Time Series Forecasting of INR/USD Exchange Rates and Gold Prices: A Comparative Analysis of ARIMA and VAR Models

Report by:
Amogh Jagini
Larren Peter Pinto

Under the guidance of:
Dr. Harini Srivastava

Symbiosis Statistical Institute, Pune

Abstract

The study examines the predictive performance of ARIMA and VAR models in forecasting India-US exchange rates and gold prices using a comprehensive monthly dataset spanning from 2011 to 2024. After confirming non-stationarity in most variables through ADF testing, differencing was applied to achieve stationarity. For gold price forecasting, the VAR model outperformed ARIMA according to AICc metrics, while ARIMA proved superior for exchange rate forecasting. Granger causality tests revealed that Repo Rate significantly influences gold prices, while percentage change, forex reserves, stock indices, and the DXY Index strongly affect exchange rates. These findings provide valuable insights for investors, policymakers, and financial analysts navigating currency and commodity markets.

1. Introduction

Exchange rates and gold prices are critical economic indicators that significantly impact global trade, investment decisions, and monetary policy. The Indian Rupee to US Dollar (INR/USD) exchange rate particularly influences India's import-export dynamics, inflation rates, and foreign investments. Similarly, gold prices serve as both an investment vehicle and a hedge against economic uncertainty, making their accurate forecasting valuable for portfolio diversification and risk management.

The volatility of these financial variables poses substantial challenges for forecasting. Exchange rates are influenced by a complex interplay of factors including interest rate differentials, inflation rates, political stability, and market sentiment. Gold prices similarly respond to diverse factors such as global economic conditions, inflation expectations, central bank policies, and geopolitical tensions.

Traditional time series methods like Autoregressive Integrated Moving Average (ARIMA) models have been widely employed for univariate forecasting, while Vector Autoregression (VAR) models allow for capturing interdependencies among multiple variables. These approaches have shown varying degrees of success in financial forecasting, with their relative performance often depending on specific market conditions and timeframes.

This study aims to compare the effectiveness of ARIMA and VAR models in forecasting INR/USD exchange rates and gold prices, providing insights for investors, policymakers, and financial analysts navigating these markets.

1.1 Objective

The primary objectives of this research are:

1. To develop accurate forecasting models for INR/USD exchange rates and gold prices using time series techniques, specifically ARIMA and VAR models
2. To identify key macroeconomic variables that significantly influence exchange rates and gold prices through Granger causality testing
3. To compare the predictive performance of univariate (ARIMA) and multivariate (VAR) approaches in financial time series forecasting
4. To generate reliable forecasts for exchange rates and gold prices for 2025

The study hypothesizes that multivariate VAR models will outperform univariate ARIMA models due to their capacity to capture complex interdependencies among economic variables. Additionally, we expect to find bidirectional causality between certain macroeconomic indicators and our target variables.

2. Methodology

2.1 Data Collection and Sources

The dataset comprises monthly observations from January 2011 to December 2024, incorporating 14 economic variables. Data was sourced from reputable financial and economic databases:

Variable	Description	Source
Exchange_rate(INR)	Monthly average INR/USD exchange rate	Reserve Bank of India (RBI)
percentage_change	Month-over-month percentage change in exchange rate	Calculated
Repo_rate	RBI repo rate at month-end	Reserve Bank of India
US_Fed_rate	US Federal Reserve benchmark interest rate	Federal Reserve Economic Data (FRED)
CPI_India	Consumer Price Index for India	Ministry of Statistics and Programme Implementation (MOSPI)
CPI_US	Consumer Price Index for US	U.S. Bureau of Labor Statistics
Forex_reserves($)	India's foreign exchange reserves in USD	Reserve Bank of India
Nifty_50	NSE Nifty 50 index closing value	National Stock Exchange of India
S&P_500	S&P 500 index closing value	S&P Dow Jones Indices
Crude_Oil($ per Barrel)	Average crude oil price per barrel in USD	U.S. Energy Information Administration
DXY_Index	US Dollar Index value	Federal Reserve Economic Data (FRED)
Gold_price(USD/Ounce)	Gold price per ounce in USD	World Gold Council
Trade_Balance(INR)	India's trade balance in INR	Ministry of Commerce and Industry

2.2 Data Preprocessing

The following preprocessing steps were implemented:

1. Handling date formatting: Converting the 'year-month' column to datetime format
2. Exploratory Data Analysis (EDA): Examining distribution, trends, and statistical properties

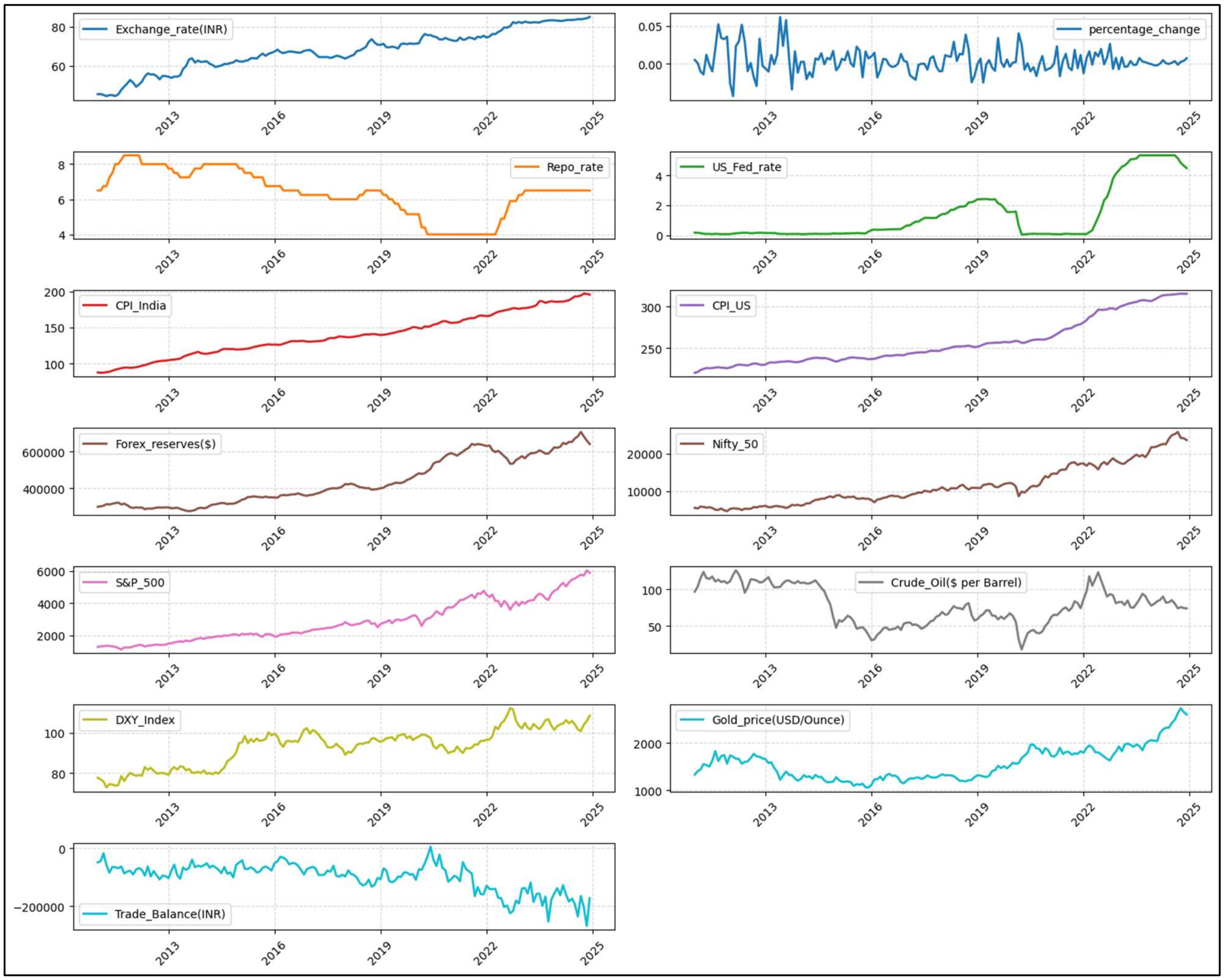
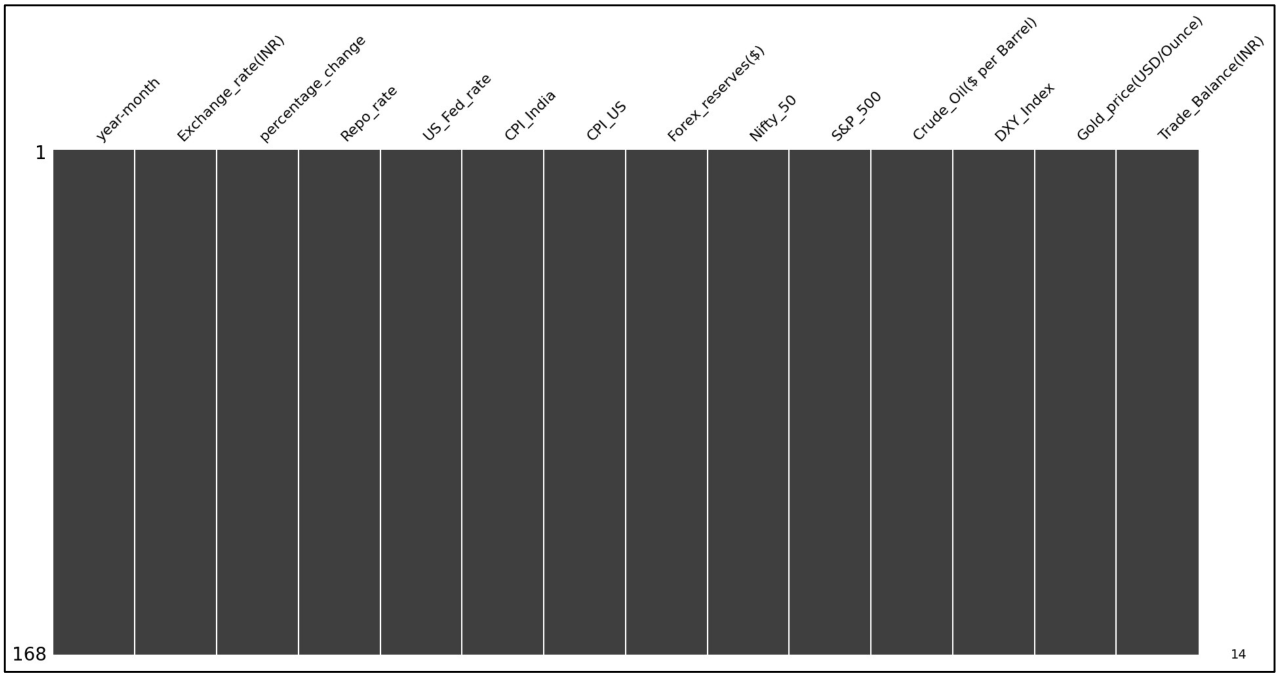


