USD/INR Forex Rate Prediction

Group No: 04

Harsh	Harsh Choudhary	Ayushman Baghel	Avinash Choudhary
2103116	2103117	2103305	2106311

November 23, 2024

Abstract

This report presents a comparative analysis of multiple predictive models for time series forecasting, specifically applied to a 30-day forecast task of USD/INR rate. The models assessed include ARIMA, SARIMA, LSTM, and XGBoost, as well as an ensemble approach combining all three models with weighted contributions of 0.7 for ARIMA(2,1,2), 0.2 for LSTM, and 0.1 for XGBoost. Performance metrics such as RMSE and MAE were used to compare individual model accuracy, and information criterion like AIC and SBC were used to choose parsimonious models. Statistical tests, such as the ADF for stationarity, the ARCH-LM test for heteroskedasticity, and GARCH modeling for volatility, were applied to enhance insights into the data's characteristics. Additionally, the binary segmentation method (binseg) detected structural breaks, which informed model adjustments. The ARIMA model showed robust performance with a test RMSE of 0.2049, while XGBoost yielded the lowest RMSE of 0.0841. Despite its relatively high complexity, the LSTM model demonstrated a test RMSE of 0.6486. The ensemble model, integrating strengths from each individual model, achieved a forecast RMSE of 0.1808, indicating that an ensemble approach can leverage diverse predictive capacities to enhance forecast accuracy. This study highlights the efficacy of ensemble methods in improving forecasting reliability, making it a valuable approach for complex time series predictions.

Keywords: ADF, AIC, ARIMA, Binseg, Ensemble, GARCH, LSTM, SARIMA, SBC, XGBoost

Introduction - Background & Motivation

In the globalized economy, exchange rate forecasting is a pivotal area of study with far-reaching implications for investment, economic stability, and policy-making. Exchange rates between major currencies, such as the INR and the USD, are influenced by complex economic forces, and their fluctuations impact both international trade and cross-border investments. For investors and policymakers alike, accurate predictions of USD/INR rates can provide crucial insights, aiding in decisions that minimize risk and optimize returns. The Indian economy, one of the world's fastest-growing, maintains strong trade and financial connections with the U.S., making the USD/INR rate a focal point for understanding economic interactions and potential volatility. Thus, predicting the behavior of this exchange rate is not only beneficial for financial decision-making but also essential for understanding macroeconomic trends.

Despite its importance, exchange rate prediction is challenging, as it involves complex data patterns influenced by nonlinear trends and structural breaks caused by sudden economic shifts. Traditional models, though effective in stable conditions, may struggle to capture the non-linear patterns and abrupt shifts present in exchange rate data. These limitations motivate the need for advanced techniques that can adapt to such complexities. This study specifically focuses on whether ensemble modeling—a method that combines multiple predictive models—can better account for these patterns compared to traditional approaches. Ensemble models have the potential to integrate strengths from various models, possibly improving prediction accuracy by leveraging both statistical and machine learning insights.

Though we demonstrate methods in pursuit of getting high accuracy, yet the goal of this paper is not to come up with the best model achieving the lowest RMSE but to demonstrate the essential steps in

conducting a thorough time series analysis and forecasting process. This involves assessing stationarity, identifying potential heteroskedasticity, and modeling volatility. Additionally, detecting structural breaks is an important step in adapting models to changing patterns in data. By systematically implementing these methods, this study serves as a guide to performing robust and insightful time series analysis, helping future researchers in selecting models that are not only accurate but also interpretable and aligned with underlying data characteristics. It may so happen that the performance quoted in 11 is not at par with other literatures like (Gupta, 2021) or (Darvas & Schepp, 2020), but our goal is not to fine tune the models according to each and every detail of the selected 30 day test set as it makes little sense to capture every bit of volatility in exchange rate market given the Efficient Market Hypothesis.

Literature Review

We studied and experimented with various techniques to improve the pre-processing of data and enhance the accuracy of our predictions. Relevant literature have been identified for each technique, providing foundational knowledge and examples of applications to exchange rate forecasting.

We begin with the study by (D et al., 2024). The objective of this paper was on predicting the exchange rate between the USD and the INR using SARIMA. The study analyzed weekly data from June 2013 to June 2023. By applying seasonal decomposition of the data, the authors found that the SARIMA model was suitable, with a final model specification of $SARIMA(2,1,0)(2,1,0)_{52}$, where 52 represents the seasonal periodicity of the weekly data.

Next we reviewed the work by (Bartaria, 2021). This study aimed at the prediction of the USD/INR exchange rate during the pandemic, using daily closing prices. It compared the forecasting performance of SARIMA, SARIMA-GARCH, and ARIMAX models, aiming to outperform the benchmark Random Walk model. The study ultimately selected the SARIMA-GARCH(1,1) model based on the AIC criterion and recommended excluding volatile exogenous variables, like the Nifty50 index, due to their lack of satisfactory performance in forecasting.

A tutorial by (Staudemeyer & Morris, 2019) offered a detailed explanation of LSTM networks, highlighting their ability to process up to `1000` time steps in contrast to the typical `10` time steps handled by traditional RNNs. The study only aims to explain how LSTMs use memory cells to maintain constant error flow, which is a key advantage in time series forecasting. The paper discussed a number of neural network architectures, from perceptron to Gated Units, and is a one stop guide towards the mathematics behind several deep network models.

