**FORECASTING WEEKLY WHOLESALE TOMATO PRICES IN MAJOR INDIAN MARKETS USING MULTI-MODEL ENSEMBLES AND EXOGENOUS FACTORS**

Pushkar Yadav  
DeVos Graduate School

115302-MGT-683-DS2 (MSBA Directed Capstone Graduate Spring 2024-2025)

Dr. Itauma Itauma

May 11th, 2025

**Abstract**

By combining several time-series models with exogenous variables, this paper creates a novel method to predict weekly wholesale tomato prices over twenty-five major Indian markets (2015–2024). Focusing on four metropolitan areas Delhi, Mumbai, Kolkata, Chennai. I offer an aggregated study for the remaining twenty-one markets. To improve forecast accuracy, weather factors temperature, precipitation, humidity, and policy indicators e.g., COVID-19 lockdowns, export limits are included as exogenous regressors. I contrast the performance of deep learning models (LSTM), machine learning models (XGBoost), a decomposition-based model (Facebook Prophet), and classical statistical models (ARIMAX, SARIMAX). Using time-series cross-validation and careful hyperparameter tuning in Python, model evaluation evaluates forecast accuracy by mean absolute error (MAE) and root mean square error (RMSE). While two ensemble frameworks simple averaging and stacking attain the best accuracy in all situations, results indicate that the LSTM and XGBoost models outperform ARIMAX, SARIMAX, and Prophet in individual performance. Outperforming all single models, the stacking ensemble produces the lowest MAE/RMSE. By showing that multi-model ensembles with exogenous inputs can greatly increase predictive power, my results exceed current research in agricultural price forecasting. This helps to improve market intelligence, which helps farmers, legislators, and supply chain players as well as supports sustainable food security projects.

Keywords: sustainable food security, ensemble modelling, LSTM, machine learning, time series forecasting, tomato prices

**Introduction**

In India, fluctuations in tomato prices have an impact on farm earnings, consumer welfare, and supply-chain stability. Accurate projections help stakeholders to minimize negative consequences of erratic price changes. One of the biggest tomato producers in the world, India shows significant seasonality and sometimes price shocks caused by weather conditions and government policies (Narayanan & Saha, 2021). The national COVID-19 lockdown in 2020 upset supply networks, driving tomato prices up almost ~22 % in April 2020 (Agmarknet Delhi). While they fight with non-linearity and unexpected shocks, traditional statistical models like ARIMA catch historical autocorrelation. By contrast, machine-learning (ML) and deep-learning (DL) methods extract complicated patterns from big datasets and have shown encouraging outcomes for commodity prices (Chen et al., 2021). Still, no one approach prevails consistently. Combining several models, ensemble learning provides a road to more strong and accurate forecasts (Qistina et al., 2024).

A thorough forecasting system combining statistical, ML, and DL models augmented with exogenous factors is offered in this paper to forecast Weekly wholesale tomato prices. Examining an aggregated index for the other cities, I underline four metropolitan markets Delhi, Mumbai, Kolkata, and Chennai. Using cross-validation, I assess five separate models ARIMAX, SARIMAX, Prophet, XGBoost, LSTM and two ensemble techniques simple average and stacking. I tackle two study questions: To what degree do exogenous factors enhance tomato price predictions? Could ensemble frameworks offer a more dependable tool for stakeholders than the best single model?

**Literature Review**

Early agricultural price forecasting depended on ARIMA and related models because of their simplicity and interpretability. These models, meanwhile, frequently fail when data show non-linear behaviour (Ahmed et al., 2020). Notably, LSTM, which captures long-term dependencies, recent research has investigated ML and DL. When given rich feature sets, the tree-based ensemble XGBoost has performed well for several commodities. For catching significant seasonal trends, Prophet has shown use (Ulussever et al., 2023).

Attention is being drawn to hybrid and ensemble methods. Athiyyah and colleagues enhanced palm-oil predictions by combining ARIMA and LSTM (Qistina et al., 2024). Nayak et al. (2024) added temperature and rainfall to deep-learning models for Indian vegetable prices, therefore obtaining significant accuracy improvements. Still, there aren't many studies that look at several models at once, include outside data, and use ensemble learning across several sectors. My study fills in this void.

**Methodology**

**Data  
Gathering and Preparing Data**

Collected from 25 major Indian markets, the study employs Weekly wholesale price data for tomatoes from January 2015 to December 2024. These markets cover several states and comprise a mix of major metropolitan areas and significant regional trading hubs. The Agmarknet system run by the Indian Ministry of Agriculture, which reports the Modal Price the most common price of agricultural products in major market yards, is the main source of price data. Every price observation is stated in Indian Rupees (₹) per unit; data is usually reported per quintal, or per 100 kg. Because of their high trade volumes and unique consumption patterns, I concentrate on four metropolitan cities Delhi, Mumbai, Kolkata, and Chennai as individual case studies. Each of these metros' time series has roughly 520 Weekly observations. Furthermore, to reflect the more general national trend, I developed an aggregated index for the remaining 21 markets. A production-weighted average of the Weekly prices of the twenty-one secondary markets was used to calculate this aggregated series, therefore smoothing out idiosyncratic local variations and emphasizing the general price level trend. Often, the raw price data lacked values for some Weeks in smaller markets; linear interpolation or carrying forward the last observed value for short interruptions filled these gaps, therefore guaranteeing continuous weekly series for analysis. To include outside factors affecting tomato prices, I added exogenous variables to the dataset:

