearity in the time series using BDS test. Science Journal of Applied Mathematics and Statistics. 2015;3(4):184-7. https://doi.org/10.11648/j.sjams.20150304.13

17. Jaiswal R, Jha GK, Choudhary K, Kumar RR. STL decomposition based LSTM model for seasonal agricultural price forecasting. 2022. https://doi.org/10.21203/rs.3.rs-1350423/v1

18. Lem KH, editor The STL-ARIMA approach for seasonal time series forecast: A preliminary study. ITM Web of Conferences. EDP Sciences; 2024. https://doi.org/10.1051/itmconf/20246701008

19. Xu X, Zhang Y. Thermal coal price forecasting via the neural network. Intelligent Systems with Applications. Intelligent Systems with Applications. 2022;14:200084. https://doi.org/10.1016/j.iswa.2022.200084

20. Ish-Horowicz J, Reid J. Mutual information estimation for transcriptional regulatory network inference. BioRxiv. 2017:132647. https://doi.org/10.1101/132647

21. Contreras-Reyes JE, Brito A. Refined cross-sample entropy based on freedman-diaconis rule: application to foreign exchange time series. Journal of Applied and Computational Mechanics. 2022;8(3):1005-13. https://doi.org/10.22055/jacm.2022.39470.3412

22. Nikou M, Mansourfar G, Bagherzadeh J. Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. Intelligent Systems in Accounting, Finance and Management. 2019;26(4):164-74. https://doi.org/10.1002/isaf.1459

23. Banerjee C, Mukherjee T, Pasiliao Jr E, editors. An empirical study on generalizations of the ReLU activation function. Proceedings of the 2019 ACM Southeast Conference; 2019. https://doi.org/10.1145/3299815.3314450

24. Mastromichalakis S. ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance. 2020. https://doi.org/10.48550/arXiv.2012.07564

25. Narkhede MV, Bartakke PP, Sutaone MS. A review on weight initialization strategies for neural networks. Artificial Intelligence Review. 2022;55(1):291-322. https://doi.org/10.1007/s10462-021-10033-z

26. Yang M, Lim MK, Qu Y, Li X, Ni D. Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction. Expert Systems with Applications. 2023;213:118873. https://doi.org/10.1016/j.eswa.2022.118873

27. Guo L. Extreme learning machine with elastic net regularization. Intelligent Automation & Soft Computing. 2020;26(3):421-7. https://doi.org/10.32604/iasc.2020.013918

28. Srivastava N. Improving neural networks with dropout. MSc [dissertation]. Toronto: University of Toronto; 2013. Available from: https://www.cs.toronto.edu/~nitish/msc_thesis.pdf

29. Scientific Research Publishing. Advances in Artificial Neural Networks. Wuhan (China): Scientific Research Publishing; 2012. Available from: https://www.scirp.org/book/detailedinforofabook?bookid=2716&booktypeid=1

30. Bland C, Tonello L, Biganzoli E, Snowdon D, Antuono P, Lanza M. Advances in artificial neural networks. Scientific Research Books; 2020. p. 119.

31. Nayak GH, Alam MW, Singh K, Avinash G, Kumar RR, Ray M, et al. Exogenous variable driven deep learning models for improved price forecasting of TOP crops in India. Scientific Reports. 2024;14(1):17203.

32. Jha GK, Choudhary K, Jaiswal R. EEMD-FCR-TDNN: a hybrid model for forecasting agricultural commodity prices. Indian Journal Extension Education. 2023;77:79-88.

33. Zou H, Xia G, Yang F, Wang H. An investigation and comparison of artificial neural network and time series models for Chinese food grain price forecasting. Neurocomputing. 2007;70(16-18):2913-23. https://doi.org/10.1016/j.neucom.2007.01.009

34. Jha GK, Sinha K. Agricultural price forecasting using neural network model: An innovative information delivery system. Agricultural Economics Research Review. 2013;26(2):229-39. http://dx.doi.org/10.22004/ag.econ.162150

35. Mehtab S, Sen J, Dutta A, editors. Stock price prediction using machine learning and LSTM-based deep learning models. Symposium on machine learning and metaheuristics algorithms and applications. Springer; 2020. https://doi.org/10.1007/978-981-16-0419-5_8

36. Rezaei H, Faaljou H, Mansourfar G. Stock price prediction using deep learning and frequency decomposition. Expert Systems with Applications. 2021;169:114332. https://doi.org/10.1016/j.eswa.2020.114332

37. Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint. 2014. arXiv:1412.3555. https://arxiv.org/abs/1412.3555

38. Kurumatani K. Time series forecasting of agricultural product prices based on recurrent neural networks and its evaluation method. SN Applied Sciences. 2020;2(8):1434. https://doi.org/10.1007/s42452-020-03225-9

39. Chaudhary M. AI aided tools for fresh produce yield and price forecasting: deep learning approaches: University of Waterloo; 2021.

Additional information