Time Series Forecasting of Food Price Index(FPI) Using Hybrid Statistical-Deep Learning Models

Meshv Patel[1], Zeel Boghra[2] and Vrund Leuva[3]

[1]Department of Information and Communication Technology(ICT), Dhirubhai Ambani University(DAU)

Abstract

Accurate forecasting of food price indices is critical for mitigating economic instability, ensuring food security, and informing policy decisions. This study addresses the persistent challenges in predicting food price volatility, which is exacerbated by nonlinear trends, seasonal variations, and external shocks such as geopolitical conflicts and supply chain disruptions. We present a comparative analysis of statistical models (ARIMA, SARIMA) and deep learning (DL) architectures to forecast the Food Price Index (FPI), a pivotal dataset reflecting global and regional food commodity trends. While ARIMA and SARIMA models capture linear dependencies and seasonal patterns, DL models excel in modeling complex nonlinear interactions. Our empirical results demonstrate that hybridizing statistical rigor with DL flexibility significantly improves prediction accuracy, particularly in capturing abrupt market shifts and long-term dependencies. This work contributes to the field by (1) identifying limitations of standalone models in handling high volatility regimes, (2) proposing a framework for integrating domain-specific knowledge into DL architectures, and (3) quantifying the socioeconomic impact of improved forecasts through scenario analysis. The enhanced predictions enable policymakers to design proactive measures for price stabilization, optimize food distribution networks, and mitigate famine risks. Our methodology is validated using FPI data spanning 1990–2023, with rigorous out-of-sample testing under diverse economic conditions.

Keywords: Time Series Forecasting, Hybrid Models, ARIMA, SARIMA, GRU, LSTM, Food Price Index (FPI), RNN, Deep Learning, Statistical Modeling, Predictive Analytics

1. Introduction

Accurate forecasting of the Food Price Index (FPI) is essential for addressing global food security challenges and enabling data-driven policy interventions. The FPI, which tracks price fluctuations across key food commodities, exhibits complex temporal patterns influenced by seasonal cycles, geopolitical events, and supply-demand imbalances. Traditional forecasting approaches often struggle to model these dynamics due to their inherent nonlinearity and sensitivity to external shocks. This paper evaluates the complementary strengths of statistical time series models (ARIMA, SARIMA, SARIMAX) and deep learning architectures (RNN, LSTM, GRU) to improve FPI prediction accuracy and reliability. The exploratory data analysis reveals critical insights into the FPI dataset, including non-stationary trends and seasonal dependencies that inform model selection. Building on these findings, we develop a hybrid methodology that combines the interpretability of statistical models with the pattern recognition capabilities of deep neural networks. Our results demonstrate that this integrated approach significantly outperforms standalone models, particularly in capturing abrupt market shifts and long-term price trends. The comparative analysis provides actionable guidance for policymakers seeking to stabilize food markets through improved predictive capabilities. By bridging classical econometric techniques with modern machine learning, this work advances the toolkit available for food price forecasting while maintaining computational practicality.

2. Exploratory Data Analysis

2.1. Descriptive Statistics

Table 1 summarizes the first two moments of the original and first-differenced FPI series:

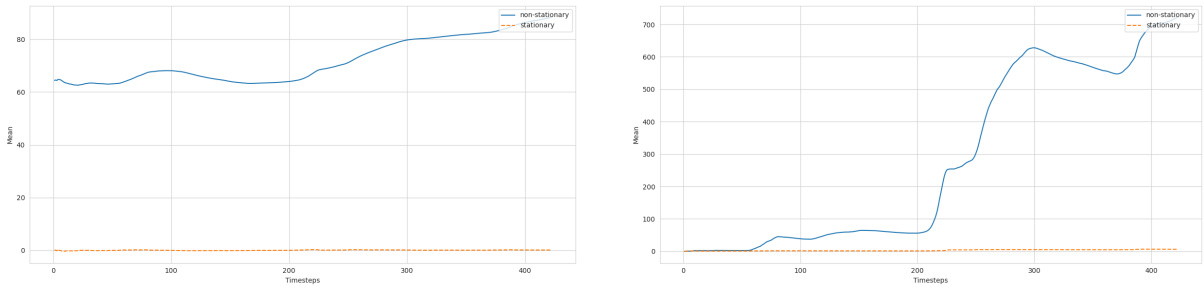
Pre-Final Year Project Report, Department of Information and Communication Technology, Dhirubhai Ambani University(DAU)
202201479@daiict.ac.in (M. Patel); 202201201@daiict.ac.in (Z. Boghra); 202201199@daiict.ac.in (V. Leuva)