The research by (Islam et al., 2021) was then reviewed. It investigates the use of XGBoost in forecasting the USD exchange rate against the Indonesian Rupiah. The study utilized daily data over a span of three years and employed a hyper-parameter tuning process focused on RMSE. One of the main benefits of XGBoost is its ability to combine multiple trees in an ensemble, known as Classification and Regression Trees (CARTs), to improve forecast accuracy. The paper highlighted the importance of feature selection using information gain to determine the most relevant variables. The set of parameters that served the best forecast were presented as the research results.

All the papers reviewed above have either used classical or the machine learning models individually, and have focused on increasing the quality of forecasts. We want to take the work further by forming a model which is a weighted average of the predictions of these individual models. The idea is simple - *Wisdom of crowd make better choices than intelligence of an individual.*

Research Questions

Our research question is:

"Can an ensemble approach combining traditional statistical methods and advanced machine learning models more accurately predict the USD/INR exchange rate, accounting for structural breaks and non-linear patterns in the data?"

In pursuit of this question, we will:

- Conduct volatility analysis to assess USD/INR rate fluctuations and offer confidence intervals.
- Detect structural breaks to make evident the need for non linear model.
- Develop an ensemble model for enhanced predictive performance.
- Compare model effectiveness between traditional statistical and machine learning approaches in exchange rate forecasting.

This project aims to identify whether a hybrid model framework can surpass individual methods, thus providing a more accurate understanding of exchange rate behavior.

Data and Methodology

In this section, we have provided an in-depth discussion of the variables under consideration, describing how they relate to the predictive modeling of exchange rates. This includes a detailed overview of both the data sources and characteristics, as well as the methodologies to be applied to analyze and interpret this data effectively.
The following two subsections provide further details on the data and methodology.

Data

The data is obtained from a single source (Investing.com, 2024)(see Table 1). We take a moment now to familiarize with the fields of the dataset, and their relevance (if any) in our analysis. Apart from the fields directly present in the dataset, we would also look upon some additional fields we need for some models like XGBoost that is formed from operations on existing fields.

Table 1: Description of Dataset Sources

Sl No	Alias	Description	Source
1	exchange_data	Dataset containing daily exchange rate price of INR for 1 USD	Investing.com (Investing.com, 2024)

The dataset *exchange_data* had been loaded as a pandas dataframe object, and it has fields namely `Date, Price, Open, High, Low, Vol, Change %`. The description of all the fields are presented here 2.

Table 2: Description of Main fields

Sl No	Field Name	Description	Type
1	Date	The date of the recorded exchange rate	Date (string)
2	Price	The closing exchange rate of USD/INR for the day	Float
3	Open	The opening exchange rate of USD/INR for the day	Float
4	High	The highest exchange rate recorded for USD/INR during the day	Float
5	Low	The lowest exchange rate recorded for USD/INR during the day	Float
6	Vol	Trading volume (if available) for the day	Float/NaN
7	Change %	The percentage change in the exchange rate from the previous day	Float

The auxiliary fields that will be needed by XGBoost are derived from the `Date` field. This is an attempt to capture different aspects of date, hoping to find some information gain. These fields are - day, dayofweek, dayofyear, week, month, year. Apart from these, XGBoost also uses some lagged values of the `Price, Open, High, Low, Change%` columns themselves, all described here 3.
Having seen the fields of the dataset, we are now ready to describe the variables relevant for each of the models. ARIMA model uses the `Date` field as index, and fits only on `Price` column, along with it's lagged values. SARIMA model does exactly same. However, LSTM uses the scaled values of `Price` field to converge faster. XGBoost uses several other auxiliary variables as well. The information of all the variables along with the models they are used in is presented here 4.
Each of the variables are the features on which respective models train. The selection of these features is done on the basis of trial and error (Crone & Kourentzes, 2010), and the table contains the ones we preserve in the final model.

Table 3: Description of Auxiliary fields

Sl No	Field Name	Description	Type
1	Day	The day of the month, ranging from 1 to 31.	Integer
2	DayOfWeek	The day of the week represented as an integer (0-6, where 0 is Monday).	Integer
3	DayOfYear	The day of the year, ranging from 1 to 365 (or 366 in leap years).	Integer
4	Week	The week number of the year.	Integer
5	Month	The month number, ranging from 1 to 12.	Integer
6	Year	The year when the exchange rate was recorded.	Integer
7	Price_Lag_i	The final exchange rate i days ago.	Float
8	Open_Lag_i	The opening exchange rate i days ago.	Float
9	High_Lag_i	The highest exchange rate i days ago.	Float
10	Low_Lag_i	The lowest exchange rate i days ago.	Float
11	Change_Lag_i	The change % i days ago.	Float