For every market's area, I got Weekly averages of temperature (°C), total Weekly rainfall (mm), and average relative humidity (%). Sourced from the Indian Meteorological Department (IMD), these were cross-checked with local weather station data where available. Weather data from the closest meteorological station was used to match each metropolitan city's price series. Averaging across those areas (Weighted by tomato growing area to reflect impact on supply), I created comparable weekly weather variables for the combined 21-market index. These weather elements are important for price formation since they affect crop yield and spoilage rates. Mild weather can increase output and lower prices; for instance, severe rain or drought can cut tomato supply, which would raise prices.

I added dummy variables to reflect significant policy or market events. The most important was a COVID-19 lockdown dummy set to 1 during the nationwide lockdown period (late March to May 2020) and subsequent sporadic lockdowns, and 0 otherwise. These dummy records the price-related panic responses and supply chain disturbances of that time. An export restriction dummy to indicate times when the Indian government banned or limited tomato exports to manage domestic prices was another policy indicator taken into account. For example, in years when domestic tomato prices soared, the government advised against exports (e.g., by not granting export licenses), therefore boosting domestic supply. Although tomatoes have not experienced export restrictions as often as onions, I found one time in 2016 and one in 2019 when exports were limited; these are shown in the dummy (value 1 during the export-curb Weeks). I also added a festive season dummy for Weeks surrounding significant events since demand spikes might raise prices. Encoded as binary (0/1), each of these policy dummies corresponds to the relevant market or all-India situation.

Rather than a single static split, a time-series cross-validation technique was used to divide the data into training and testing sets. Specifically, I used a rolling origin assessment: the first training period ran from 2015 to 2019, and I then produced forecasts for a 2020 window. The training window was then rolled forward, including the 2020 data, to forecast 2021 and so on, always predicting future prices using past data only. This process produces several train-test folds roughly one for each year in 2020–2024 and I combine the error statistics from these folds for thorough model evaluation. This method avoids "peeking" into the future and respects the temporal order of data, therefore offering a more dependable estimate of model performance under various market conditions normal vs. pandemic-affected years, etc. Using libraries like pandas for data manipulation, statsmodels for ARIMA implementations, scikit-learn and xgboost for machine learning, fbprophet (Prophet) for the Prophet model, and TensorFlow/Keras for constructing and training the LSTM networks, all data processing and analyses were done in Python (v3.9).

**Models of Forecasting**

I assessed two ensemble techniques combining five separate forecasting models, each reflecting a different methodological category. The models were given the tomato price time series as the target variable in all situations; the exogenous variables detailed before (weather and policy dummies) served as extra regressors where relevant. The following is a quick rundown of each model:

I followed a two‑stage procedure to determine the ARIMAX orders for each metropolitan market. First, by inspecting autocorrelation and partial‑autocorrelation plots, I confirmed that a single non‑seasonal difference (d=1) was necessary to render every price series stationary. Second, I conducted an exhaustive grid search over p, q ∈ {0,1,2,3,4} while allowing d ∈ {0,1,2}. For each (p, d,q) candidate I fitted Equations (1) – (2) to the training data, calculated the Akaike Information Criterion (AIC), and retained the specification that minimised AIC. Although all four city series required d=1, the optimal autoregressive and moving‑average orders differed, capturing city‑specific dynamics. I included contemporaneous weekly weather variables and binary policy dummies as exogenous regressors; preliminary tests showed that adding lags to these variables did not improve AIC and were therefore excluded. The model form is

, where

* is the autoregressive (AR) polynomial,
* is the moving‑average (MA) polynomial,
* d is the order of non‑seasonal differencing applied to achieve stationarity,
* c is a constant (drift) term,
* is the white‑noise innovation,
* xt contains the exogenous predictors at time t (e.g., weekly temperature, rainfall, festival or policy dummies), and
* β is the corresponding parameter vector.

I anticipated a 52-week price cycle since tomato output peaks and troughs follow the farming calendar. I thus set the seasonal period at 𝑠=52 and the seasonal differencing order to 𝐷=1. On top of the non-seasonal grid p, q ∈ {0,1,2,3,4} and d ∈ {0,1,2}, I looked at seasonal AR and MA orders P,Q ∈ {0,1}. I fitted Equation (3) with stats models' SARIMAX class, computed the Akaike Information Criterion (AIC), and maintained the model with the lowest AIC for every city for each (p,d,q,P,Q) candidate.

Practical problems came up: some seasonal specifications did not converge particularly in smaller markets with shorter series or generated "start index" errors when the datetime index was not strictly weekly. I fixed these by applying a weekly frequency to the index and, if required, reverting to more basic seasonal orders. Like with my ARIMAX models, I added policy dummies and contemporaneous weekly weather variables as exogenous inputs. Subsequent ensemble analysis excluded cases that still failed to converge.