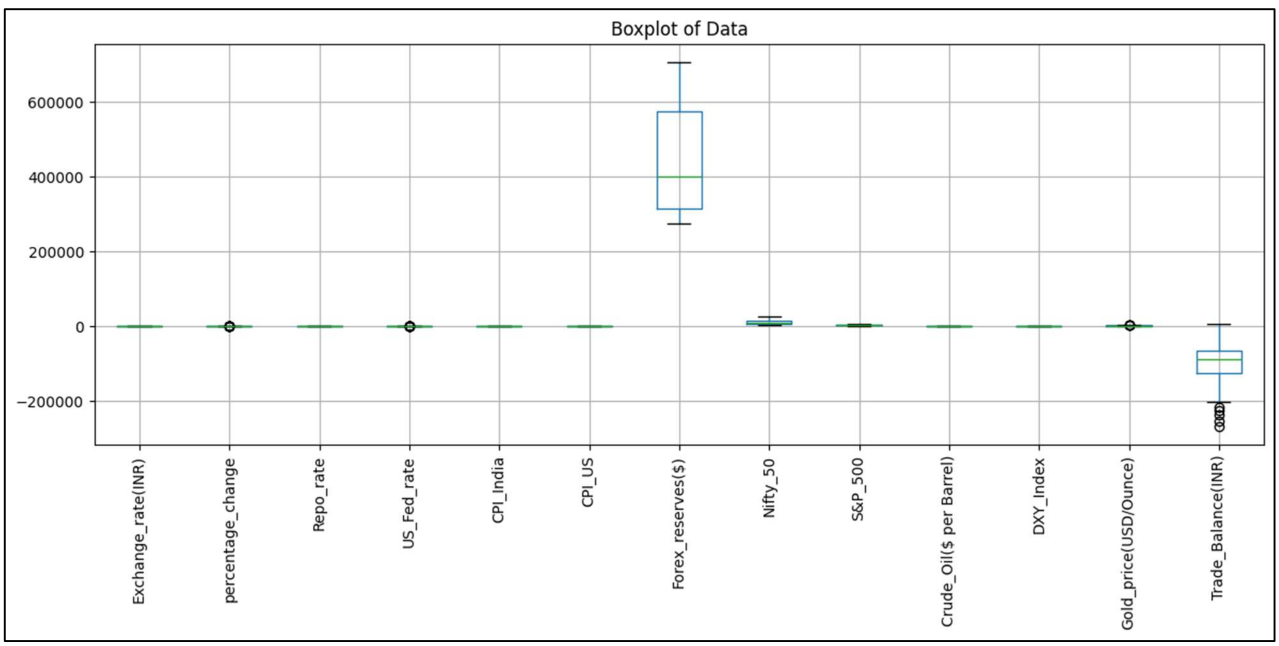
Fig 1: TIME SERIES TRENDS FOR ALL INDICATORS

3. Missing value detection: Confirmed no missing values in the dataset



<u>Fig 2</u>: MISSING PLOT

4. Outlier detection: Visualized using boxplots and standardized data for better comparison



<u>Fig 3</u>: BOXPLOT FOR DETECTING OUTLIERS

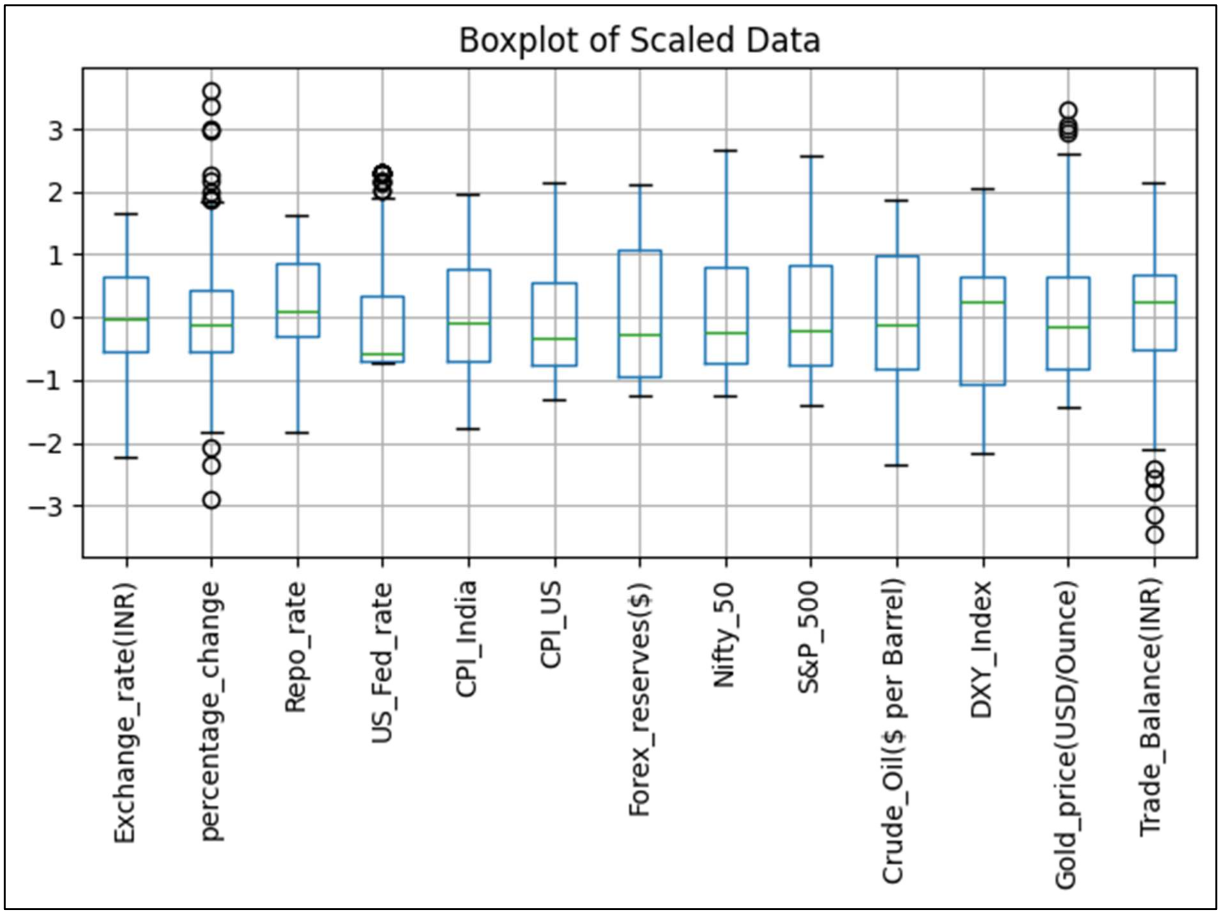


Fig 4: BOXPLOT FOR DETECTING OUTLIERS (SCALED DATA)

5. Correlation analysis: Identified relationships between variables using heatmaps

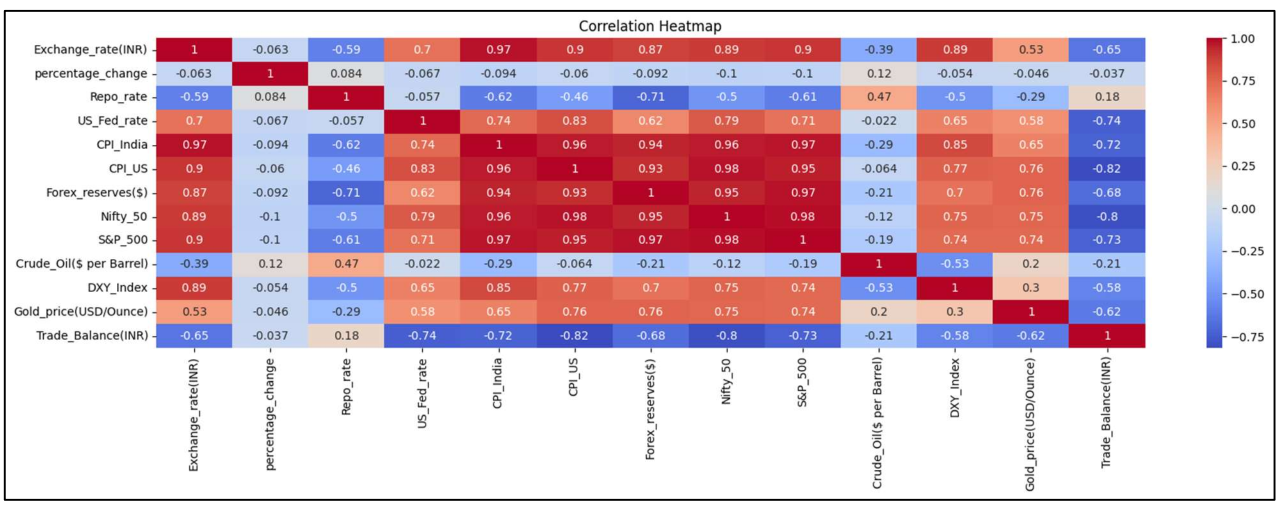


Fig 5: CORRELATION HEATMAP

2.3 Stationarity Testing

The Augmented Dickey-Fuller (ADF) test was applied to check stationarity, a critical assumption for time series modeling. Most variables were found to be non-stationary, requiring differencing to achieve stationarity:

- First-order differencing made exchange rates, repo rates, Fed rates, forex reserves, stock indices, crude oil prices, DXY index, and gold prices stationary
- Second-order differencing was required for CPI (India and US) and trade balance

This preprocessing ensured that the data met the assumptions required for reliable time series modeling.

2.4 Modeling Techniques

ARIMA Model

The Autoregressive Integrated Moving Average (ARIMA) model combines three components:

- Autoregressive (AR) terms: Using past values to predict future values
- Integrated (I) component: Differencing to achieve stationarity
- Moving Average (MA) terms: Incorporating past forecast errors

The ARIMA model is specified as ARIMA(p,d,q) where:

- p: Order of the autoregressive component
- d: Degree of differencing
- q: Order of the moving average component

Model selection was guided by analyzing Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots to identify appropriate orders.

Vector Autoregression (VAR) Model

The VAR model is a multivariate extension that captures linear interdependencies among multiple time series. Each variable is modeled as a function of past values of itself and past values of other variables in the system. For a two-variable system with lag order p, the VAR model is represented as:

$$y_{1t} = c_1 + \varphi_{11,1}y_{1,t-1} + \dots + \varphi_{11,p}y_{1,t-p} + \varphi_{12,1}y_{2,t-1} + \dots + \varphi_{12,p}y_{2,t-p} + \varepsilon_{1t}$$

$$y_{2t} = c_2 + \varphi_{21,1}y_{1,t-1} + \dots + \varphi_{21,p}y_{1,t-p} + \varphi_{22,1}y_{2,t-1} + \dots + \varphi_{22,p}y_{2,t-p} + \varepsilon_{2t}$$

Optimal lag selection was determined using information criteria including AIC, BIC, and HQIC.