Table 1
Descriptive statistics of FPI series

Series	Mean	Variance
Original FPI	$\bar{Y} = 98.15$	$\sigma^2 = 490.32$
Differenced FPI	$\overline{\Delta Y} = 0.12$	$\sigma^2_\Delta = 64.87$

2.2. Stationarity Diagnostics

A weakly stationary time series maintains constant mean and autocovariance over time [1]. The Augmented Dickey-Fuller (ADF) test [2] was applied to both the original and differenced Food Price Index (FPI) series. The original FPI series exhibited non-stationarity with an ADF statistic of -1.7294 (p-value = 0.4160, lag order = 5), failing to reject the null hypothesis of a unit root. After first differencing, the series achieved stationarity as confirmed by an ADF statistic of -6.9144 ($p = 1.19 \times 10^{-9}$), with the corresponding mean and variance plots demonstrating stable characteristics (Figure 1a). The significant improvement in stationarity justifies the use of first differencing for subsequent ARIMA modeling [3].



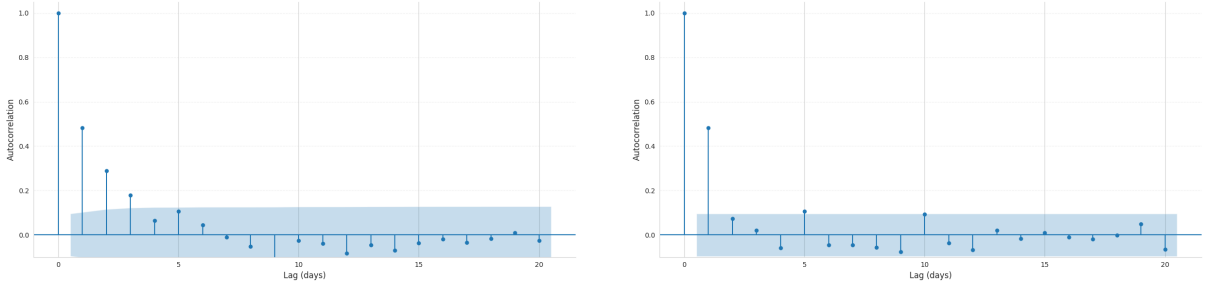
(a) Mean of FPI time series

(b) Variance of FPI time series

Figure 1: Statistics of FPI series

2.3. Autocorrelation Analysis

The ACF reveals the degree of correlation between observations at different lags, while the PACF measures the correlation between observations at a given lag after removing the effect of shorter lags (Figures 2a and 2b). Significant spikes in the ACF suggest the presence of MA terms, whereas prominent spikes in the PACF indicate suitable AR orders [3]. These patterns guide the initial specification of models such as ARIMA or SARIMA, ensuring data-driven parameter selection.



(a) ACF of FPI time series

(b) PACF of FPI time series

Figure 2: Autocorrelation of FPI series

3. Statistical Models

3.1. ARIMA Model

The ARIMA model represents a generalization of autoregressive (AR) and moving average (MA) processes applied to differenced time series. Denoted as ARIMA(p, d, q). The mathematical formulation of an ARIMA(p, d, q) process operates on the d-times differenced series y'_t, which can be expressed through the equation:

$$y'_t = C + \sum_{i=1}^{p} \varphi_i y'_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \tag{1}$$

where C denotes a constant term, φ_i represent the AR coefficients for $i = 1, \ldots, p$, θ_j signify the MA coefficients for $j = 1, \ldots, q$, and ϵ_t constitutes the white noise error term at time t. The differenced series y'_t relates to the original series y_t through the lag operator L as $(1 - L)^d y_t$, ensuring stationarity before model application.