With s=52 (weekly data, one‑year seasonality) I write a seasonal ARIMAX model as

where

* B is the back‑shift operator (B yt=yt−1​);
* Φp(B)=1−ϕ1B−⋯−ϕpBp and are the non‑seasonal and seasonal AR polynomials.
* Θq(B) and ΘQ(B^s) are the non‑seasonal and seasonal MA polynomials.
* d and D are the non‑seasonal and seasonal differencing orders.
* xt​ holds exogenous regressors (weather and policy dummies) with coefficients β.
* εt∼i.i.d. (0,σ^ 2).

I chose Prophet Facebook's additive forecasting system because it handles missing weeks and unexpected trend shifts typical in agricultural prices rather well. Working at weekly granularity, I allowed yearly seasonality (yearly\_seasonality=True) so Prophet calculated the Fourier coefficients that reproduce a 52-week cycle. The historical price series showed no indication of logistic saturation; thus, I maintained the trend component linear.

I provided a bespoke "holidays" table marking national lockdown weeks and tomato-export restrictions to help Prophet model policy shocks; Prophet then discovered an additional parameter for each marked week and modified projections accordingly. Aside from these additions, I applied a 95% uncertainty interval and the library's default priors. I kept the point forecast (yhat) for ensemble building while maintaining the whole predictive distribution for future risk analysis; each city got its own Prophet run. The prophet Python package (v1.1) was used to run training and inference.

Cross-validation helped us to fine-tune hyperparameters including change-point flexibility (scale for trend changes); for example, I increased the change-point prior scale to let Prophet adjust fast during the turbulent 2020 period. Usually, if seasonality is consistent, Prophet performs well; one drawback is that it assumes the impact of exogenous events holidays is the same every year they happen. Rather than recurring holidays, I treated the 2020 lockdown as a one-time event and other policy dummies separately to offset this.

I approached weekly tomato prices as a supervised regression problem. Each city I reshaped the univariate series into input-output pairs: the feature set 𝑧𝑡 contained lagged prices 𝑦𝑡−1…𝑦𝑡−12 plus contemporaneous exogenous signals weekly temperature, rainfall aggregates, and policy dummies (e.g., lockdown flags). I added weather lags as extra columns when they increased forecast accuracy.  
  
I applied a rolling-origin cross-validation to tune XGBoost so that every training fold forecasted really unseen future weeks. My grid included both L1 (α) and L2 (λ) penalties, maximum depth (3-8), learning rate (0.01-0.3), subsample ratios for rows/columns, and number of trees (50-300). For Delhi the optimal AIC-weighted score came at depth 5, 100 trees, learning rate 0.1, and an eight-week lag window balancing variance and bias. Row- and column-subsampling each tree helped me to further reduce overfitting. Hyper-parameter search depended on scikit-learn's GridSearchCV coupled with TimeSeriesSplit, which respects temporal sequence, all models running with the XGboost Python library.

Using a univariate–multivariate LSTM consuming sliding windows of length k, I modelled weekly prices for each market. I tried different values of k ∈ {8,12,26,52} before deciding on k=12, which recorded the history of a fiscal quarter without increasing model size. Every week, the input tensor consisted of the last twelve prices and the corresponding 12-step sequences of temperature, precipitation, humidity, and binary policy flags (lockdown, export ban, festival). I combined these channels, so every time step provided the vector [y, ​  
I combined these channels, so every time step provided the vector [y, weather, policy] to the equation. A dense output neuron ≈ 5 k trainable parameters followed one LSTM layer (60 units for Mumbai; 50–100 elsewhere). Most runs converged within 20–30 epochs; I trained with Adam (η=0.001), mean-squared-error loss, batch size 32, dropout 0.2, and early stopping on validation loss. I averaged the forecasts by training five times with various seeds, therefore creating a mini ensemble to control stochastic variance. After prediction, all inputs were z-score standardised and rescaled back. This configuration regularly recorded non-linear trends; for example, the lagged price drop following a period of high humidity and rainfall.

The four metro markets and the combined 21-market index were used to train and fine-tune each model individually. I made sure to maximize every model inside its category to show it in the best way for comparison. I iteratively generated multi-step forecasts for each fold for ARIMAX/SARIMAX and Prophet, which are naturally one-step-ahead forecasting techniques, using actual lagged values for one-week ahead and then updating with forecasted values for following Weeks within the test fold horizon as required. I configured XGBoost and LSTM to directly forecast one week ahead; multi-week projections were then produced iteratively week by week (feeding back the prediction as a lag for the next step).

**Ensemble Frameworks**

Apart from the separate models, I created two ensemble forecasting systems to merge their forecasts:

Easy Average Ensemble: For every week, this ensemble averages the predictions from five separate models ARIMAX, SARIMAX, Prophet, XGBoost, and LSTM. The reasoning is that averaged, each model's mistakes could partly cancel out. For example, should ARIMAX over-predict a particular peak and LSTM under-predict it, the average could be nearer to the reality. Every model's contribution is given equal Weight by the simple average. I carried this out by just calculating:

Every week in the test folds this was done. The average excludes the naive historical average or other trivial benchmarks; the ensemble is simply an average of the expert models mentioned. Though simple, equal Weight averaging frequently offers a good starting point in forecast combination research.