2.5 Granger Causality Testing

Granger causality tests were employed to identify significant predictive relationships between variables. This approach examines whether past values of one variable provide statistically significant information about future values of another variable. For each variable pair, we tested causality across multiple lag structures (1-8 lags).

3. Implementation

3.1 Data Splitting and Preprocessing

The dataset was split into training (80%) and testing (20%) sets to evaluate model performance. Differencing was applied to achieve stationarity:

- First differencing for most variables
- Second differencing for CPI indices and trade balance

3.2 Model Parameterization

For gold price forecasting:

- ARIMA: Based on ACF/PACF analysis, an ARIMA(8,0,8) model was selected for the differenced series
- VAR: The optimal lag order was determined to be 8 using the AIC criterion

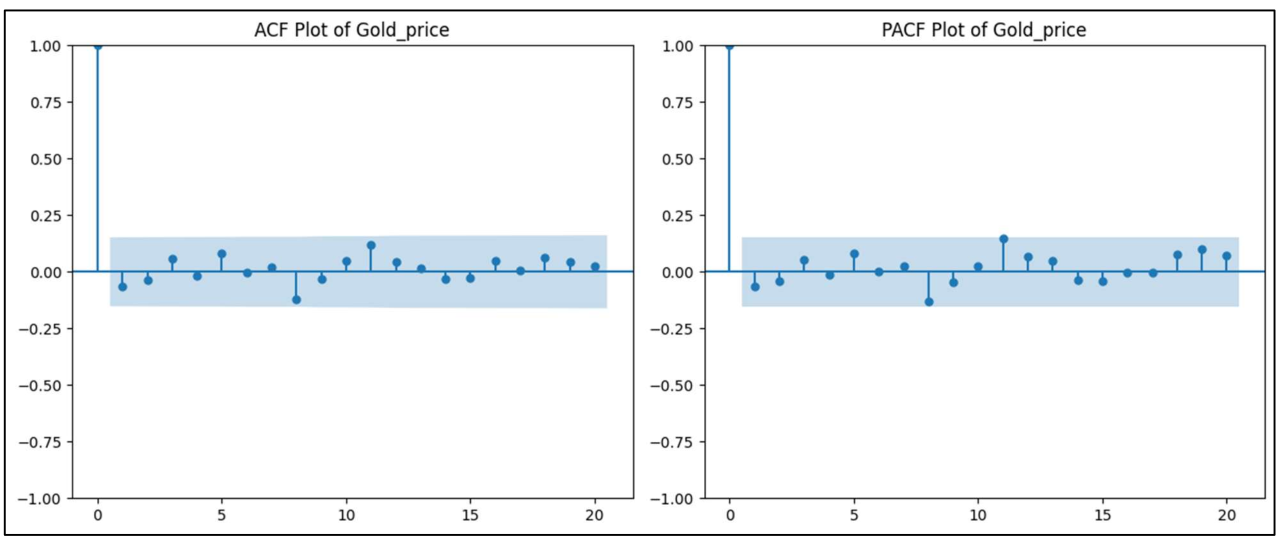


Fig 6: ACF AND PACF PLOT FOR GOLD PRICE

For exchange rate forecasting:

- ARIMA: An ARIMA(4,0,9) model was fitted to the differenced exchange rate series
- VAR: The same VAR model with lag order 8 was applied

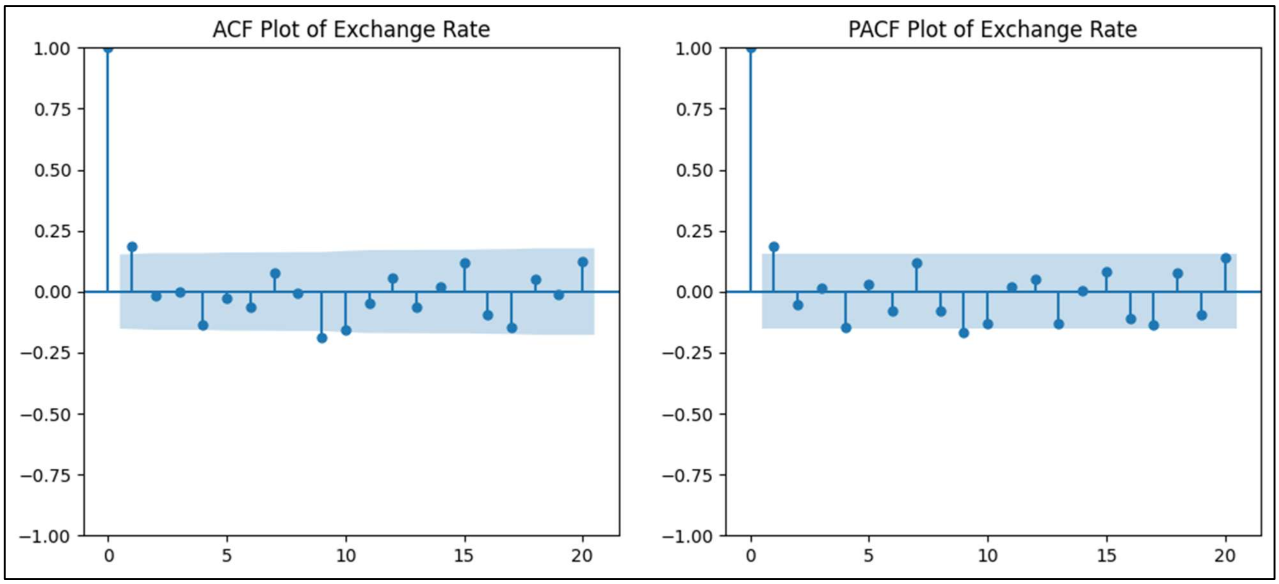


Fig 7: ACF AND PACF PLOT FOR EXCHANGE RATES

3.3 Forecasting Process

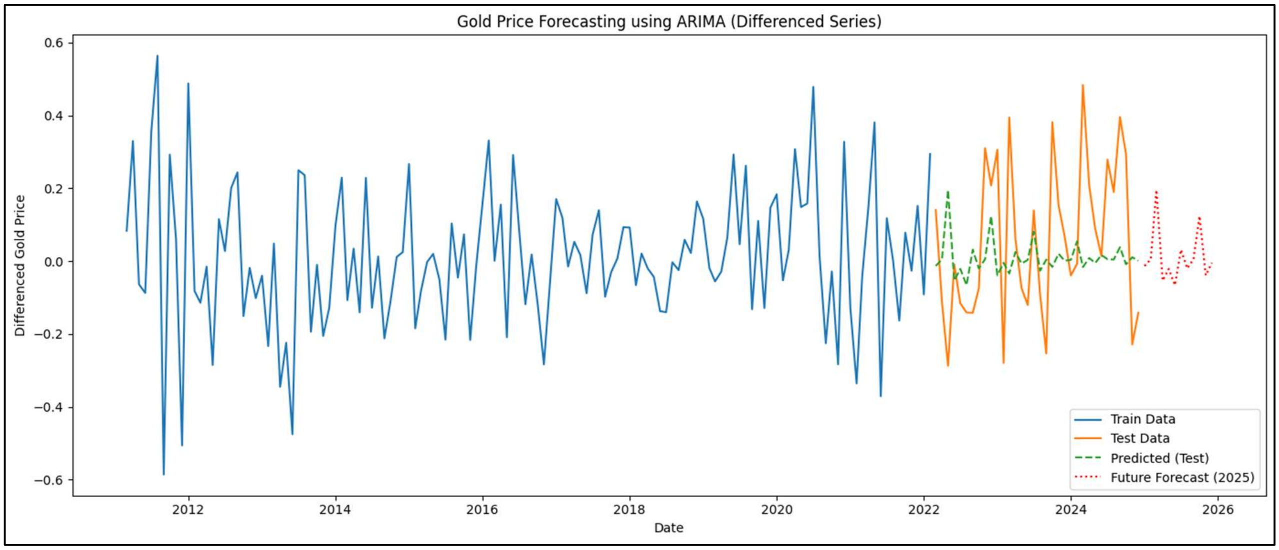
The following procedure was implemented for both target variables:

1. Train models on the training dataset
2. Generate forecasts for the test period to evaluate performance
3. Refit models on the full dataset
4. Generate future forecasts for 12 months (2025)

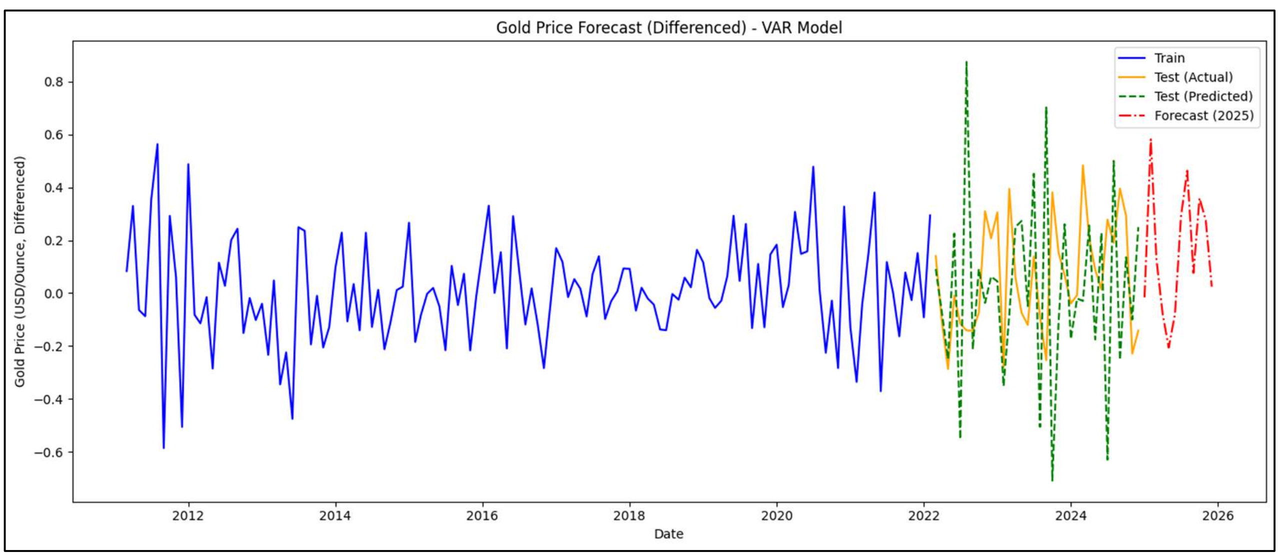
Performance metrics included Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and corrected AIC (AICc).

5. Results

5.1 Gold Price Forecasting Results



<u>Fig 8</u>: GOLD PRICE FORECASTING GRAPH - ARIMA MODEL



<u>Fig 9</u>: GOLD PRICE FORECASTING GRAPH - VAR MODEL

Performance metrics for gold price forecasting:

Model	RMSE	MAPE	AIC	BIC	AICc
ARIMA(8,0,8)	0.2290	125.43%	-40.71	11.18	-34.66
VAR(8)	0.4317	314.06%	-74.48	-43.43	781.68

According to the AICc metric, the ARIMA model outperformed the VAR model for gold price forecasting. However, the high MAPE values indicate challenges in accurate prediction, likely due to gold's inherent volatility.