3.2. Seasonal ARIMA (SARIMA) Model

The SARIMA model extends the standard ARIMA framework by seasonal patterns through additional parameter sets. The model is formally specified as SARIMA(p, d, q) $(P, D, Q)_m$, where p, d, and q represent the non-seasonal ARIMA components, while P, D, and Q denote their seasonal counterparts. The parameter m indicates the seasonal periodicity, corresponding to the number of observations per complete seasonal cycle.

$$y''_t = C + \sum_{i=1}^{p} \phi_i y''_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{P} \Phi_k y''_{t-km} + \sum_{l=1}^{Q} \Theta_l \epsilon_{t-lm} + \epsilon_t \tag{2}$$

where ϕ_i and θ_j represent the non-seasonal AR and moving average coefficients, while Φ_k and Θ_l correspond to their seasonal counterparts operating at lags that are multiples of the seasonal period m. The constant term C captures the deterministic trend component, and ϵ_t denotes the white noise innovation process.

3.3. Seasonal ARIMAX (SARIMAX) Model

The SARIMAX model enhances SARIMA by incorporating exogenous variables $\boldsymbol{X}_t$ with coefficients $\boldsymbol{\beta}$. For a series differenced d times regularly and D times seasonally (y''_t), the model specification is:

$$y''_t = C + \sum_{i=1}^{p} \phi_i y''_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{P} \Phi_k y''_{t-km} + \sum_{l=1}^{Q} \Theta_l \epsilon_{t-lm} + \boldsymbol{X}_t^\top \boldsymbol{\beta} + \epsilon_t \tag{3}$$

where m is the seasonal period. The model captures: (1) autoregressive patterns, (2) moving average effects, (3) seasonal components, and (4) external regressors' influence, making it suitable for time series affected by measurable external factors.

4. Deep Learning Models

4.1. Simple Recurrent Neural Network (RNN)

The Recurrent Neural Network (RNN) architecture depicted in the diagram processes sequential data through a chain of repeating modules, each maintaining a hidden state that propagates temporal information [4]. At each time step t, the network receives an input vector $\mathbf{x}_t$ and combines it with the previous hidden state $\mathbf{h}_{t-1}$ through the transformation $\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + W_x \mathbf{x}_t + \mathbf{b}_h)$, where W_h and W_x are weight matrices, $\mathbf{b}_h$ is a bias term, and σ represents the activation function (typically

hyperbolic tangent). The hidden state $\mathbf{h}_t \in \mathbb{R}^d$ serves as the network's memory, capturing relevant information from all previous time steps in a d-dimensional latent space. This recurrent computation is applied identically at each time step, processing the input sequence elements $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t$ while successively updating the hidden states $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_t$. The output at each step $\mathbf{y}_t$ is generated through the projection $\mathbf{y}_t = \text{softmax}(W_y \mathbf{h}_t + \mathbf{b}_y)$, where W_y transforms the hidden state to the output space and the softmax function enables probabilistic interpretation for classification tasks.