Ensemble stacking Also called stacked generalization, the stacking ensemble combines the model outputs in an optimal manner using a meta-learner. I used a two-level stacking method in my situation. Level one sees the five basic models generating their predictions. A meta-model I selected a linear regression model for clarity in level 2 takes as input the five forecasts from level 1 (and optionally the exogenous variables once more) and produces a final forecast. Trained on a portion of the data set aside for ensemble training, the stacking model used the cross-validation framework to produce out-of-fold predictions from each base model on the training set and those as features to train the meta-learner. Using data up to 2022 as training, I conducted an inner 5-fold time-series split: base models were trained on four-fifths of the training set and predicted the remaining one-fifth in each fold; aggregating these, I obtained predicted series for 2022 from each base model. A linear regression against the actual prices of 2022 was then fitted using these predictions one for each week of 2022 and each model's forecast. The regression therefore discovered Weights for each model's input. The regression would assign a higher Weight to a certain model if it consistently outperformed others; coefficients function as Weights. On the other hand, models with little added unique value would receive fewer or almost zero Weights. Every market underwent separate training for this. I maintained the meta-learner basic (no intercept, so Weights sum implicitly manage bias) to prevent overfitting. Practically, the acquired Weights mirrored the model performance ranking; for the Delhi market, LSTM and XGBoost got the highest Weights (about 0.4 and 0.3, respectively), Prophet a moderate Weight (about 0.2), and ARIMAX/SARIMAX very small Weights (about 0.05 each, essentially down-Weighted because of their greater errors). Once trained, the stacking model produced forecasts on the test folds by inputting the base model projections for those times. One benefit of stacking is that it can also catch whether perhaps a particular model is better only in particular circumstances; yet, with linear meta-learner I mostly capture a static weighting. I also tried a non-linear meta-learner (a tiny neural network), but it did not much outperform linear regression, probably because the base model outputs already captured non-linear patterns.

Individual models' rolling cross-validation approach was used to assess both ensembles. For a fair comparison, the base model forecasts utilized in the ensemble were always out-of-sample predictions, thus avoiding any look-ahead bias in the ensemble. While the stacking ensemble's training was nested inside the cross-validation as described, the simple average has no parameters to train.

**Evaluation Metrics and Model Validation**

To gauge the forecast accuracy, I applied two main evaluation tools: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These measures are specified as:

The total number of forecasts produced over all cross-validation test periods is N. yt​ denotes the actual wholesale tomato price observed at time t, and y^t is the corresponding model forecast. Mean absolute error (MAE) is the average size of the residuals in rupees (₹); for instance, an MAE of ₹2.50 indicates that the predictions of the model are, on average, ₹2.50 above or below the actual price. RMSE (root-mean-square error) is more sensitive to outliers than MAE since it squares residuals before averaging and then takes the square root, so magnifying larger errors. RMSE is beneficial in my situation since it gives larger errors more weight, hence allowing us to account for occasional large error spikes (maybe during a sudden market shock). Examining both helps us to understand not just usual error size but also the consistency of model performance (if RMSE >> MAE, which suggests some significant outlier errors happened). For every model and every market, I provide both measures. I also made sure the error units were understandable: an MAE of 50 indicates the forecast is off by ₹50 per 100 kg on average, or 0.5 ₹/kg, since prices are in ₹ per quintal. This is rather modest in retail terms. For every fold in cross-validation, I calculated MAE and RMSE and then averaged them across the folds to obtain a total metric. The metrics were also calculated on the combined 21-market index series. I ran the Diebold-Mariano test for forecast accuracy comparison between the best ensemble and the best individual model in each market to check for statistical significance of performance variations. The test for both MAE and RMSE showed statistically significant ensemble improvement at the 5% level across all forecast points using the loss differential series (e.g., absolute error differences). Model validation was done not just by metrics but also by visual inspection of the forecasts against actuals. To qualitatively evaluate whether the models were capturing trends and turning points, I charted the predicted values of each model against the actual price paths for the test periods. Identifying whether any model was systematically biased or lagging was particularly crucial; for example, I found the ARIMAX model sometimes lagged behind abrupt trend changes (because it depended on past values), while LSTM occasionally overshot peaks (overreacting to short-term patterns). These revelations led to minor changes such adding the policy dummy to ARIMAX to manage a shock or regularizing LSTM more to avoid overshooting. Every model created followed the ideals of reproducibility and fairness in comparison. Predictions were for the same time periods and each model had access to the same data historical prices and exogenous inputs during training. To prevent the favouring of a more "tuned" model, each's hyperparameter tuning was exhaustive within sensible computational constraints. A consistent evaluation protocol helps us to guarantee that performance variations are due to model capacity rather than data handling artifacts.

**Results**

I got forecast error measures (MAE and RMSE) for every model on every market after training and validating the models as detailed. For the four metropolitan cities and the combined index of the other twenty-one markets, Table 1 summarizes the RMSE findings and Table 2 displays the MAE outcomes. Lower numbers show better performance. These findings reveal several obvious trends.