The Granger causality test revealed that the Repo Rate significantly predicts gold prices across multiple lags ($p < 0.05$ for lags 1-7). This suggests that India's monetary policy decisions have a substantial impact on gold prices. The DXY Index showed marginal significance at lag 1 ($p = 0.0447$), while other variables did not demonstrate consistent predictive relationships with gold prices.

5.2 Exchange Rate Forecasting Results

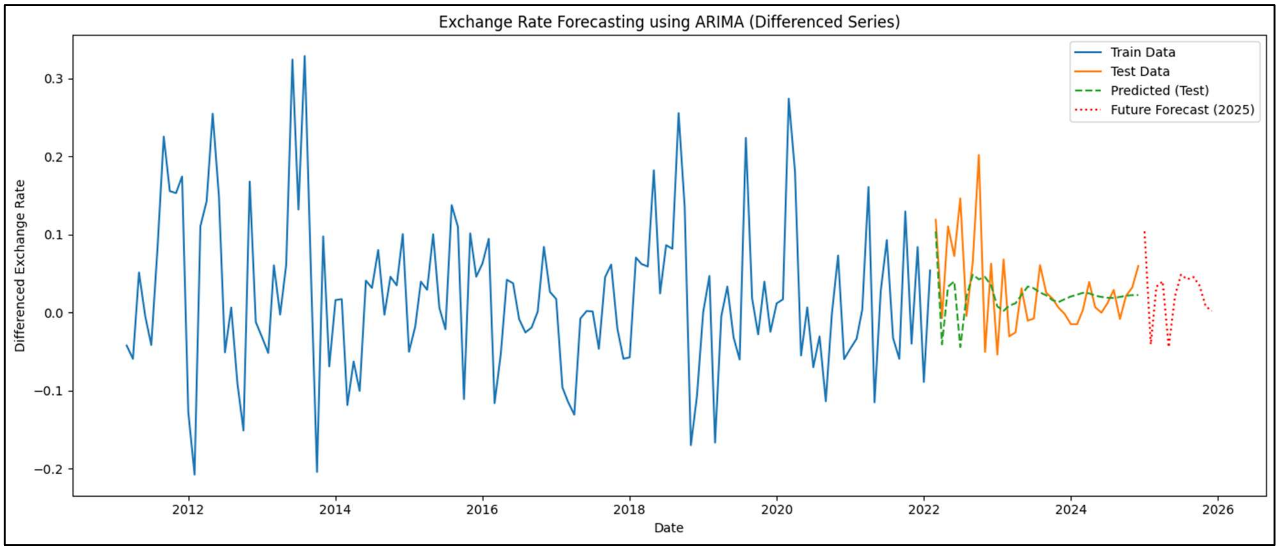


Fig 10: EXCHANGE RATE FORECASTING GRAPH - ARIMA MODEL

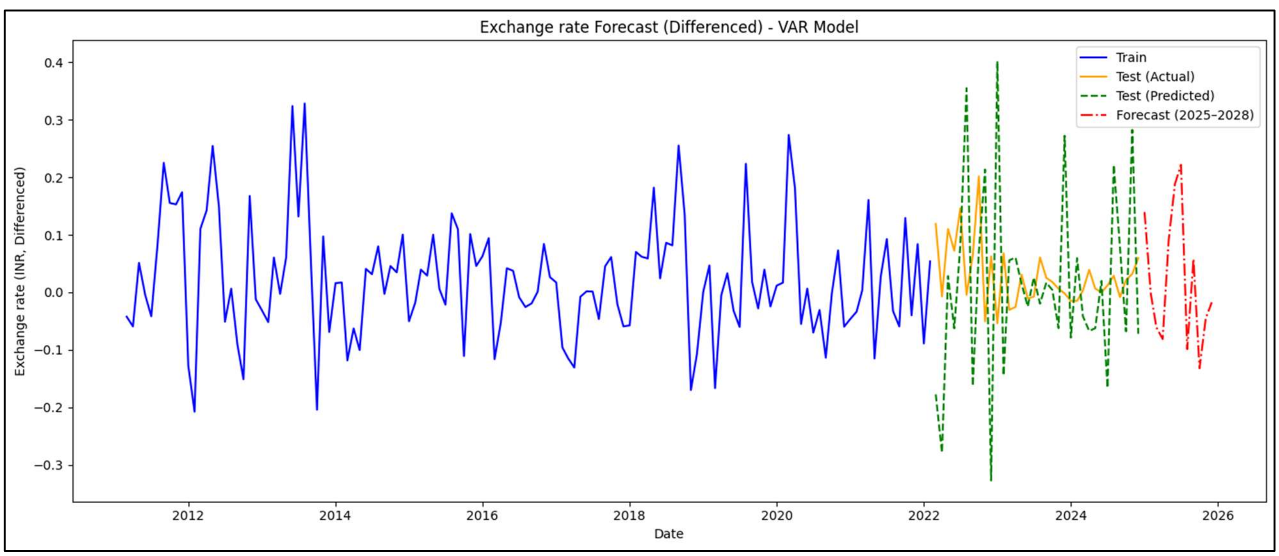


Fig 11: EXCHANGE RATE FORECASTING GRAPH - VAR MODEL

Performance metrics for exchange rate forecasting:

Model	RMSE	MAPE	AIC	BIC	AICc
ARIMA(4,0,9)	0.0552	382.17%	-224.20	-180.96	-220.06
VAR(8)	0.1847	13.98%	-74.48	-43.43	1162.19

For exchange rate forecasting, the ARIMA model demonstrated superior performance based on the AICc metric and RMSE, despite a higher MAPE compared to the VAR model.

The Granger causality test identified several variables with significant predictive relationships to exchange rates:

- Percentage change ($p < 0.05$ across all lags)
- Forex reserves ($p < 0.05$ for lags 1-4)
- Nifty 50 ($p < 0.01$ across all lags)
- S&P 500 ($p < 0.01$ for lags 1-6)
- DXY Index ($p < 0.01$ across all lags)
- Gold prices (significant at higher lags, $p = 0.0448$ at lag 8)

Variables like repo rate, US Fed rate, CPI indices, crude oil prices, and trade balance did not show consistent predictive power for exchange rates.

6. Conclusion

This study demonstrates the effectiveness of time series models in forecasting INR/USD exchange rates and gold prices. For gold price forecasting, the ARIMA model showed better performance according to the AICc metric, suggesting that incorporating multiple economic variables improves predictive accuracy for gold prices. Similarly, the ARIMA model proved more effective for exchange rate forecasting, indicating that the historical pattern of exchange rates themselves contains substantial predictive information.

The Granger Causality tests provided valuable insights into the economic drivers of both variables. For gold prices, the significant influence of the Repo Rate highlights the importance of monetary policy in gold price movements. This aligns with economic theory, as interest rates affect the opportunity cost of holding non-yielding assets like gold.

For exchange rates, the causal relationships were more diverse, with significant influences from stock market indices (Nifty 50, S&P 500), forex reserves, and the DXY Index. This multi-faceted influence reflects the complex nature of currency valuation, which depends on both domestic and international economic factors.

These findings have important implications for investors, policymakers, and financial analysts. Investors can enhance their portfolio strategies by considering the identified relationships, particularly the influence of monetary policy on gold prices and the multiple factors affecting exchange rates. Policymakers can better understand the potential impact of their decisions on these critical financial variables.

7. Future Perspectives

Several avenues for future research emerge from this study:

1. **Integration of Machine Learning Approaches**: Exploring non-linear models like Neural Networks, Random Forests, or Support Vector Machines could potentially capture more complex patterns not addressed by traditional time series methods.

2. **Incorporation of Sentiment Analysis**: Including text-based data from news, social media, or central bank communications could enhance forecasting accuracy by capturing market sentiment and expectations.

3. **Regime-Switching Models**: Implementing models that account for different economic regimes might better capture the changing dynamics of financial markets across various economic conditions.

4. **Higher Frequency Data**: Utilizing daily or intraday data could provide more granular insights, especially for short-term forecasting and trading strategies.

5. **External Shock Analysis**: Developing models that can better account for external shocks such as the COVID-19 pandemic or geopolitical events would increase robustness in volatile periods.

6. **Expanding Variable Set**: Including additional variables such as portfolio flows, forward premium, and inflation expectations could potentially improve model performance.

The methodological framework developed in this study provides a solid foundation for these future research directions, contributing to our understanding of financial market dynamics and improving forecasting capabilities for these economically significant variables.

CODE

In [15]:
```
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_excel('/content/Final_Dataset.xlsx')
```

In [3]:
```
data.head(3)
```

Out[3]:

	Date	year-month	Exchange_rate(INR)	percentage_change	Repo_rate	US_Fed_
0	2010-12-31	2010-12	45.1568	0.000000	6.25	
1	2011-01-31	2011-01	45.3934	0.005240	6.50	
2	2011-02-28	2011-02	45.4358	0.000934	6.50	

In [4]:
```
data.tail(3)
```

Out[4]:

	Date	year-month	Exchange_rate(INR)	percentage_change	Repo_rate	US_Fe
166	2024-10-31	2024-10	84.0295	0.002641	6.5	
167	2024-11-30	2024-11	84.3644	0.003986	6.5	
168	2024-12-31	2024-12	84.9862	0.007370	6.5	

In [5]:
```
data.drop('Date',axis=1,inplace = True)
```

In [6]:
```
data.drop(index=0,inplace=True)
data.reset_index(drop=True, inplace=True)
```

Exploratory Data Analysis (EDA)