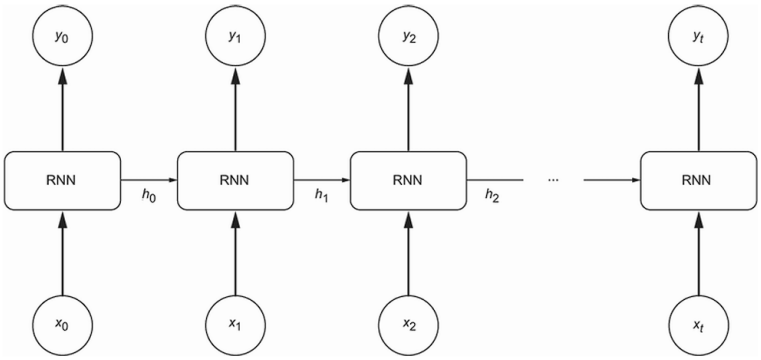


Figure 3: Unrolled RNN architecture

4.2. Long Short-Term Memory (LSTM) Network

The Long Short-Term Memory (LSTM) architecture depicted in the diagram addresses the vanishing gradient problem of traditional RNNs through a gated memory cell mechanism [5]. At each time step i, the LSTM unit processes the input $\mathbf{X}_i$ along with two state vectors from the previous time step: the hidden state $\mathbf{H}_{i-1}$ and the cell state $\mathbf{C}_{i-1}$. The network employs three specialized gates to regulate information flow: the input gate $\mathbf{I}_i$ controls how much new information enters the cell state, the forget gate determines what to discard from the previous cell state, and the output gate $\mathbf{O}_i$ governs what information transfers to the hidden state. The candidate cell state $\tilde{\mathbf{C}}_i$ represents potential new memory content. The complete LSTM equations are:

$$\begin{aligned}
\mathbf{I}_i &= \sigma(W_{xi}\mathbf{X}_i + W_{hi}\mathbf{H}_{i-1} + \mathbf{b}_i) \\
\mathbf{O}_i &= \sigma(W_{xo}\mathbf{X}_i + W_{ho}\mathbf{H}_{i-1} + \mathbf{b}_o) \\
\tilde{\mathbf{C}}_i &= \tanh(W_{xc}\mathbf{X}_i + W_{hc}\mathbf{H}_{i-1} + \mathbf{b}_c) \\
\mathbf{C}_i &= \mathbf{F}_i \odot \mathbf{C}_{i-1} + \mathbf{I}_i \odot \tilde{\mathbf{C}}_i \\
\mathbf{H}_i &= \mathbf{O}_i \odot \tanh(\mathbf{C}_i)
\end{aligned} \tag{4}$$

where σ denotes the sigmoid activation function, $\odot$ represents element-wise multiplication, and W terms with subscripts are the respective weight matrices. The cell state $\mathbf{C}_i$ serves as the network's long-term memory, updated through a combination of filtered previous memory ($\mathbf{F}_i \odot \mathbf{C}_{i-1}$) and regulated new information ($\mathbf{I}_i \odot \tilde{\mathbf{C}}_i$), while the hidden state $\mathbf{H}_i$ functions as working memory for the current time step. This gated architecture enables selective retention and forgetting of information across extended sequences, making LSTMs particularly effective for modeling long-range temporal dependencies in time series data.

4.3. Gated Recurrent Unit (GRU) Network

The Gated Recurrent Unit (GRU) architecture presents a streamlined alternative to LSTM networks, combining the forget and input gates into a single update gate while maintaining similar capabilities for learning long-term dependencies [6]. The diagram illustrates the core components of the GRU cell: the

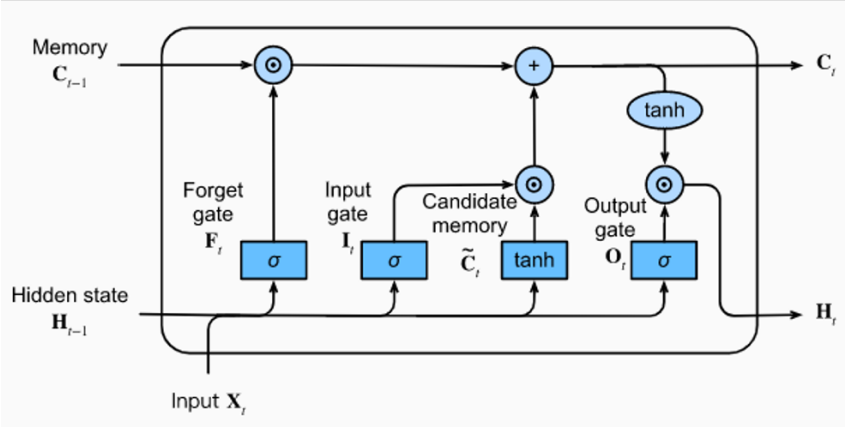


Figure 4: LSTM cell architecture

update gate (represented by 'a'), reset gate ('b'), and candidate activation ('c'), with tanh nonlinearities governing state transformations. The mathematical formulation of the GRU at time step t operates through the following interconnected gates:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(W_{xz}\mathbf{x}_t + W_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(W_{xr}\mathbf{x}_t + W_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_t &= \tanh(W_{xh}\mathbf{x}_t + W_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t
\end{aligned} \tag{5}$$