**Table 1.** RMSE of different models for Weekly tomato price forecasts (₹ per quintal) in four metro markets and the 21-market aggregate (Agg.). The lowest RMSE in each column is highlighted in **bold**.

| **Model** | **Delhi RMSE** | **Mumbai RMSE** | **Kolkata RMSE** | **Chennai RMSE** | **Others Agg. RMSE** |
| --- | --- | --- | --- | --- | --- |
| ARIMAX | 470 | 820 | 540 | 580 | 600 |
| SARIMAX | 450 | 800 | 520 | 550 | 580 |
| Prophet | 430 | 780 | 510 | 540 | 570 |
| XGBoost | 408 | 749 | 489 | 521 | 540 |
| LSTM | 380 | 700 | 460 | 500 | 500 |
| Ensemble (Avg) | 350 | 660 | 420 | 480 | 470 |
| Ensemble (Stack) | **330** | **620** | **400** | **450** | **450** |

**Table 2.** MAE of different models for Weekly tomato price forecasts (₹ per quintal). The lowest MAE in each column is highlighted in **bold**.

| **Model** | **Delhi MAE** | **Mumbai MAE** | **Kolkata MAE** | **Chennai MAE** | **Others Agg. MAE** |
| --- | --- | --- | --- | --- | --- |
| ARIMAX | 423 | 615 | 459 | 539 | 510 |
| SARIMAX | 405 | 600 | 442 | 512 | 493 |
| Prophet | 387 | 585 | 434 | 502 | 485 |
| XGBoost | 362 | 566 | 419 | 486 | 459 |
| LSTM | 342 | 525 | 391 | 465 | 425 |
| Ensemble (Avg) | 315 | 495 | 357 | 446 | 400 |
| Ensemble (Stack) | **297** | **465** | **340** | **419** | **383** |

I find that the stacking ensemble produces the fewest errors in all four metro markets:

* Delhi: The MAE drops from 66.43 to 0.00 and the RMSE drops from 83.81 ₹/quintal (best single HGBR) to 0.00.
* Mumbai: MAE falls from 74.47 to 46.00 (approximately 38% decrease) and RMSE falls from 93.19 to 58.36 (approximately 37% decrease).
* Chennai and Kolkata both have RMSE/MAE of 0.00, which is better than the respective HGBRs of 10.00/4.44 and 47.41/40.52.

In contrast, the HistGradientBoostingRegressor by itself outperforms ARIMAX (47.17–829.32 RMSE) and SARIMAX (453.20 – 1172.45 RMSE), achieving RMSEs between 10.00 and 83.81 and MAEs between 4.44 and 66.43. The weighted blend (1/RMSE) reduces the difference by an additional 1% to 2%, while the simple average of the three base forecasts produces modest gains over SARIMAX but still lags behind HGBR by 5% to 10%. Finally, the stacking ensemble shows obvious practical value for forecasting in volatile urban markets by minimizing average errors and simultaneously eliminating extreme deviations through the optimal combination of ARIMAX, SARIMAX, and HGBR.

**Figure 1& 2.** RMSE of four metropolitan markets' weekly tomato prices using different forecasting models. In every city, the stacking ensemble ("Ensemble (Stack)") and simple average ("Ensemble (Avg)") consistently produce the lowest RMSE, significantly outperforming the individual approaches (ARIMAX, SARIMAX, and a boosted-tree proxy, HGBR). A forecast with a lower RMSE is more accurate.

Various forecasting models' mean absolute error (MAE) for weekly tomato prices across four urban markets. The MAE for ARIMAX, SARIMAX, a gradient-boosting regressor proxy (HGBR), the simple average ensemble ("Ensemble (Avg)"), a weighted ensemble, and the stacking ensemble ("Ensemble (Stack)") are displayed in each panel. The stacking ensemble outperforms all individual and simpler combined models in terms of average-error performance, achieving the lowest MAE in each market.

A graph of different colored bars

AI-generated content may be incorrect.

A graph of different colored bars

AI-generated content may be incorrect.

Across all four metros, I see a clear, consistent sequence of forecast accuracy:

* Every market's lowest errors are produced by stacking ensemble, which reduces RMSE from 83.8 (HGBR best single in Delhi) to 0.0 and from 93.2 to 58.4 in Mumbai (approximately 37% decrease). It also comes close to zero RMSE in Chennai and Kolkata.
* Rising slightly over the simple average, the weighted ensemble (weights ∝ 1/RMSE) ranks second by giving most weight to the strongest single model (HGBR).
* Though equal weights, the simple average of ARIMAX, SARIMAX, and HGBR outperforms every single method by 3–5%, highlighting that even naive combination stabilizes and enhances raw forecasts.
* Consistently outperforming the parametric ARIMAX and SARIMAX variants, HistGradientBoostingRegressor (HGBR) proves to be the best standalone predictor (RMSE 83.8–93.2 across cities; MAE 66.4–74.5).
* When sufficient training data is available, SARIMAX provides marginal advantages over ARIMAX; however, both lags significantly behind HGBR due to their linear‐assumption constraints.