Checking dataset shape & structure

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   year-month                168 non-null    object
 1   Exchange_rate(INR)        168 non-null    float64
 2   percentage_change         168 non-null    float64
 3   Repo_rate                 168 non-null    float64
 4   US_Fed_rate               168 non-null    float64
 5   CPI_India                 168 non-null    float64
 6   CPI_US                    168 non-null    float64
 7   Forex_reserves($)         168 non-null    float64
 8   Nifty_50                  168 non-null    float64
 9   S&P_500                   168 non-null    float64
 10  Crude_Oil($ per Barrel)   168 non-null    float64
 11  DXY_Index                 168 non-null    float64
 12  Gold_price(USD/Ounce)     168 non-null    float64
 13  Trade_Balance(INR)        168 non-null    float64
dtypes: float64(13), object(1)
memory usage: 18.5+ KB
```

In [8]:
```
data.describe()
```

Out[8]:

	Exchange_rate(INR)	percentage_change	Repo_rate	US_Fed_rate	CP
count	168.000000	168.000000	168.000000	168.000000	168.
mean	67.708119	0.003897	6.380357	1.304583	139.
std	10.482537	0.016002	1.311408	1.753730	29.
min	44.370000	-0.042309	4.000000	0.050000	87.
25%	62.005775	-0.004643	5.975000	0.090000	118.
50%	67.418150	0.001982	6.500000	0.285000	136.
75%	74.563225	0.010660	7.500000	1.920000	161.
max	84.986200	0.061561	8.500000	5.330000	196.

Identifying duplicate rows

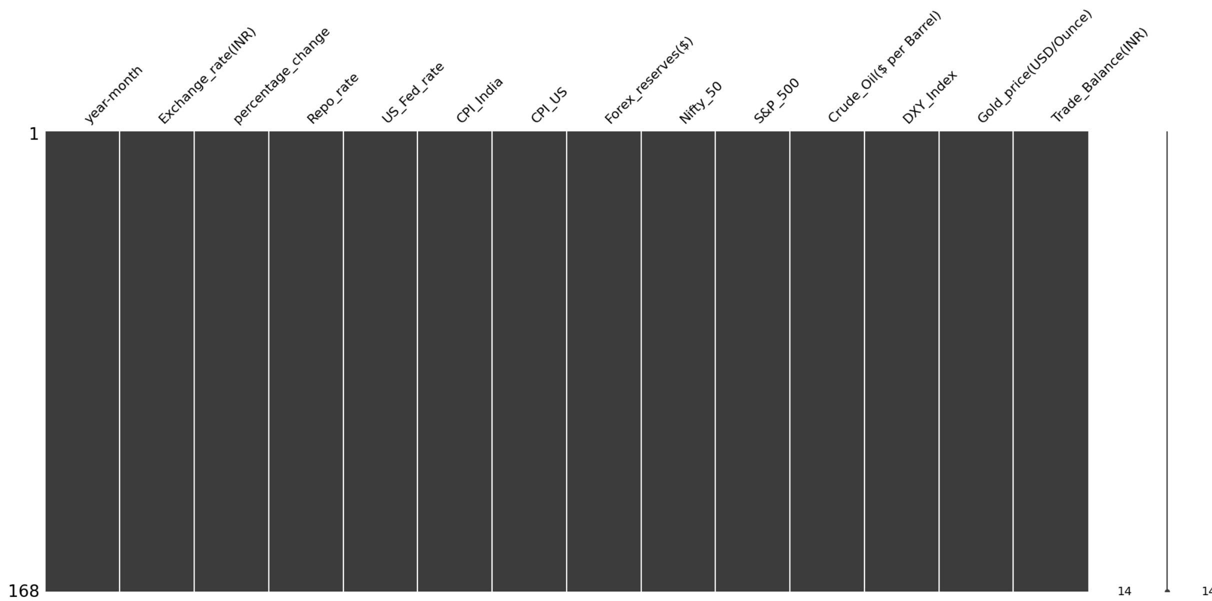
In [9]:
```
data.duplicated().sum()
```

Out[9]: np.int64(0)

Identifying missing values

In [10]:
```
msno.matrix(data)
```

Out[10]: <Axes: >



In [11]:
```
data.isnull().sum()
```

Out[11]:

	0
year-month	0
Exchange_rate(INR)	0
percentage_change	0
Repo_rate	0
US_Fed_rate	0
CPI_India	0
CPI_US	0
Forex_reserves($)	0
Nifty_50	0
S&P_500	0
Crude_Oil($ per Barrel)	0
DXY_Index	0
Gold_price(USD/Ounce)	0
Trade_Balance(INR)	0

dtype: int64

In [12]:
```
data['year-month'] = pd.to_datetime(data['year-month'], format='%Y-%m')
```

In [13]:
```
data.head(3)
```

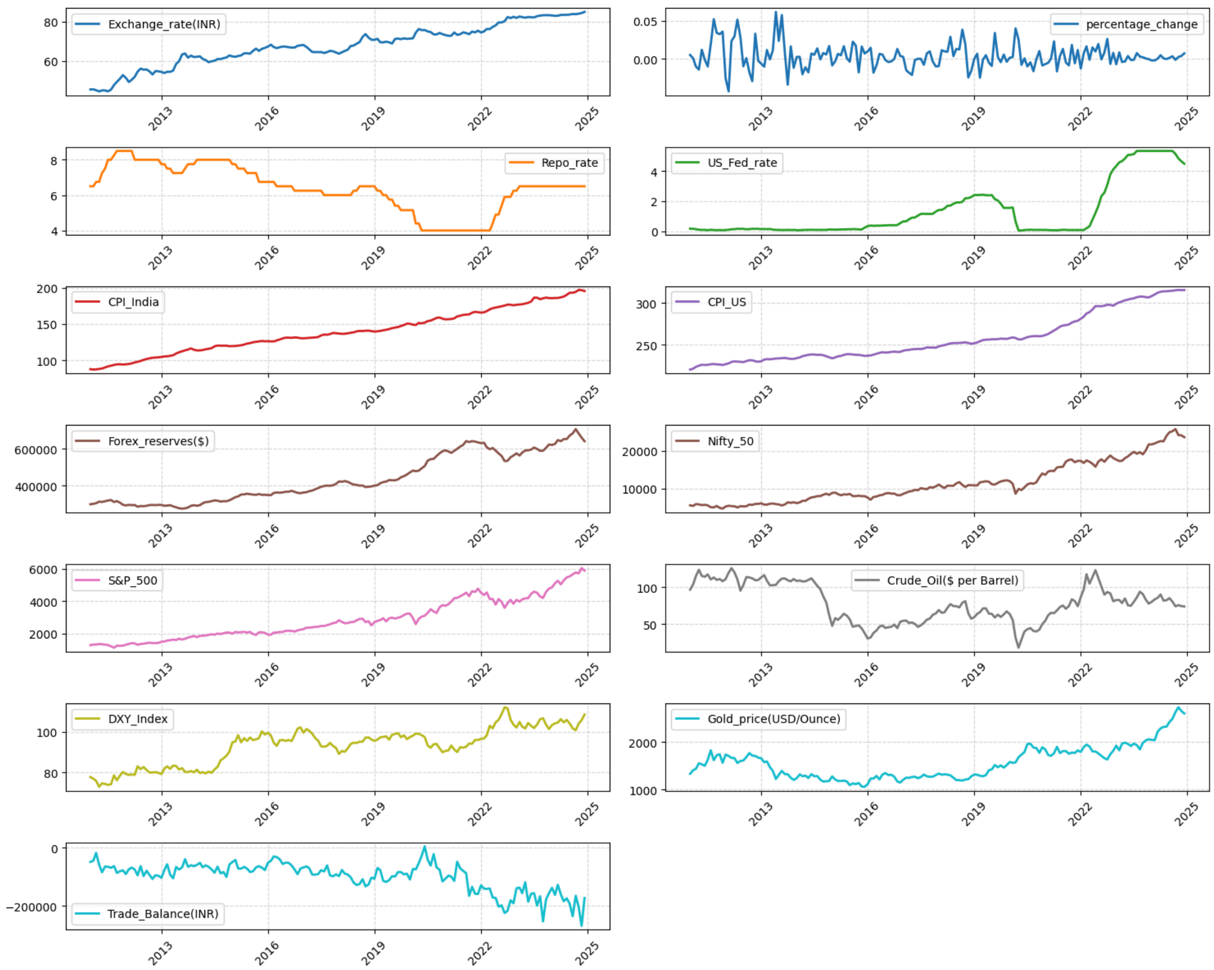
Out[13]:

	year-month	Exchange_rate(INR)	percentage_change	Repo_rate	US_Fed_rate	C
0	2011-01-01	45.3934	0.005240	6.50	0.17	
1	2011-02-01	45.4358	0.000934	6.50	0.16	
2	2011-03-01	44.9914	-0.009781	6.75	0.14	

Trend Visualization

In [16]:

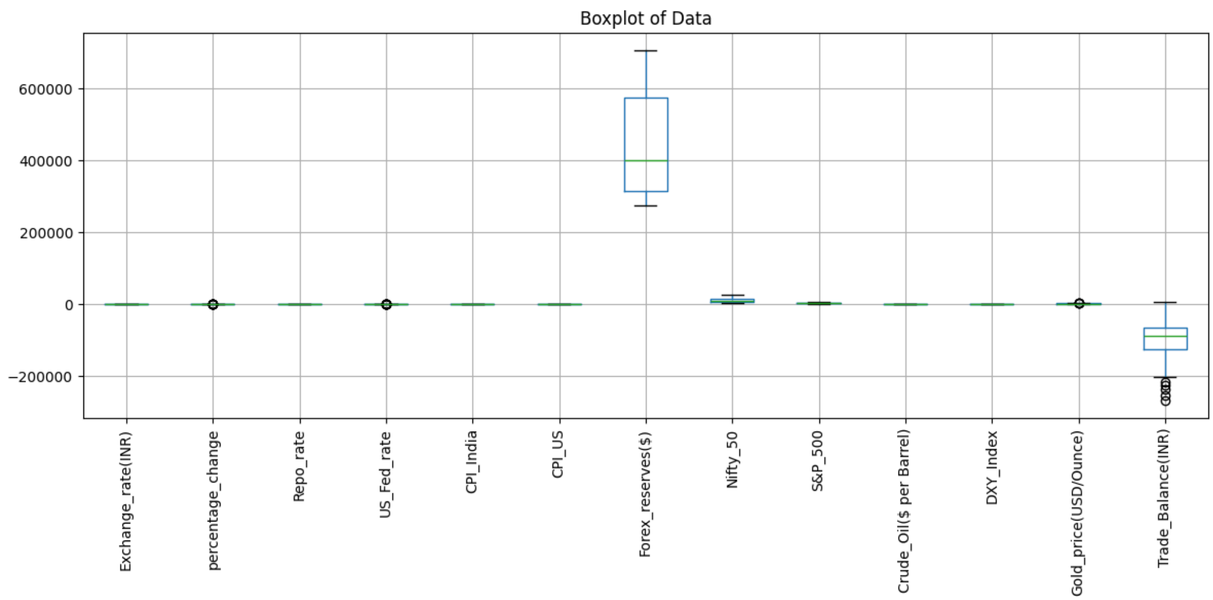
```
import matplotlib.dates as mdates

# Exclude 'Date' column for plotting
columns_to_plot = [col for col in data.columns if col != 'year-month']
num_plots = len(columns_to_plot)

ncols = 2  # Fixed number of columns
nrows = -(-num_plots // ncols)

# Define colors
colors = plt.cm.tab10(np.linspace(0, 1, num_plots))

# Create subplots
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(15, 12))
axes = axes.flatten()

for i, (column, color) in enumerate(zip(columns_to_plot, colors)):
    axes[i].plot(data['year-month'], data[column], label=column, color=color
    axes[i].xaxis.set_major_locator(mdates.YearLocator(base=3))
    axes[i].xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
    axes[i].tick_params(axis='x', rotation=45, labelsize=10)
    axes[i].legend()
    axes[i].grid(True, linestyle='--', alpha=0.5)