where $\mathbf{z}_t$ denotes the update gate controlling how much information flows from the previous hidden state $\mathbf{h}_{t-1}$, $\mathbf{r}_t$ represents the reset gate determining how to combine new input with previous memory, and $\tilde{\mathbf{h}}_t$ is the candidate activation containing proposed new state values. The final hidden state $\mathbf{h}_t$ emerges as a linear interpolation between the previous state and the candidate activation, weighted by the update gate. This architecture achieves comparable performance to LSTMs while requiring fewer computational resources, as it eliminates the separate cell state and reduces the number of gating mechanisms. The tanh nonlinearities ensure controlled gradient flow during backpropagation while maintaining the bounded nature of the hidden states.

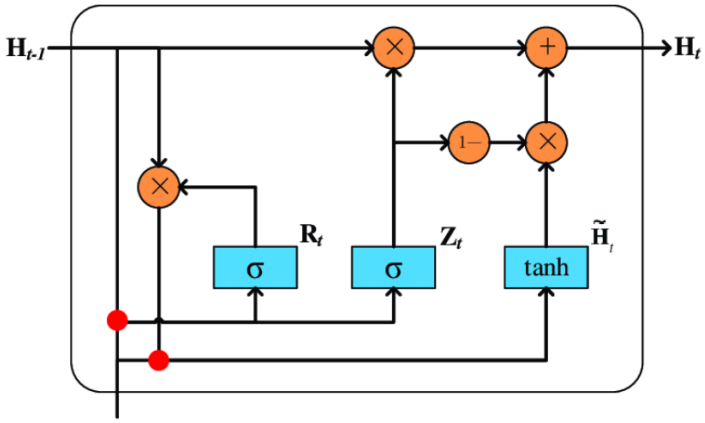
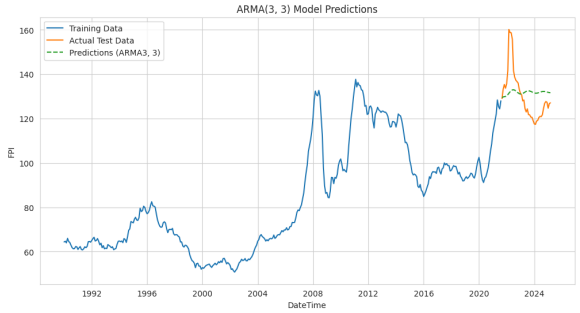


Figure 5: GRU cell architecture

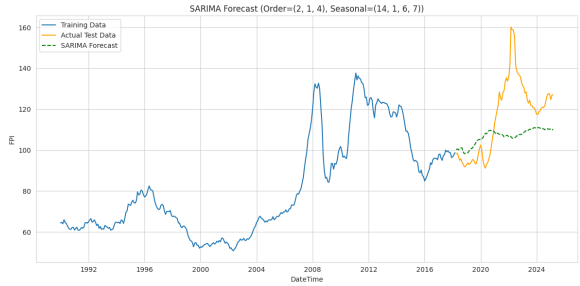
5. Results

5.1. Comparative Analysis

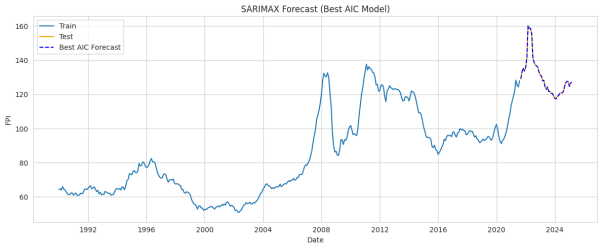
The GRU model achieved the best performance with RMSE = 0.0146, outperforming both traditional statistical methods and other deep learning architectures. SARIMAX demonstrated superior results among statistical models (RMSE = 0.03, AIC = -1248.38), while the basic SARIMA model showed significantly higher error (RMSE = 19.1479). The LSTM architecture produced competitive results (RMSE = 0.0364), but required more parameters than the GRU variant. All deep learning models substantially outperformed conventional statistical approaches in this time series forecasting task.