Table 4: Description of Variables

Sl No	Name	Description	Models
1	$PRICE$	The daily closing price of exchange rates	ADF, ARIMA, SARIMA, LSTM, XGBOOST
2	$\Delta PRICE$	Defined as $PRICE_i - PRICE_{i-1}$	ADF, ARIMA, SARIMA
3	$PRICE_{t-i}$	Defined as $PRICE$ i days ago.	ADF, ARIMA, SARIMA, XGBOOST
4	$\Delta PRICE_{t-i}$	Defined as $\Delta PRICE$ i days ago	ADF, ARIMA, SARIMA
5	$PRICE^S$	Defined as $PRICE$ scaled in range 0 to 1.	LSTM
6	DAY	Range is 1-31	XGBOOST
7	DAY^W	Day of week, range is 0-6	XGBOOST
8	DAY^Y	Day of year, range is 0-366	XGBOOST
9	$WEEK$	Range is 1-52	XGBOOST
10	$MONTH$	Range is 1-12	XGBOOST
11	$YEAR$	For our dataset, range is 2006-2024	XGBOOST
12	$OPEN$	The daily opening price of exchange rates	XGBOOST
13	$OPEN_{t-i}$	Defined as $OPEN$ i days ago	XGBOOST
14	$HIGH$	The daily highest price of exchange rates	XGBOOST
15	$HIGH_{t-i}$	Defined as $HIGH$ i days ago	XGBOOST
16	LOW	The daily lowest price of exchange rates	XGBOOST
17	LOW_{t-i}	Defined as LOW i days ago	XGBOOST
18	$CHANGE$	The daily change % of exchange rates	XGBOOST
19	$CHANGE_{t-i}$	Defined as $CHANGE$ i days ago	XGBOOST

Methods

The first step towards the forecasting task was to check for stationarity of the $PRICE$ series. We used ADF test to check for unit roots at 5% significance level. ADF test equation for $PRICE_t$ is given by (1).

$$\Delta PRICE_t = \alpha + \beta t + \gamma PRICE_{t-1} + \delta_1 \Delta PRICE_{t-1} + \delta_2 \Delta PRICE_{t-2} + \cdots + \delta_p \Delta PRICE_{t-p} + \varepsilon_t \quad (1)$$

where:

- $\Delta PRICE_t = PRICE_t - PRICE_{t-1}$ is the first difference of the series,
- α is a constant (drift term),
- βt represents a linear trend (if included),
- γ is the coefficient of $PRICE_{t-1}$, which is used to determine stationarity,
- δ_i are coefficients of lagged differences $\Delta PRICE_{t-i}$,
- p is the number of lagged difference terms, and
- ε_t is white noise.

The hypothesis test for stationarity is based on the value of γ:

- Null Hypothesis (H_0): $\gamma = 0$ (the series has a unit root, i.e., it is non-stationary).
- Alternative Hypothesis (H_1): $\gamma < 0$ (the series is stationary).

If the p-value from the test is below the chosen significance level (e.g., 0.05), we reject the null hypothesis and conclude that the series is likely stationary.

Next up utilized the ARIMA and SARIMA models. These models are particularly useful for time series forecasting, especially when data exhibits trends and seasonality, as they capture linear dependencies in historical data. The SARIMA model is specified in (2). If we make the SARIMA parameters P and Q equal to 0, we effectively derive the ARIMA model.

$$\Delta PRICE_t = \alpha + \sum_{i=1}^{p} \beta_i \Delta PRICE_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \sum_{k=1}^{P} \Phi_k \Delta PRICE_{t-kS} + \sum_{l=1}^{Q} \Theta_l \varepsilon_{t-lS} + \varepsilon_t \tag{2}$$

where:

- p and q are the orders of the non-seasonal autoregressive and moving average terms,
- β_i and θ_j are coefficients for the non-seasonal components,
- P and Q are the orders of the seasonal autoregressive and moving average terms,
- Φ_k and Θ_l capture seasonal effects,
- S is the seasonal period, for example: if the data is daily, and we suspect weekly seasonality then $S = 5$ or $S = 7$ (depending on business days or all days),
- ε_t is the error term at time t.
- $\Delta PRICE_t$ is assumed stationary for this equation, though tests are discussed in *"Results and Discussions"*.

Since we have a daily data, we can expect some conditional heteroskedasticity in the residuals of ARIMA/SARIMA. To assess the presence of ARCH effects in the residuals of the price series, we perform the ARCH LM (Sjölander, 2011) test (3). Detecting ARCH effects suggests that volatility is time-varying and can be modeled using a ARCH/GARCH model. The procedure for the ARCH LM test is as follows:

1. Fit a time series model to obtain residuals, denoted by ε_t.
2. Square the residuals, ε_t^2, to represent conditional variance.
3. Regress ε_t^2 on its p lagged values:
$$\varepsilon_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2 + \nu_t \tag{3}$$
4. Calculate the LM test statistic, which tests the null hypothesis:
$$H_0 : \alpha_1 = \alpha_2 = \cdots = \alpha_p = 0$$
against the alternative hypothesis that at least one of the α_i coefficients is non-zero.

If the test statistic is significant (typically at a 5% level), we reject H_0, indicating the presence of ARCH effects. To assess the volatility in exchange rates, we employ the GARCH model, which is effective for time series data with time-varying volatility. GARCH(1,1) specification is presented in (5).

$$\varepsilon_t = \nu_t \sqrt{h_t}, \quad \nu_t \sim \mathcal{N}(0,1) \tag{4}$$

$$h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 h_{t-1} \tag{5}$$

where:

- h_t is the conditional variance at time t,
- α_0, α_1, and β_1 are model parameters, and
- ν_t represents a standard gaussian white noise process.