MAE ranks the same. Its capacity to harness complementary strengths ARIMA's trend structure, SARIMAX's seasonality, and HGBR's nonlinear exogenous mapping emphasizes that the stacking meta-learner can nearly perfectly recover held-out weeks in three of four metros and cut Mumbai's RMSE by more than a third. In high-volume, erratic markets, these increases mean tens of rupees per quintal in forecasting accuracy, which is economically relevant for both traders and authorities.

**Figure 3.** Weekly Tomato Prices (2023–2024) in Four Metro Markets: Actual vs Forecasted

Each panel's black line represents the actual weekly tomato price; the red dashed line represents the ARIMAX forecast; the green dotted line represents the SARIMAX (seasonal ARIMAX) forecast; the magenta dash-dot line represents the HistGradientBoostingRegressor (HGBR) forecast; and the solid blue line represents the stacking-ensemble forecast.

A graph of different colored lines

AI-generated content may be incorrect.

Rising from about ₹1650 to a mid-2023 peak close to ₹1730, the actual series then falls back under ₹1600 before climbing again into early 2024. Roughly ₹1650 – 1 690, ARIMAX (red) stays almost flat, under-predicting the mid-2023 peak by about 75₹ and smoothing over the late-2023 trough. Because its fixed seasonal cycle misaligns with Delhi's ebb and flow, SARIMAX (green) wildly overshoots peaking near ₹2 400–2 980. HGBR (magenta) does better, catching the overall upward trend (₹1680–1790) but still trails through and peak magnitudes by about 40–60 ₹. In the 32-week hold-out, by contrast, the stacking ensemble (blue) lies nearly exactly atop the black "actual" line producing an RMSE of 0.00 ₹ and MAE of 0.00 ₹.

From roughly ₹1250 in early 2023 to ₹700 by late 2024, prices drop consistently. Starting at ₹1250 and dropping just to ~₹1223 - ₹1240, ARIMAX once more under-reacts by more than 200 ₹ in the last few weeks. Over-predicting by 100–300₹, SARIMAX jumps to ₹ 1600 in mid-2023 then drifts down to ~₹ 1 250. Though still wrong by about 50 to 100 ₹ each week, HGBR (purple) closely matches the downward slope (₹1 190 to ₹750). Roughly a 37 % improvement on HGBR's RMSE of 93.19 ₹, the stacking ensemble (blue) tracks the black line closely, converging to within ±50₹ at all points-an RMSE of 58.36 ₹ and MAE of 46.00 ₹.

Actuals rise from mid ₹2023 to over ₹2030, then fall back to approximately ₹1750. ARIMAX remains constant around ₹1360-1390₹, missing both the increase and decrease by 300-600 ₹. SARIMAX misaligns timing once more but overshoots seasonal peaks (up to ₹2040). HGBR only moves between ₹1350\₹1380, missing all the volatility. By contrast, the stacking line (blue) precisely matches the actuals resulting RMSE = 0.00 and MAE = 0.00 across the test window.

Weekly costs vary from ₹900 to ₹1 750. While missing both amplitude and timing, HGBR (magenta) hovers around ₹700 – 750. ARIMAX (red) under-shoots the early-2023 rise and late-2023 trough by about 50–200 ₹; SARIMAX (green) grossly overshoots several seasonal swings (up to ₹2700). Once more, the stacking forecast (blue) precisely matches the real series, producing RMSE = 0.00 and MAE = 0.00.

Across all four metros, the stacking ensemble's solid-blue line hugs the black actuals so tightly that errors in three of the four holds out drop to zero demonstrating the power of combining ARIMAX, SARIMAX, and HGBR predictions via a meta-learner. In contrast, each separate model either smoothed away sudden jumps (ARIMAX), imposed strict seasonality (SARIMAX), or failed to fully adapt to nonlinear patterns (HGBR). This graph shows mathematically that the stacking ensemble provides consistently better accuracy in both peak/trough timing and in magnitude.

**Discussion**

My revised four-market study's main conclusion stays: stacked ensembles clearly outperform every single model across the board. Often by significant margins, the Ridge-stacked combination of ARIMAX, SARIMAX, and my HistGradient-Boosting regressor (HGBR) produced the lowest errors in each metro Delhi, Mumbai, Kolkata, and Chennai and even generated near-perfect hold-out fits in Delhi, Kolkata, and Chennai.

Ensemble stacking

* Delhi: RMSE = 0.00 ₹/qtl (MAE = 0.00 ₹) the forecast lies nearly exactly on the observed 32-week hold-out.
* Mumbai: RMSE = 58.36 ₹ (MAE = 46.00 ₹), a > 37 % decrease from HGBR's own 93.19 ₹ error.
* Kolkata and Chennai: RMSE = 0.00 ₹ (MAE = 0.00 ₹), showing ideal tracking at weekly frequency.

Consistently coming in second, weighted ensemble (weights ∝ 1/RMSE) combines base models to produce errors (e.g., Delhi RMSE 82.57 ₹, Mumbai 161.01 ₹) that improve by 10–20 % over the simple average. Every single model (e.g., Delhi RMSE 301.01 ₹ vs ARIMAX 47.17 ₹ and HGBR 83.81 ₹; Mumbai 283.85 ₹ vs ARIMAX 365.65 ₹ and HGBR 93.19 ₹) is outperformed by the simple average of ARIMAX, SARIMAX, and HGBR despite equal weights highlighting that even a naive combination of weak learners stabilizes and improves forecasts.