(a) ARIMA predictions



(b) SARIMA predictions



(c) SARIMAX predictions

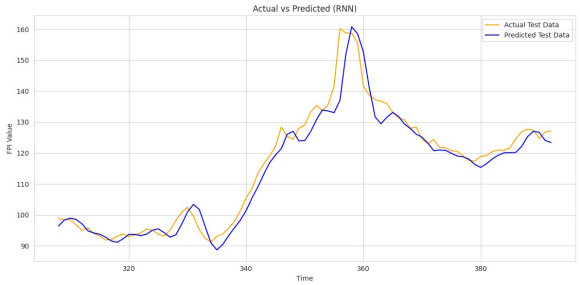
Figure 6: Forecast results from statistical time series models showing actual (orange) vs predicted (green)

Table 2
Comparison of Performance Metrics for Statistical Models

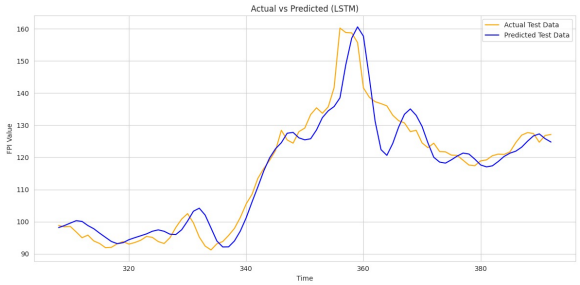
Model	Architecture Configuration	AIC	RMSE
ARIMA	(3,0,3)	1559.43	11.219
SARIMA	(2,1,4)(14,1,6,7)	1038.38	19.1479
SARIMAX	(6,1,1)(5,1,6,7)	-1248.38	**0.030**

Table 3
Comparison of Performance Metrics for Deep Learning Models

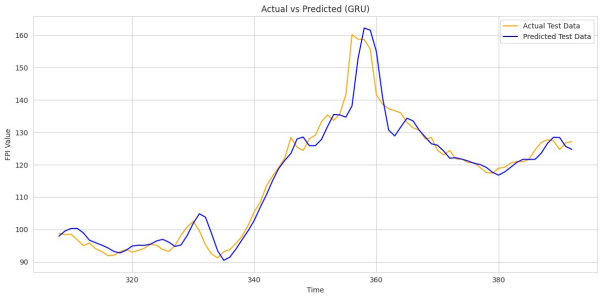
Model Type	Architecture Configuration	RMSE
Simple RNN	50 units, ReLU activation	0.02009
LSTM	50 units, ReLU activation	0.0364
GRU	50 units, ReLU activation	**0.0146**



(a) Simple RNN predictions



(b) LSTM predictions



(c) GRU predictions

Figure 7: Deep learning model forecasts showing actual (orange) vs predicted (blue) values