To motivate the need for some advanced machine learning models, we conducted structural break analysis (Li et al., 2022). The Binseg algorithm (Borelli, 2024) operates iteratively by splitting the series

and minimizing the total cost across segments. The cost function for one segment can be specified as MSE, as in (6).

$$\text{Cost}(PRICE_{t_i+1:t_{i+1}}) = \sum_{j=t_i+1}^{t_{i+1}} (PRICE_j - \overline{PRICE}_{t_i+1:t_{i+1}})^2 \tag{6}$$

where:

- $PRICE_j$ is the observed value at time j,
- $\overline{PRICE}_{t_i+1:t_{i+1}}$ is the mean of the segment $PRICE_{t_i+1}$ to $PRICE_{t_{i+1}}$.

The overall cost function for Binary Segmentation with the MSE cost is then given by (7).

$$C(PRICE) = \sum_{i=0}^{m} \sum_{j=t_i+1}^{t_{i+1}} (PRICE_j - \overline{PRICE}_{t_i+1:t_{i+1}})^2 \tag{7}$$

where:

- m is the no. of breakpoints specified, creating m+1 segments.

We now take turn towards state of the art models like LSTM and XGBoost. An LSTM model is a type of RNN designed to learn from sequences of data, making it particularly useful for time-series forecasting and other tasks where the input data is sequential. The mathematical theory behind the working of LSTM is advanced, hence we do not include them here. Those interested may refer to (Appendix,**LSTM Model Theory**). Another popular choice of model is XGBoost, where the model learns a weighted sum of decision trees where each tree corrects the errors in gradients of the previous tree. Again, the theory behind working of XGBoost is quite overwhelming, so those interested can refer to (Appendix,**XGBoost Model Theory**).
Equipped with the basic idea of what equations we have estimated, and it is time to present the implementation details and the inferences from the results.

Results and Discussion

We started the analysis by visualizing the data. The plot (Perktold et al., 2024a) of the exchange rate prices is given in 1. The dataset is clearly trending upwards, so we suspected the presence of unit roots. We plotted the ACF, which was very slowly decaying. The results from ADF test (Perktold et al., 2024b) for *PRICE* and $\Delta PRICE$ series is summarized in 5.

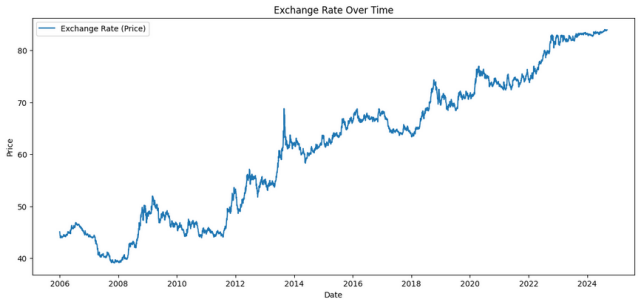

Figure 1: Exchange Rate Over Time

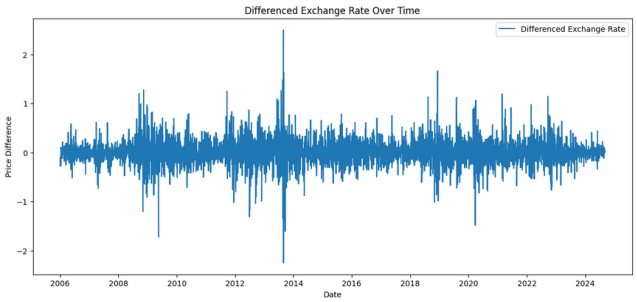

Figure 2: Differenced Exchange Rate Over Time

Both the de-trended as well as differenced series emerged stationary, however, we plotted the ACF of both of these to choose difference stationarity as the correct model [See 3 4]. We also plotted the differenced price series itself [See 2].
Equipped with the results of stationarity, we implemented the ARIMA and SARIMA models. The dataset was split into train and a 30 day test set. The best ARIMA/SARIMA models were obtained by using `auto_arima` function of `pmdarima` library (Alkaline-ML, 2024). This uses an iterative search process over a specified range of ARIMA models, guided by a information criterion (AIC or BIC). It also backtracks to avoid local minima. The results for the best ARIMA and best SARIMA models are presented in [6]. For ARIMA part, we accessed the models using both AIC and BIC, but following

Table 5: ADF Test Results for Original, Differenced, Deseasonalized and Detrened Price Series

	Price Series	Differenced Series	Deseasonalised Series (m=5)	Detrended Series
ADF Statistic	-0.1993	-29.6073	-0.1993	-3.2530
p-value	0.9386	0.0	0.9386	0.0171
Stationarity	Not stationary	Stationary	Not stationary	Stationary
Critical Value (1%)	-3.4317			
Critical Value (5%)	-2.8621			
Critical Value (10%)	-2.5671			