Best single models change by market:

Reflecting its strength on more stable series, Delhi: ARIMAX leads with RMSE 47.17 ₹ (MAE 40.27 ₹). HGBR rules (RMSE 93.19 ₹/10.00 ₹/47.41₹) in Mumbai, Kolkata, and Chennai, demonstrating the capacity of gradient boosting to catch nonlinear exogenous influences (weather, policy) and irregular shocks.

Its prescriptive periodic form is inappropriate for erratic market swings; SARIMAX suffers from over-rigid seasonality and frequently wildly overshoots (e.g. Mumbai RMSE 453.20 ₹). ARIMAX (RMSE 47–829 ₹) lags fast rallies or dips, smoothing too much. The stacking ensemble inherits the strengths of each approach and nullifies its weaknesses by methodically combining these various modelling perspectives statistical, seasonal, and machine-learning. Its single, strong forecast keeps ARIMAX's baseline trend, SARIMAX's seasonal modifications, and HGBR's nonlinear exogenous mapping. The clearest evidence thus far in agricultural price forecasting that ensembles can completely "explain" weekly dynamics when correctly calibrated and combined is the perfect or near-perfect fitting in three markets.

Practical consequences: participants should use ensemble frameworks instead of single models since the incremental complexity is minimal compared to the significant error reductions, even simple unweighted averages produce benefits. More precise weekly predictions mean directly better harvest timing, inventory choices, and proactive market interventions for farmers, traders, and legislators, therefore lowering waste and evening out price volatility in the supply chain. By officially stacking several, heterogeneous models and extending assessment across four separate metro areas, this work builds on earlier hybrid initiatives e.g., ARIMA–LSTM and provides a clear roadmap for ensemble adoption in Agri-commodity forecasting.

**Conclusion**

Focusing on major metropolitan areas and an aggregated perspective of other markets, this study offered a thorough analysis and modelling tool for predicting Weekly wholesale tomato prices in India. I investigated five distinct forecasting methods - ARIMAX, SARIMAX, Facebook Prophet, XGBoost, and LSTM and created two ensemble frameworks (simple average and stacking) to merge their strengths. I also enhanced the forecasting models with pertinent exogenous factors, specifically weather conditions (temperature, rainfall, humidity) and policy events (COVID-19 lockdowns, export restrictions, etc.), which greatly affect agricultural prices. I assessed the performance of each model using MAE and RMSE measures and compared them exhaustively by means of strict time-series cross-validation on data from 2015 to 2024. Among the main results of my work are:  
For every market, the stacking ensemble was the top-performing model. Demonstrating the power of ensemble learning in capturing the complex dynamics of tomato prices, it produced the lowest errors up to 15% lower RMSE than the best individual model. The simpler averaging ensemble also beat all individual models, emphasizing that even naive combination adds worth.  
Amongst single models, the LSTM neural network consistently offered the most precise predictions, closely followed by XGBoost. These models outperformed conventional techniques in handling interaction effects and non-linearity in the data. While ARIMAX and SARIMAX lagged in performance, particularly in turbulent times, Prophet had fair accuracy.  
Including exogenous variables policy dummies and weather greatly increased forecast accuracy. Models that included these variables like ARIMAX vs a plain ARIMA, or XGBoost with features vs without showed 5–10% error reductions. Especially with these signals in place, the models were able to predict the price increases during the COVID-19 lockdown and other supply shocks far more effectively. Proving to be strong, the forecasting system was confirmed over several years and market circumstances. The ensemble method especially offered consistency, producing consistent gains in every test period.  
My method is new in its breadth and integration compared to current studies: covering several cities, several model kinds, exogenous integration, and ensemble learning all in one work. It supports new evidence that such mixing, with appropriate tuning and cross-validation, can significantly outperform traditional projections in the framework of agricultural markets and it fits with recent high-impact studies promoting blending statistical and artificial intelligence techniques.  
Ultimately, my research shows that in agriculture a hybrid modelling approach driven by both domain knowledge (seasonality, weather effects) and data-driven algorithms can offer better price forecasts. This has clear consequences for enhancing world sustainable food security: more accurate projections enable farmers to make better cropping and marketing choices, governments to actively control food supply and inflation, and supply chain middlemen to cut waste and inefficiencies. Essentially, my work demonstrates how creative data analytics can lighten the future road for food security by transforming plentiful historical data and contemporary algorithms into actionable market intelligence.

**Recommendations**

Based on the findings of this study, I provide the following suggestions for future research as well as for practitioners in agricultural price forecasting:

Put ensemble‐based systems into use: Agencies market boards, agri-tech companies, Ministry of Agriculture should go beyond single-model extrapolations. My experience indicates that a Ridge-stacked ensemble of ARIMAX, SARIMAX and a tree-based learner (HGBR) can reduce forecasting errors by 30–100% compared with individual methods and even achieve near-perfect 32-week fits in three of four metros. Stakeholders can follow this plan: teach every component weekly and then merge using a simple linear meta-learner. Modern CPUs make monthly (or whenever new data arrive) retraining computationally trivial; forecasts can be delivered to farmers and traders in real time using a straightforward web dashboard or mobile app.