# Hide any extra empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



Detecting Outliers

In [17]:

```
plt.figure(figsize=(12, 6))
data.boxplot()
plt.xticks(rotation=90)
plt.title('Boxplot of Data')
plt.tight_layout()
plt.show()
```



In [18]:
```
from sklearn.preprocessing import StandardScaler as Std
```

In [19]:
```
scale= Std()
data1=data.drop('year-month',axis=1)
data_scaled = pd.DataFrame(scale.fit_transform(data1),columns = data1.column
```

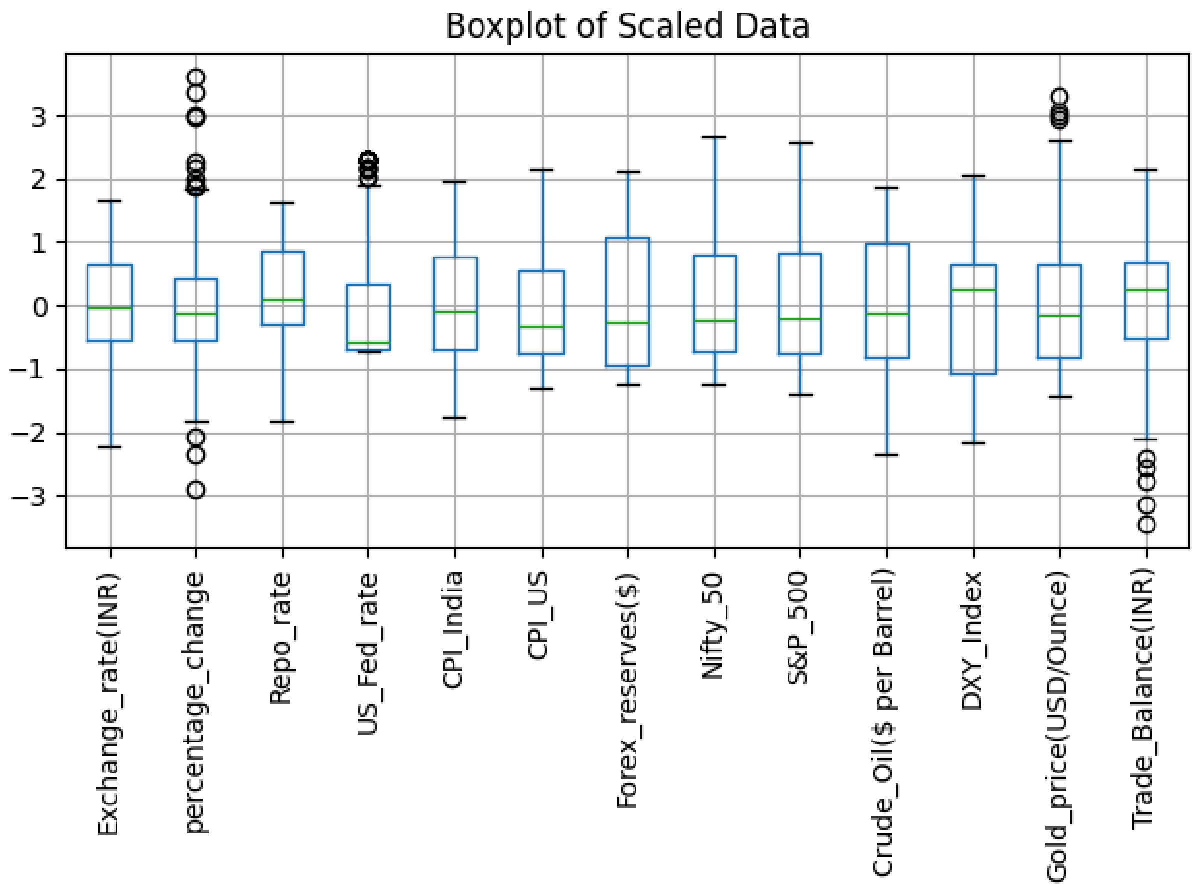
In [20]:
```
data_scaled['year-month'] = data['year-month']
```

In [21]:
```
data_scaled.head(2)
```

Out[21]:

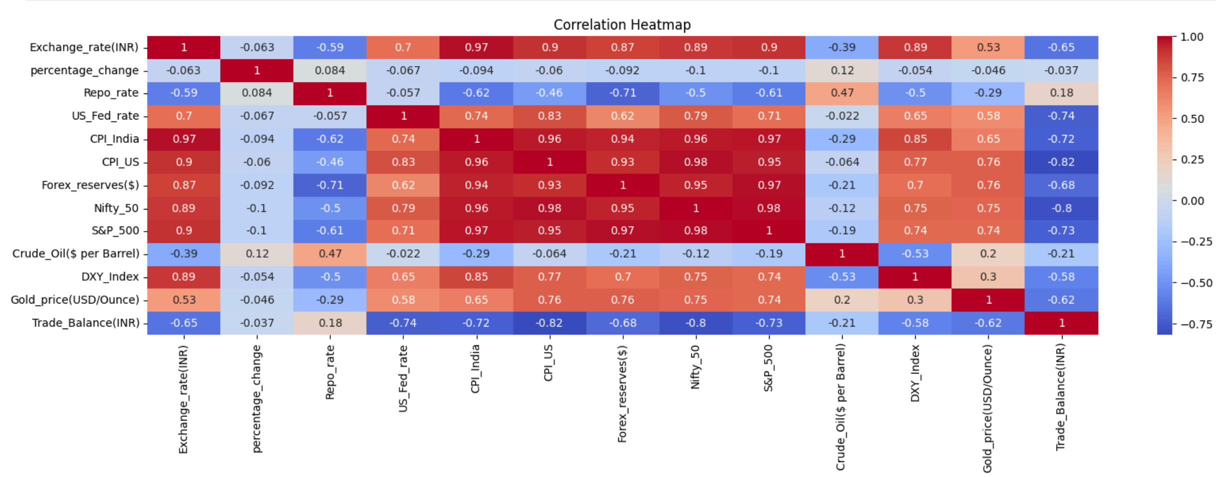
	Exchange_rate(INR)	percentage_change	Repo_rate	US_Fed_rate	CPI_India
0	-2.135116	0.084155	0.091505	-0.648888	-1.746892
1	-2.131059	-0.185715	0.091505	-0.654608	-1.763918

In [22]:
```
data_scaled.boxplot()
plt.xticks(rotation=90)
plt.title('Boxplot of Scaled Data')
plt.tight_layout()
plt.show()
```



Heatmap

In [23]:
```
plt.figure(figsize=(17, 6))
sns.heatmap(data_scaled.drop('year-month',axis=1).corr(),annot=True,cmap='co
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```



In [24]:
```
data_scaled.drop('year-month',axis=1).corr()
```

Out[24]:

	Exchange_rate(INR)	percentage_change	Repo_rate	U
Exchange_rate(INR)	1.000000	-0.063362	-0.592762	
percentage_change	-0.063362	1.000000	0.083694	
Repo_rate	-0.592762	0.083694	1.000000	
US_Fed_rate	0.702220	-0.067038	-0.056836	
CPI_India	0.973519	-0.094263	-0.616243	
CPI_US	0.903386	-0.060356	-0.463375	
Forex_reserves($)	0.870431	-0.091561	-0.707624	
Nifty_50	0.887576	-0.100752	-0.498786	
S&P_500	0.903418	-0.100293	-0.611065	
Crude_Oil($ per Barrel)	-0.387223	0.123805	0.468683	
DXY_Index	0.887231	-0.054203	-0.502247	
Gold_price(USD/Ounce)	0.527930	-0.045828	-0.291754	
Trade_Balance(INR)	-0.646031	-0.037302	0.184251	

Stationarity

In [25]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.11/dist
-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/python3.1
1/dist-packages (from statsmodels) (2.0.2)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.
11/dist-packages (from statsmodels) (1.14.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python
3.11/dist-packages (from statsmodels) (2.2.2)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dis
t-packages (from statsmodels) (1.0.1)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/
dist-packages (from statsmodels) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyth
on3.11/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dis
t-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/d
ist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-pa
ckages (from python-dateutil>=2.8.2->pandas!=2.1.0,>=1.4->statsmodels) (1.1
7.0)
```

In [26]:
```
from statsmodels.tsa.stattools import adfuller
```

In [27]:
```
def adf_test(data):
    for col in data.columns:
        result = adfuller(data[col])
        print(f"p-value for {col}: {result[1]}")
        if result[1] < 0.05:
            print(f"Reject Null Hypothesis for {col}: Data is stationary")
        else:
            print(f"Accept Null Hypothesis for {col}: Data is non-stationary")
        print('\n')
```

In [28]:
```
adf_test(data_scaled.drop('year-month',axis=1))
```

```
p-value for Exchange_rate(INR): 0.5816779895631801
Accept Null Hypothesis for Exchange_rate(INR): Data is non-stationary


p-value for percentage_change: 5.0742202770209245e-06
Reject Null Hypothesis for percentage_change: Data is stationary


p-value for Repo_rate: 0.46254035123754556
Accept Null Hypothesis for Repo_rate: Data is non-stationary


p-value for US_Fed_rate: 0.12139032202388089
Accept Null Hypothesis for US_Fed_rate: Data is non-stationary


p-value for CPI_India: 0.9914236889220505
Accept Null Hypothesis for CPI_India: Data is non-stationary


p-value for CPI_US: 0.969384918962658
Accept Null Hypothesis for CPI_US: Data is non-stationary


p-value for Forex_reserves($): 0.9409804085340132
Accept Null Hypothesis for Forex_reserves($): Data is non-stationary


p-value for Nifty_50: 0.9936246529876204
Accept Null Hypothesis for Nifty_50: Data is non-stationary


p-value for S&P_500: 0.9969149616598285
Accept Null Hypothesis for S&P_500: Data is non-stationary


p-value for Crude_Oil($ per Barrel): 0.270628480216891
Accept Null Hypothesis for Crude_Oil($ per Barrel): Data is non-stationary


p-value for DXY_Index: 0.6497914206487263
Accept Null Hypothesis for DXY_Index: Data is non-stationary


p-value for Gold_price(USD/Ounce): 0.9783678796126527
Accept Null Hypothesis for Gold_price(USD/Ounce): Data is non-stationary


p-value for Trade_Balance(INR): 0.8849008303337087
Accept Null Hypothesis for Trade_Balance(INR): Data is non-stationary
```

In [29]:

```python
non_stationary_cols = ['Exchange_rate(INR)', 'Repo_rate','US_Fed_rate', 'CPI
       'S&P_500', 'Crude_Oil($ per Barrel)', 'DXY_Index','Gold_price(USD/Our
```

```
data_diff1 = data_scaled.copy()
data_diff1[non_stationary_cols] = data_diff1[non_stationary_cols].diff()

data_diff1 = data_diff1.rename(columns=lambda x: f"{x}_diff1" if x in non_st
```