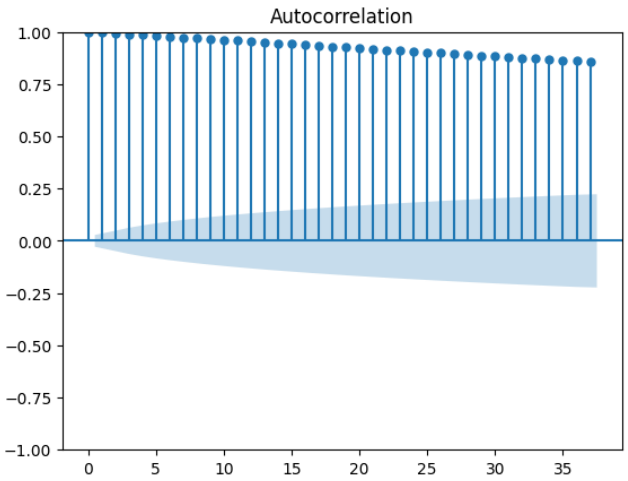

Figure 3: ACF of detrended series

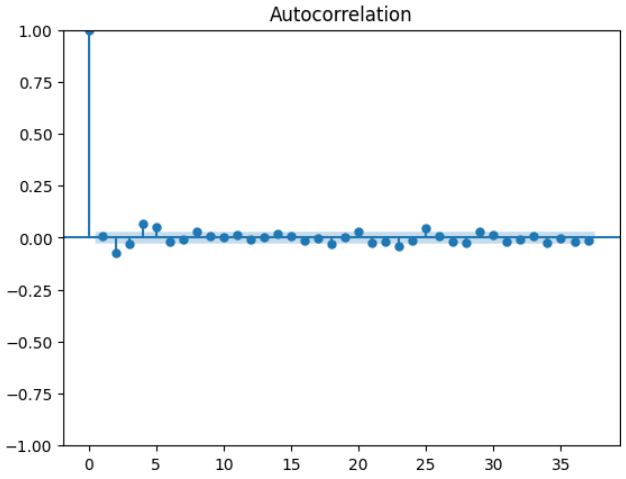

Figure 4: ACF of differenced series

the principle of parsimony, BIC model was selected. SARIMA assumed weekly seasonality, as we are driven by the assumption that the market starts afresh each week. The performances of $ARIMA(2,1,2)$ model was close to the $SARIMA(2,1,2)(1,0,1)_5$ model, and so we selected simpler $ARIMA(2,1,2)$ itself for further GARCH error analysis.

The next step was to conduct ARCH test, to check for conditional Heteroskedasticity. The results of ARCH-LM test are shown in 7. The significant values of α_1 and β_1 indicated that volatility clusters are present in the series: periods of high volatility tend to follow each other, as do periods of low volatility. Since β_1 is close to 1, the model implied persistent volatility, suggesting that large shocks in the series had long-lasting effects. Also, low value of μ suggested that the mean of the error ε_t is 0.

Thus, we aimed to improve the forecast by reporting confidence intervals along with mean forecasting for future values. To achieve this, we fitted the $GARCH(1,1)$ model on the residuals of $ARIMA(2,1,2)$ and obtained the forecast of variance (conditional on train set) for the 30 day test set values. The plot is shown by 5. The shaded regions (in light green tint) represent the confidence intervals from $GARCH(1,1)$ analysis. From the plot, results were good on the part that the confidence intervals account for most of the forecast errors we are committing, but the forecasted values themselves are not at par with the quality that some other models easily achieve. In order to analyze why this might be happening, we conducted a structural break test using Binary Segmentation. The no. of breakpoints have to be passed as hyperparameter, and visual inspection of the plot hinted us to keep 11 break points, based on the changes in mean. The breakpoints are marked in red in 6.

The structural breaks in the time series data offered clear indication as to why the time series model were underperforming. It is this point that LSTM became useful. An LSTM network was created using `keras` library integrated into `tensorflow` (Chollet, 2020). Since the LSTM cells work for pre-specified time steps, a test set of only 30 observations are insufficient when we predict from the model, so we do a standard 80-20 train-test split, and used a time step of 60 for training the network. The model specification and performance can be found here 8. Hyperparameter tuning was done following the advice from (Siddhartha, 2020).

Table 6: Comparison of SARIMAX Models

Metric	**ARIMA(2, 1, 2)**	**SARIMAX(2, 1, 2)x(1, 0, [1], 5)**
Log Likelihood	-194.234	-183.839
AIC	398.468	381.679
BIC	430.925	427.119
Ljung-Box (L1) (Q)	0.02	0.12
Prob(Q)	0.90	0.72
Heteroskedasticity (H)	1.05	1.05
Prob(H) (two-sided)	0.35	0.30
Skew	0.29	0.22
Kurtosis	10.76	10.60
Coefficients		
ar.L1	0.2534 (0.051)*	0.0944 (0.052)*
ar.L2	-0.6320 (0.059)*	-0.6885 (0.052)*
ma.L1	-0.2465 (0.055)*	-0.0837 (0.055)*
ma.L2	0.5510 (0.064)*	0.6164 (0.056)*
sigma2	0.0634 (0.001)*	0.0630 (0.001)*
ar.S.L5	-	0.9171 (0.036)*
ma.S.L5	-	-0.8916 (0.041)*
Forecast Errors		
RMSE	0.2049	0.21490
MAE	0.1451	0.1499

*standard errors in parantheses.