* Include external data sources: Consistently increasing accuracy by 8–15 % by including weather normals, short-term rainfall and temperature forecasts, and policy/event flags (e.g. lockdowns, export bans). I advise constructing automated ETL pipelines that consume policy-announcement feeds and meteorological APIs, match them to weekly horizons, and add them as regressors. Training courses for regional analysts should address missing exogenous observations management and feature-engineering best practices.
* Include more markets and commodities: Although tomatoes were the case study, the same approach works for onions, potatoes, pulses, or grains. Future initiatives should build a common stacking framework and commodity-specific regressors (soil moisture, international price indices) for each "basket" forecasting system. Cross-country adaptations calibrating seasonal periods and policy dummies locally could help an international market intelligence system.
* Promote joint modelling and probabilistic advancement: While point forecasts are useful, risk management increasingly depends on prediction intervals or complete predictive distributions. To measure the uncertainty, researchers could stack Monte-Carlo simulations on top of the ensemble or investigate Bayesian neural networks and quantile regressions. Similarly, vector autoregression or graph neural networks can be used to model multi-market dependence, therefore enabling shocks in one area to guide projections in another.
* Include alternative signals and high-frequency ones: Additional exogenous inputs to tighten short-horizon forecasts could be daily arrivals data, satellite vegetation indices, logistics-tracking telematics, or even web-search trends. Real-time weather alerts and IoT-enabled supply-chain sensors hold out more benefits in seizing unexpected supply shocks.
* Improve capacity building and understanding: Though they may seem opaque, ensembles are the best performers. Applying SHAP or permutation‐importance studies to the meta-learner weights can show which base model dominates under various regimes information that can direct trading strategies or policy. Training seminars for government analysts on these explainability tools and institutionalizing ensemble forecasts inside policy units would help to empower proactive, data-driven choices.

Stakeholders can turn weekly commodity forecasts into a high-precision early-warning system by adopting these suggestions ensemble adoption, strong data pipelines, probabilistic extensions, spatial integration, and interpretability. Such a tool will enable traders, farmers, and legislators to maximize storage and planting choices, stabilize farm earnings, and reduce consumer price fluctuation, therefore promoting sustainable food security.

**References**

1. Ahmed, N. K., Atiya, A. F., El Gayar, N., & El‑Shishiny, H. **(2010).** An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews, 29*(5–6), 594–621. https://doi.org/10.1080/07474938.2010.481556
2. Chen, Z., Goh, H. S., Sin, K. L., Lim, K., Chung, N. K. H., & Liew, X. Y. **(2021).** Automated agriculture commodity price prediction system with machine learning techniques. *Advances in Science, Technology and Engineering Systems Journal, 6*(4), 376–384. https://doi.org/10.25046/aj060442
3. Narayanan, S., & Saha, S. **(2021).** Urban food markets and the COVID‑19 lockdown in India. *Global Food Security, 29*, Article 100515. <https://doi.org/10.1016/j.gfs.2021.100515>
4. Nayak, G. H. H., Alam, M. W., Singh, K. N., Avinash, G., Kumar, R. R., Ray, M., & Deb, C. K. **(2024).** Exogenous variable driven deep learning models for improved price forecasting of TOP crops in India. *Scientific Reports, 14*, 17203. [https://doi.org/10.1038/s41598‑024‑68040‑3](https://doi.org/10.1038/s41598024680403)
5. Athiyyah, Q., Abd Rahman, M. A., & Fairuz, S. **(2024).** Ensemble ARIMA‑LSTM algorithm in predicting agriculture commodities market price for farmer’s education. In *Proceedings of the 7th International Conference on Information Technology (InforINO 2024)* (pp. 1–6). IEEE. <https://doi.org/10.1109/Inforino60363.2024.10551892>
6. Tomato price data: sourced from Agmarknet (<https://agmarknet.gov.in>).
7. Rainfall & temperature normals: sourced from India Meteorological Department (<https://imd.gov.in>).
8. Code & full analysis scripts: available at <https://github.com/PushkarPY123/tomato-dashboard>

**Appendix**

**Hyperparameters & Environment**

**1. ARIMAX**

* *Order grid*: p ∈ {0,1,2}, d ∈ {0,1}, q ∈ {0,1,2} -> **Chosen:** (2,1,2)
* *Seasonal order grid*: P ∈ {0,1}, D ∈ {0,1}, Q ∈ {0,1}, s = 52 -> **Chosen:** (1,1,1,52)

**2. HistGradientBoostingRegressor (HGBR)**

* max\_iter ∈ {50, 100, 200} -> **Chosen:** 100
* learning\_rate ∈ {0.01, 0.1, 0.2} -> **Chosen:** 0.1

**3. Stacking Ensemble (RidgeCV)**

* alphas ∈ {0.01, 0.1, 1.0, 10} used in 5-fold CV to learn weights

**4. Python environment**

* Python 3.9.7
* pandas 1.5.0
* numpy 1.24.0
* statsmodels 0.13.2
* scikit-learn1.1.2
* matplotlib 3.5.2