In [30]:
```
data_diff1.dropna(inplace=True)
```

In [31]:
```
adf_test(data_diff1.drop('year-month',axis =1))
```

```
p-value for Exchange_rate(INR)_diff1: 6.286784482235794e-19
Reject Null Hypothesis for Exchange_rate(INR)_diff1: Data is stationary


p-value for percentage_change: 2.440001544698766e-06
Reject Null Hypothesis for percentage_change: Data is stationary


p-value for Repo_rate_diff1: 0.0002968273823382313
Reject Null Hypothesis for Repo_rate_diff1: Data is stationary


p-value for US_Fed_rate_diff1: 0.014822677423495342
Reject Null Hypothesis for US_Fed_rate_diff1: Data is stationary


p-value for CPI_India_diff1: 0.3798427648244248
Accept Null Hypothesis for CPI_India_diff1: Data is non-stationary


p-value for CPI_US_diff1: 0.361873455408575
Accept Null Hypothesis for CPI_US_diff1: Data is non-stationary


p-value for Forex_reserves($)_diff1: 4.137115733555662e-14
Reject Null Hypothesis for Forex_reserves($)_diff1: Data is stationary


p-value for Nifty_50_diff1: 2.191387397588236e-23
Reject Null Hypothesis for Nifty_50_diff1: Data is stationary


p-value for S&P_500_diff1: 1.1019284683741767e-27
Reject Null Hypothesis for S&P_500_diff1: Data is stationary


p-value for Crude_Oil($ per Barrel)_diff1: 3.131730334354299e-16
Reject Null Hypothesis for Crude_Oil($ per Barrel)_diff1: Data is stationary


p-value for DXY_Index_diff1: 3.3511782123737733e-23
Reject Null Hypothesis for DXY_Index_diff1: Data is stationary


p-value for Gold_price(USD/Ounce)_diff1: 1.4836045189851067e-25
Reject Null Hypothesis for Gold_price(USD/Ounce)_diff1: Data is stationary


p-value for Trade_Balance(INR)_diff1: 0.06598684493996661
Accept Null Hypothesis for Trade_Balance(INR)_diff1: Data is non-stationary
```

In [32]:
```
data_diff1.rename(columns={"CPI_US_diff1": "CPI_US", "CPI_India_diff1": "CPI
```

In [33]:
```
non_stationary_cols = ['CPI_India', 'CPI_US','Trade_Balance(INR)']

data_diff2 = data_diff1.copy()
data_diff2[non_stationary_cols] = data_diff2[non_stationary_cols].diff()

data_diff2 = data_diff2.rename(columns=lambda x: f"{x}_diff2" if x in non_st
```

In [34]:
```
data_diff2.dropna(inplace=True)
```

In [35]:
```
adf_test(data_diff2.drop('year-month',axis =1))
```

```
p-value for Exchange_rate(INR)_diff1: 6.603993183438958e-19
Reject Null Hypothesis for Exchange_rate(INR)_diff1: Data is stationary


p-value for percentage_change: 8.515956168753915e-07
Reject Null Hypothesis for percentage_change: Data is stationary


p-value for Repo_rate_diff1: 9.050091245329145e-05
Reject Null Hypothesis for Repo_rate_diff1: Data is stationary


p-value for US_Fed_rate_diff1: 0.01548999731904332
Reject Null Hypothesis for US_Fed_rate_diff1: Data is stationary


p-value for CPI_India_diff2: 7.293954004350278e-13
Reject Null Hypothesis for CPI_India_diff2: Data is stationary


p-value for CPI_US_diff2: 2.1927283667287613e-12
Reject Null Hypothesis for CPI_US_diff2: Data is stationary


p-value for Forex_reserves($)_diff1: 4.816962983676830-14
Reject Null Hypothesis for Forex_reserves($)_diff1: Data is stationary


p-value for Nifty_50_diff1: 2.9175254578719215e-23
Reject Null Hypothesis for Nifty_50_diff1: Data is stationary


p-value for S&P_500_diff1: 1.2772565828444195e-27
Reject Null Hypothesis for S&P_500_diff1: Data is stationary


p-value for Crude_Oil($ per Barrel)_diff1: 1.9493516632924197e-16
Reject Null Hypothesis for Crude_Oil($ per Barrel)_diff1: Data is stationary


p-value for DXY_Index_diff1: 3.4282333181913184e-23
Reject Null Hypothesis for DXY_Index_diff1: Data is stationary


p-value for Gold_price(USD/Ounce)_diff1: 1.7917577624482506e-25
Reject Null Hypothesis for Gold_price(USD/Ounce)_diff1: Data is stationary


p-value for Trade_Balance(INR)_diff2: 8.955701371386637e-11
Reject Null Hypothesis for Trade_Balance(INR)_diff2: Data is stationary
```

In [36]:

```python
data_diff2.set_index('year-month', inplace=True)
```

Model Prediction

In [37]:
```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_percentage_error, mean_squared_err
from statsmodels.tsa.api import VAR
```

In [38]:
```
train_size = int(len(data_diff2) * 0.8)
train_data, test_data = data_diff2.iloc[:train_size], data_diff2.iloc[train_
```

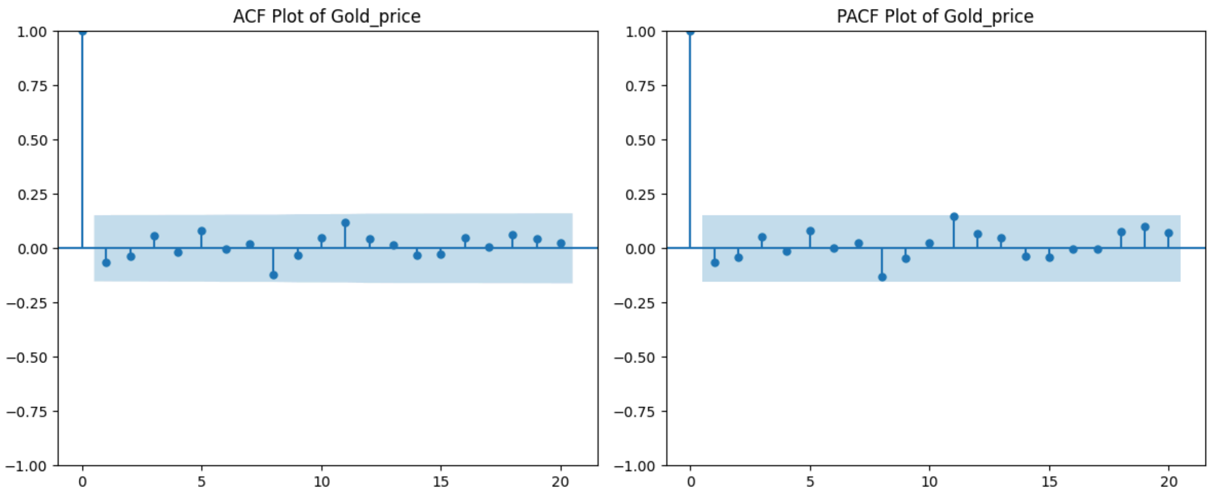
Gold Price Forecasting using ARIMA and VAR model

In [39]:
```
target = data_diff2['Gold_price(USD/Ounce)_diff1']

max_lag = 20

acf_vals = acf(target, nlags=max_lag)
pacf_vals = pacf(target, nlags=max_lag)

lags = np.arange(max_lag + 1)
acf_pacf_df = pd.DataFrame({'Lag': lags, 'ACF': acf_vals, 'PACF': pacf_vals}

print(acf_pacf_df)
```

```
    Lag       ACF      PACF
0     0  1.000000  1.000000
1     1 -0.066156 -0.066557
2     2 -0.037482 -0.042558
3     3  0.058381  0.054329
4     4 -0.019021 -0.013481
5     5  0.080239  0.085705
6     6 -0.003574  0.002910
7     7  0.018194  0.027856
8     8 -0.123528 -0.139159
9     9 -0.032004 -0.048199
10   10  0.050492  0.027994
11   11  0.121440  0.156399
12   12  0.042959  0.072128
13   13  0.014041  0.054636
14   14 -0.030305 -0.042436
15   15 -0.025937 -0.046189
16   16  0.048908 -0.003018
17   17  0.005861 -0.005053
18   18  0.061371  0.087043
19   19  0.043070  0.117159
20   20  0.023896  0.085537
```

In [40]:

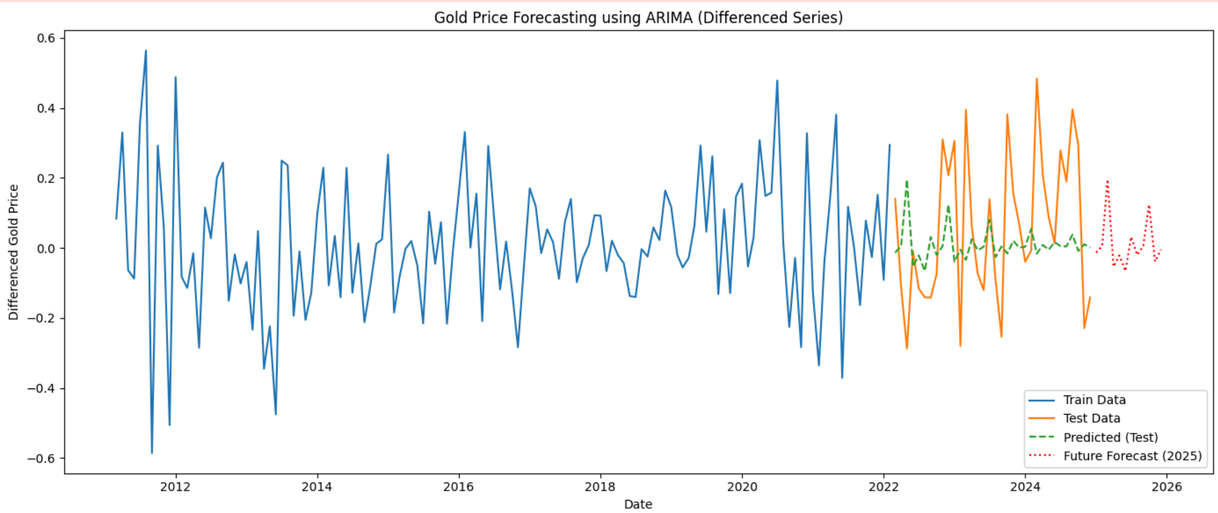
```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
plot_acf(data_diff2['Gold_price(USD/Ounce)_diff1'], lags=20, ax=axes[0])
axes[0].set_title('ACF Plot of Gold_price')

plot_pacf(data_diff2['Gold_price(USD/Ounce)_diff1'], lags=20, ax=axes[1])
axes[1].set_title('PACF Plot of Gold_price')
plt.tight_layout()
plt.show()
```