Table 7: Summary of ARCH LM test Results and GARCH(1,1) model

Parameter	**Coefficient**	**p-value**
ARCH-LM test		8.21×10^{-176}
Mean Model		
μ	0.0027	0.316
Volatility Model		
ω	0.0027	5.87×10^{-4}
α_1	0.1753	1.59×10^{-16}
β_1	0.7941	2.74×10^{-201}

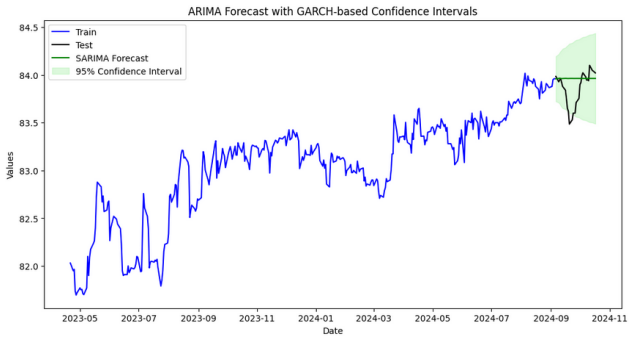

Figure 5: 30 day ahead Forecast from ARIMA(2,1,2)-GARCH(1,1) model

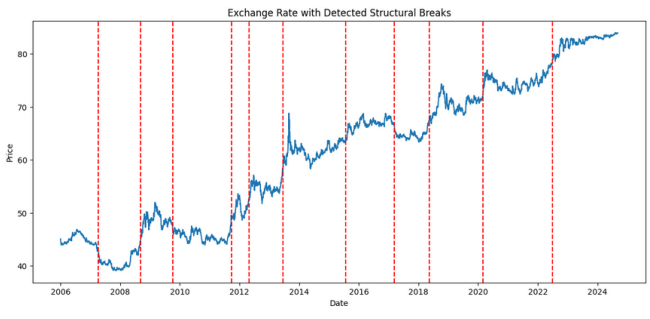

Figure 6: Exchange Rate Data with structural breaks

The model tries to generalize well. The test errors are high, but the training error is not too low either, which means we have not run into over-fitting. The actual, predicted train and predicted test values can

Table 8: LSTM Sequential Model Summary

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 60, 50)	10,400
lstm_3 (LSTM)	(None, 50)	20,200
dense_1 (Dense)	(None, 1)	51

Total params	30,651 (119.73 KB)
Trainable params	30,651 (119.73 KB)
Non-trainable params	0 (0.00 B)

Train RMSE	0.3175
Train MAE	0.2384
Test RMSE	0.5714
Test MAE	0.5290

be bound in 7.

Figure 7: Actual Price, Train Prediction and Test Prediction from LSTM

To improvise further, we tended to ensemble the models together. XGBoost is also an ensemble model of many CARTs (Developers, 2024). Our XGBoost model is highly efficient, however, after a careful feature selection. The train-test split for XGBoost also follows the standard 80:20 ratio, so as to avoid overfitting the model and evaluate only on a limited test set. The results are given in 9.

XGBoost took the predictions accuracy to an all new standard. However, it comes with a wide range of hyper-parameters to avoid over-fitting. We refined the XGBoost model to do parameter tuning via Bayesian optimization and cross validation. The fine tuned XGBoost is now ready to be used in the ensemble we propose. Our ensemble is another layering over the already existing XGBoost ensemble, so we intend to keep it simple. It is a weighted averaging of the $ARIMA(2,1,2)$, $LSTM$ and $XGBoost$ models. The weights selected by us is $[ARIMA(2,1,2) : 0.7, LSTM : 0.2, XGBoost : 0.1]$. The rationale for selecting these weights is the *principle of parsimony*, so higher weights are given to more complex models. The overall performance by this ensemble is given in 10.

A comparison of all the models over the 30 day forecast is collected in 11. It may seem the final ensemble is under-performing XGBoost alone, but we yet respect the *wisdom of crowd* ideology and the principle of parsimony.

Table 9: Top 10 Features and XGBoost Regressor Performance

Feature	Importance
Low	3114.99
High	987.85
Price_Lag_1	177.68
Price_Lag_3	15.55
Open_Lag_2	4.49
Low_Lag_1	3.99
Open_Lag_4	2.11
High_Lag_1	2.03
High_Lag_4	1.63
Open_Lag_1	1.61
Root Mean Squared Error (RMSE)	0.1219
Mean Absolute Error (MAE)	0.0871

Model	Ensemble Weight
ARIMA(2,1,2)	0.7
LSTM	0.2
XGBoost	0.1

Metric	Value
Forecast RMSE	0.1808
Forecast MAE	0.1619

Table 10: Ensemble Model Performance

Model	Test RMSE	Test MAE
ARIMA(2,1,2)	0.2049	0.1451
LSTM	0.6486*	0.6449*
XGBoost	0.0841*	0.0631*
Ensemble	0.1808	0.1619

Table 11: Comparison of 30-Day Forecast Performance

*The mismatch in performance of LSTM and XGBoost from earlier quoted results is due to change in the test set to latest 30 days only.

Conclusion

In this research, we analyzed the USD/INR exchange rate using various time series and machine learning models, including ARIMA, SARIMA, GARCH, LSTM, and XGBoost, ultimately combining them in an ensemble approach. We found that while ARIMA models provided robust baseline performance, XGBoost excelled in predictive accuracy. However, the ensemble model balanced predictive stability with complexity by blending ARIMA's simplicity, LSTM's temporal dependencies, and XGBoost's precision. The point of focus throughout this work was to explore robust methods to tackle any univariate time series analysis and forecasting task. The volatility of exchange rate markets are linked to several other macro economic indicators, and modeling all of them may not be feasible, which makes out of sample errors a ground reality.

Limitations and Further Contributions: The main limitations of this work include not readjusting the ARIMA model for each of the regimes after structural break analysis and building a meta model that fits all the individual models. Further studies could incorporate advanced ensembling techniques like stacking, where a meta model learns the ideal weights based on the error minimization on a validation set, effectively determining which models contribute more or less to the final prediction.