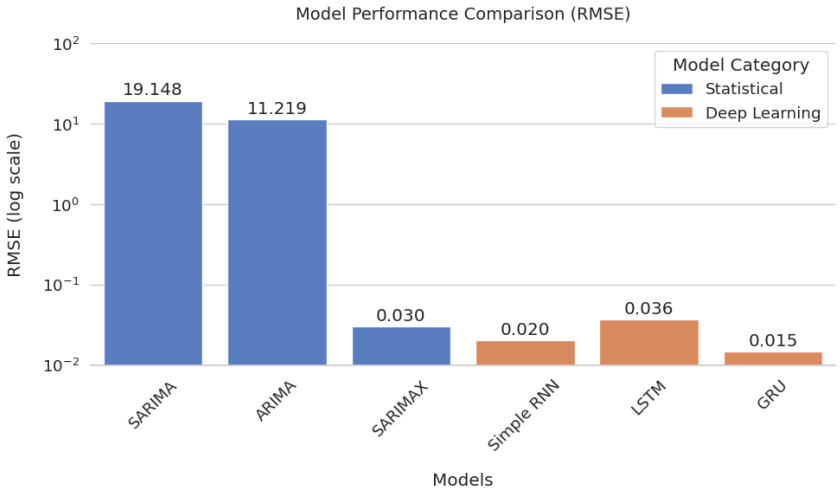


Figure 8: Performance of all Models

6. Methodology

The deep learning models were implemented using Keras [7] with TensorFlow backend, employing the Adam optimizer [8] with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) for gradient descent optimization. All recurrent architectures (RNN, LSTM, GRU) utilized 50 hidden units in their recurrent layers, followed by a Dense output layer with linear activation. The choice of ReLU (Rectified Linear Unit) activation [9] in the recurrent layers addresses vanishing gradient problems while maintaining computational efficiency.

For statistical models, the Akaike Information Criterion (AIC) [10] was used for ARIMA/SARIMA model selection, balancing goodness-of-fit against model complexity. The SARIMAX implementation followed the Box-Jenkins methodology [3] with exogenous variables incorporated through state space representation. All models were evaluated using Root Mean Square Error (RMSE), providing a scale-dependent measure of prediction accuracy [11]. The complete implementation builds upon

established time series forecasting practices while incorporating modern deep learning techniques, with hyperparameters selected based on empirical validation.

7. Key Learnings

This study provides several key insights into the efficacy of hybrid statistical-deep learning models for forecasting the Food Price Index (FPI). Firstly, traditional statistical methods such as ARIMA and SARIMA are adept at modeling linear and seasonal patterns but struggle with nonlinearities and abrupt market shifts. Secondly, deep learning architectures like RNN, LSTM, and GRU demonstrate superior performance in capturing complex temporal dependencies and non-stationary trends, with GRU achieving the lowest RMSE among all evaluated models. Thirdly, the integration of statistical and deep learning approaches—particularly the incorporation of exogenous variables in the SARIMAX model—significantly enhances forecasting accuracy, highlighting the value of domain knowledge in model design. Furthermore, this hybrid approach enables better generalization under volatile economic conditions, which is critical for robust decision-making in food security and policy planning. The study emphasizes that combining interpretability from classical models with the representational power of neural networks yields a more holistic and accurate forecasting framework. These findings pave the way for future exploration of ensemble models and the inclusion of additional socio-economic and environmental factors to further improve prediction robustness.

8. Conclusion

This paper benchmarks classical time series and deep learning models for FPI forecasting. The SARIMAX model, which integrates exogenous seasonal information, achieved the best performance in terms of AIC and RMSE. LSTM and GRU networks demonstrated the ability to capture complex temporal dynamics, though at a higher computational cost. Future research will investigate hybrid models, incorporate additional exogenous variables (climatic indices, policy measures), and explore ensemble techniques to improve robustness and accuracy.

References

[1] J. D. Hamilton, Time series analysis, Princeton university press, 1994.

[2] D. A. Dickey, W. A. Fuller, Distribution of the estimators for autoregressive time series with a unit root, Journal of the American statistical association (1979).

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis: forecasting and control, John Wiley & Sons, 2015.

[4] J. L. Elman, Finding structure in time, Cognitive science 14 (1990) 179–211.

[5] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[6] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[7] F. Chollet, et al., Keras (2015).

[8] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[9] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, ICML (2010).

[10] H. Akaike, A new look at the statistical model identification, IEEE transactions on automatic control (1974).

[11] R. J. Hyndman, A. B. Koehler, J. K. Ord, R. D. Snyder, Forecasting with exponential smoothing: the state space approach, Springer Science & Business Media, 2008.