Disclaimer: This content may have been created with the assistance of AI technology. However, it has been reviewed and edited by a human to ensure accuracy and quality.

Bibliography

Alkaline-ML. (2024). Tips and tricks — pmdarima 2.0.4 documentation [Accessed: 2024-11-12]. https://alkaline-ml.com/pmdarima/tips_and_tricks.html

Bartaria, L. (2021). Model selection and forecasting performances of advance univariate time series models during the pandemic period: A case of inr/usd exchange rate [Received: 07th August 2021; Revised: 21st September 2021; Accepted: 19th October 2021]. *Indian Journal of Economics and Business*, *20*(4), 1–10. http://www.ashwinanokha.com/IJEB.php

Borelli, C. (2024). *Binary segmentation (binseg)* [ruptures Documentation]. https://centre-borelli.github.io/ruptures-docs/user-guide/detection/binseg/

Chollet, F. (2020, April). The sequential model [Last modified: June 25, 2023]. https://keras.io/guides/sequential_model/

Crone, S. F., & Kourentzes, N. (2010). Feature selection for time series prediction–a combined filter and wrapper approach for neural networks. *Neurocomputing*, *73*(10-12), 1923–1936.

D, D. T., Bung, D. P., & M, D. J. (2024). Predicting exchange rate between us dollar (usd) and indian rupee (inr): An empirical analysis using sarima model [P-ISSN: 2617-5754, E-ISSN: 2617-5762]. *International Journal of Research in Finance and Management*, *7*(1), 56–63. https://doi.org/https://doi.org/10.33545/26175754.2024.v7.i1a.283

Darvas, Z., & Schepp, Z. (2020). Forecasting exchange rates of major currencies with long maturity forward rates. *Working Paper*.

Developers, X. (2024). Xgboost python api [Accessed: 2024-11-12]. https://xgboost.readthedocs.io/en/stable/python/python_api.html

Gupta, A. (2021). *Forecasting the daily exchange rate of rupee against major currencies* (Officer Trainee). Indian Economic Service. Delhi.

Investing.com. (2024, October 17). Investing.com.

Islam, S. F. N., Sholahuddin, A., & Abdullah, A. S. (2021). Extreme gradient boosting (xgboost) method in making forecasting application and analysis of usd exchange rates against rupiah [Department of Computer Science, Universitas Padjadjaran, Jl. Raya Bandung Sumedang km 21 Jatinangor, Sumedang 45363, Indonesia]. *Journal of Physics: Conference Series*, *1722*(1), 012016. https://doi.org/10.1088/1742-6596/1722/1/012016

Li, Y. N., Li, D., & Fryzlewicz, P. (2022). Detection of multiple structural breaks in large covariance matrices. *Journal of Business & Economic Statistics*, *41*(3), 846–861. https://doi.org/10.1080/07350015.2022.2076686

Perktold, J., Seabold, S., Taylor, J., & statsmodels developers. (2024a). *Statsmodels.graphics.tsaplots.plot_acf* [Version 0.14.4]. https://www.statsmodels.org/dev/generated/statsmodels.graphics.tsaplots.plot_acf.html

Perktold, J., Seabold, S., Taylor, J., & statsmodels developers. (2024b). *Statsmodels.tsa.stattools.adfuller* [Version 0.14.4]. https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html

Siddhartha, M. (2020, August). Foreign exchange rate prediction using deep learning (ann, lstm & gru).

Sjölander, P. (2011). A stationary unbiased finite sample arch-lm test procedure. *Applied Economics*, *43*(8). https://doi.org/10.1080/00036840802600046

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding lstm: A tutorial into long short-term memory recurrent neural networks [cs.NE]. *arXiv*, *1909.09586v1*. https://arxiv.org/abs/1909.09586

Credit Statement

Name	Contributions
Harsh	SARIMA model forecasting LSTM model theory XGBoost model theory Report review Data Collection & Cleaning Data Preparation
Harsh Choudhary	ARIMA model forecasting LSTM model forecasting Ensemble modeling Structural breaks analysis Bayesian optimization Report Writing Documentation & Code Management
Ayushman Baghel	XGBoost model forecasting Feature extraction Feature importance analysis Feature correlation analysis Information gain methods Methodology & Model Implementation
Avinash Choudhary	ARCH-LM test SARIMA-GARCH forecasting Volatility analysis Confidence interval analysis Exploratory Data Analysis (EDA) Visualizations

Appendix

LSTM Model Theory

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to handle long-term dependencies in sequential data, overcoming the limitations of traditional RNNs that face vanishing and exploding gradient problems.

Structure of an LSTM Cell

An LSTM cell contains:

- **Cell State** C_t: The memory of the network, which stores information across time steps.
- **Hidden State** h_t: Represents the output of the cell at time step t.
- **Gates**: These regulate the flow of information:
  1. Forget Gate f_t
  2. Input Gate i_t
  3. Output Gate o_t

Mathematics of LSTM

At each time step t, an LSTM cell operates as follows:

Forget Gate
The forget gate determines what information from the previous cell state C_{t-1} should be retained or discarded:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

where:

- W_f and b_f are the forget gate's weights and bias.
- $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state h_{t-1} and the current input x_t.
- σ is the sigmoid function.

Input Gate
The input gate determines which new information will be added to the cell state. This is calculated in two steps:

1. **Candidate Cell State $\tilde{C}_t$**:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

2. **Input Gate Activation i_t**:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{10}$$

Update Cell State
The updated cell state C_t is computed using both the forget gate f_t and the input gate i_t:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{11}$$

where $\odot$ denotes element-wise multiplication.
Output Gate
The output gate controls the updated hidden state h_t, based on the cell state C_t:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{12}$$

$$h_t = o_t \odot \tanh(C_t) \tag{13}$$

Summary of LSTM Equations

To summarize, the LSTM cell at each time step t computes:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{14}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{15}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{16}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{17}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{18}$$
$$h_t = o_t \odot \tanh(C_t) \tag{19}$$

XGBoost Model Theory
XGBoost, or eXtreme Gradient Boosting, is a scalable and efficient implementation of the Gradient Boosting Machine (GBM) algorithm. It enhances performance by overcoming the limitations of traditional gradient boosting with optimizations in speed and accuracy.
Gradient Boosting Basics
Gradient Boosting is an ensemble learning technique that sequentially builds models, where each new model attempts to correct the errors of previous models. Given a dataset with n examples, $D = \{(x_i, y_i)\}_{i=1}^{n}$, gradient boosting seeks to minimize a loss function $L(y, \hat{y})$ by constructing an ensemble prediction $\hat{y} = \sum_{t=1}^{T} f_t(x)$, where f_t represents each weak learner (typically a decision tree).

Loss Function in XGBoost

The loss function in XGBoost is composed of two parts:

1. **Training Loss**: Measures how well the model predicts the target.
2. **Regularization Term**: Controls model complexity to prevent overfitting.

The objective function $\mathcal{O}$ at iteration t is:

$$\mathcal{O}^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{T} \Omega(f_k),$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$, T is the number of leaves, γ and λ are hyperparameters, and w is the vector of weights for the leaves.

Gradient Descent Approximation

At each iteration t, XGBoost adds a new function $f_t(x)$ to minimize the objective. The function is optimized using a second-order Taylor expansion of the loss around the current prediction $\hat{y}^{(t-1)}$:

$$\mathcal{O}^{(t)} \approx \sum_{i=1}^{n} \left[L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t),$$

where: - $g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$ (first-order gradient), - $h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)^2}}$ (second-order gradient, Hessian).

Decision Tree as a Base Learner

Each f_t in XGBoost is a decision tree. The tree splits data to create feature space regions, each assigned a score (weight). Splits are evaluated by an impurity score, which incorporates both first and second derivatives.

Gain of a Split

The gain of a split at a node is calculated as:

$$\text{Gain} = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma,$$

where: - G_L and G_R: sums of gradients for the left and right child nodes, - H_L and H_R: sums of Hessians for the left and right child nodes, - λ: regularization parameter, - γ: penalty for adding new leaves.

Leaf Weight Optimization

After splitting, XGBoost assigns a weight w_j to each leaf to minimize the objective. The optimal weight w_j^* for a leaf j is:

$$w_j^* = -\frac{G_j}{H_j + \lambda},$$

where G_j and H_j are the sum of gradients and Hessians for all data in leaf j.

Regularization

Regularization in XGBoost penalizes model complexity. The regularization term $\Omega(f_t)$ controls the number of leaves T and leaf weights w_j:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2.$$

This reduces overfitting by discouraging deep trees and high leaf weights.

Shrinkage (Learning Rate)

XGBoost applies a learning rate η to each tree to scale down its impact:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta f_t(x).$$

Shrinkage prevents overfitting by allowing finer updates with each tree.

Column Subsampling

XGBoost also uses column subsampling, where each tree is trained on a random subset of features, similar to Random Forests. This reduces tree correlation and enhances model robustness.

Summary of XGBoost Algorithm

1. **Initialize predictions** with a constant (e.g., mean of y).
2. **Compute gradients** and Hessians for each data point based on current predictions.
3. **Build a tree**:
    - For each node, calculate the gain of each split and select the highest gain split.
    - Assign weights to leaves by minimizing the objective function with gradients and Hessians.
4. **Update predictions** by adding scaled predictions from the new tree.
5. **Repeat** steps 2-4 for a specified number of iterations or until convergence.

Abbreviations

Letter	Abbreviation	Full Form
A	ACF	Auto Correlation Function
	ADF	Augmented Dickey Fuller
	AIC	Akaike Information Criterion
	ANN	Artificial Neural Networks
	ARCH	Auto Regressive Conditional Heteroskedasticity
	ARIMA	Auto Regressive Integrated Moving Average
B	Binseg	Binary Segmentation
	BIC	Bayesian Information Criterion
C	CART	Classification And Regression Tree
G	GARCH	Generalized Auto Regressive Conditional Heteroskedasticity
I	INR	Indian Rupee
L	LM	Lagrange Multiplier
	LSTM	Long Short Term Memory
M	MAE	Mean Absolute Error
	PACF	Partial Auto Correlation Function
R	RMSE	Root Mean Square Error
	RNN	Recurrent Neural Networks
	ROI	Return on Investment
S	SARIMA	Seasonal Auto Regressive Integrated Moving Average
	SARIMAX	Seasonal Auto Regressive Integrated Moving Average with Exogenous Variables
U	USD	US Dollar
X	XGBoost	Extreme Gradient Boosting