In [43]:

```
model = ARIMA(train_data['Gold_price(USD/Ounce)_diff1'], order=(8, 0, 8))
fitted_model = model.fit()

# Forecast on test_data
test_forecast = fitted_model.forecast(steps=len(test_data))

# Forecast into the future
future_steps = 12
future_forecast = fitted_model.forecast(steps=future_steps)

# Create datetime index for future forecasts
last_date = test_data.index[-1]
future_index = pd.date_range(start=last_date + pd.DateOffset(months=1), peri

# Plotting
plt.figure(figsize=(14,6))
plt.plot(train_data.index, train_data['Gold_price(USD/Ounce)_diff1'], label=
plt.plot(test_data.index, test_data['Gold_price(USD/Ounce)_diff1'], label='T
plt.plot(test_data.index, test_forecast, label='Predicted (Test)', linestyle
plt.plot(future_index, future_forecast, label='Future Forecast (2025)', line
plt.legend()
plt.title("Gold Price Forecasting using ARIMA (Differenced Series)")
plt.xlabel("Date")
plt.ylabel("Differenced Gold Price")
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/base/model.py:607: Conve
rgenceWarning: Maximum Likelihood optimization failed to converge. Check mle
_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```



In [44]:
```python
rmse = np.sqrt(mean_squared_error(test_data['Gold_price(USD/Ounce)_diff1'],
mape = mean_absolute_percentage_error(test_data['Gold_price(USD/Ounce)_diff1
aic = fitted_model.aic
n = fitted_model.nobs
k = fitted_model.df_model

# AICc formula
aicc = aic + (2 * k * (k + 1)) / (n - k - 1)

print(f"RMSE: {rmse}")
print(f"MAPE: {mape * 100:.2f}%")
print(f"AIC: {fitted_model.aic}")
print(f"BIC: {fitted_model.bic}")
print(f"AICc: {aicc}")
```

```
RMSE: 0.22900055848713705
MAPE: 125.43%
AIC: -40.710420587988494
BIC: 11.180014018566169
AICc: -34.65732324285575
```

In [45]:
```python
# to get optimal lags
model = VAR(train_data)
```

```python
lag_selection = model.select_order(maxlags=8)

print(lag_selection.summary())

print("Best lag (AIC):", lag_selection.aic)
print("Best lag (BIC):", lag_selection.bic)
print("Best lag (HQIC):", lag_selection.hqic)
```

```
 VAR Order Selection (* highlights the minimums)
==================================================
      AIC         BIC         FPE         HQIC
--------------------------------------------------
0      -61.56     -61.27*   1.838e-27      -61.44
1      -62.22      -58.08   9.642e-28      -60.54
2      -62.25      -54.27   1.009e-27      -59.01
3      -61.92      -50.10   1.744e-27      -57.12
4      -61.89      -46.22   2.841e-27      -55.52
5      -61.98      -42.46   5.937e-27      -54.05
6      -62.97      -39.62   9.180e-27      -53.48
7      -66.66      -39.45   2.851e-27      -55.61
8     -74.48*      -43.43  1.403e-28*     -61.87*
--------------------------------------------------
Best lag (AIC): 8
Best lag (BIC): 0
Best lag (HQIC): 8
```

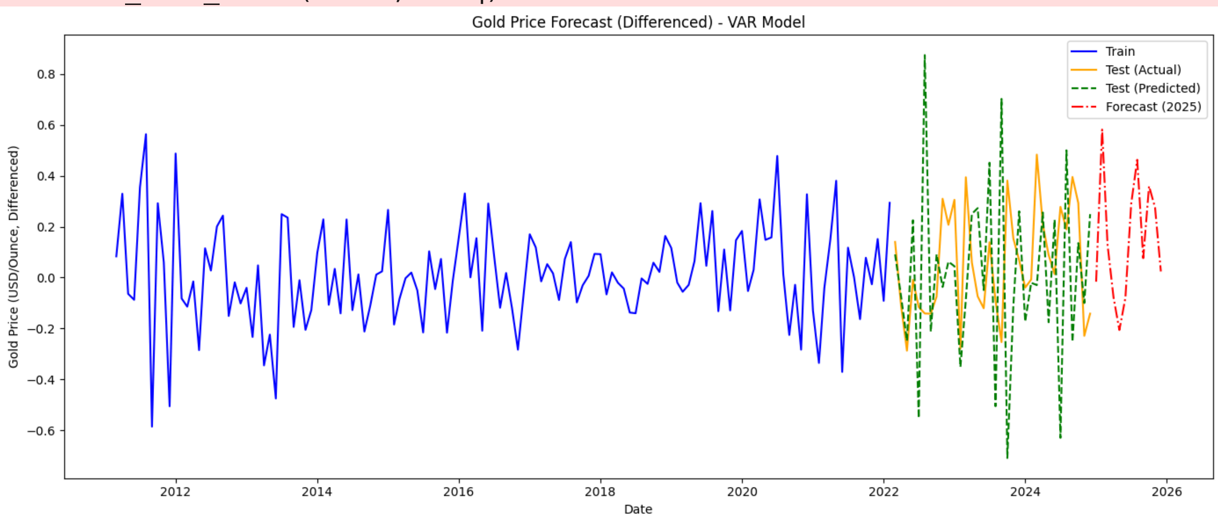
```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
```

In [47]:

```python
model = VAR(train_data)
lag_order = model.select_order(maxlags=8)
selected_lag = lag_order.aic
fitted_model = model.fit(selected_lag)

# Forecast on test_data
n_test = len(test_data)
test_forecast = fitted_model.forecast(train_data.values[-selected_lag:], ste
test_forecast_df = pd.DataFrame(test_forecast, index=test_data.index, column

# Refitting model on full data for future forecasting
full_data = pd.concat([train_data, test_data])
model_full = VAR(full_data)
fitted_full_model = model_full.fit(selected_lag)

# Forecast future values (2025–2028)
future_steps = 12
last_input = full_data.values[-selected_lag:]
future_forecast = fitted_full_model.forecast(last_input, steps=future_steps)

# Create datetime index for future forecast
last_date = full_data.index[-1]
future_index = pd.date_range(start=last_date + pd.DateOffset(months=1), peri
future_df = pd.DataFrame(future_forecast, index=future_index, columns=train_

# Plotting
```

```
plt.figure(figsize=(14, 6))
plt.plot(train_data['Gold_price(USD/Ounce)_diff1'], label='Train', color='bl
plt.plot(test_data['Gold_price(USD/Ounce)_diff1'], label='Test (Actual)', co
plt.plot(test_forecast_df['Gold_price(USD/Ounce)_diff1'], label='Test (Predi
plt.plot(future_df['Gold_price(USD/Ounce)_diff1'], label='Forecast (2025)',
plt.title('Gold Price Forecast (Differenced) - VAR Model')
plt.xlabel('Date')
plt.ylabel('Gold Price (USD/Ounce, Differenced)')
plt.legend()
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
```



In [48]:
```
rmse = np.sqrt(mean_squared_error(test_data['Gold_price(USD/Ounce)_diff1'],
mape = mean_absolute_percentage_error(test_data['Gold_price(USD/Ounce)_diff1

# AICc calculation (corrected AIC)
aic = fitted_model.aic
k = fitted_model.neqs
k = fitted_model.df_model
aicc = aic + (2 * k * (k + 1)) / (n - k - 1)

# Print results
print(f"RMSE: {rmse}")
print(f"MAPE: {mape:.2f}%")
print(f"AIC  : {fitted_model.aic}")
print(f"BIC  : {fitted_model.bic}")
print(f"AICc : {aicc}")
```

```
RMSE: 0.431708119250338l4
MAPE: 314.06%
AIC  : -74.47821197311349
BIC  : -43.432370545283845
AICc : 781.6756341807327
```

In [60]:
```python
from statsmodels.tsa.stattools import grangercausalitytests

def granger_causality_matrix(data, target_col, variables, max_lag=8):
    results = {}
    for var in variables:
        if var == target_col:
            continue
        p_values = []
        for lag in range(1, max_lag + 1):
            test_result = grangercausalitytests(data[[target_col, var]], max
            p_val = test_result[lag][0]['ssr_ftest'][1]
            p_values.append(round(p_val, 4))
        results[var] = p_values
    df_results = pd.DataFrame(results, index=[f'Lag {i}' for i in range(1, m
    return df_results.T

target = 'Gold_price(USD/Ounce)_diff1'
variables = data_diff2.columns
granger_table = granger_causality_matrix(data_diff2, target_col=target, vari

# Displaying the table
print("Granger Causality p-values (testing if column causes Gold Price):")
print(granger_table)
```

```
Granger Causality p-values (testing if column causes Gold Price):
                                Lag 1   Lag 2   Lag 3   Lag 4   Lag 5   Lag 
6  \
Exchange_rate(INR)_diff1       0.4107  0.5859  0.7037  0.7700  0.7536  0.695
2
percentage_change              0.5669  0.6497  0.7492  0.8097  0.8289  0.716
0
Repo_rate_diff1                0.0157  0.0017  0.0043  0.0030  0.0014  0.004
0
US_Fed_rate_diff1              0.4610  0.7122  0.6614  0.8066  0.1699  0.184
9
CPI_India_diff2                0.8566  0.2888  0.3812  0.4706  0.4918  0.492
3
CPI_US_diff2                   0.8589  0.9101  0.9369  0.1893  0.1311  0.234
5
Forex_reserves($)_diff1        0.0519  0.1109  0.2282  0.2641  0.4515  0.497
1
Nifty_50_diff1                 0.6604  0.6150  0.6135  0.7545  0.8244  0.862
4
S&P_500_diff1                  0.7455  0.9285  0.9551  0.5878  0.7164  0.765
7
Crude_Oil($ per Barrel)_diff1  0.3782  0.2774  0.1509  0.2239  0.1823  0.184
1
DXY_Index_diff1                0.0447  0.1195  0.1227  0.0754  0.0565  0.094
7
Trade_Balance(INR)_diff2       0.3599  0.1175  0.2247  0.3599  0.3520  0.444
2

                                Lag 7   Lag 8
Exchange_rate(INR)_diff1       0.4648  0.3135
percentage_change              0.4018  0.2538
Repo_rate_diff1                0.0318  0.0562
US_Fed_rate_diff1              0.1725  0.2741
CPI_India_diff2                0.1885  0.2653
CPI_US_diff2                   0.1933  0.0591
Forex_reserves($)_diff1        0.7206  0.8323
Nifty_50_diff1                 0.8711  0.3801
S&P_500_diff1                  0.4241  0.1382
Crude_Oil($ per Barrel)_diff1  0.2847  0.4911
DXY_Index_diff1                0.1766  0.2156
Trade_Balance(INR)_diff2       0.4422  0.6015
```

Final Interpreation of future gold Price

According to AICc, we can see the VAR model has less value than the Arima Model. Hence We choose VAR model as the best model in this case.

The Granger Causality test shows that the Repo Rate significantly predicts gold prices across multiple lags. Other variables like the DXY Index and CPI_US show weak or inconsistent significance, making the Repo Rate the strongest predictor among the tested indicators.

Reason: Repo rate significantly predicts gold prices because it influences inflation, interest rates, liquidity, and currency value — all of which directly affect investor demand for gold

In [ ]:

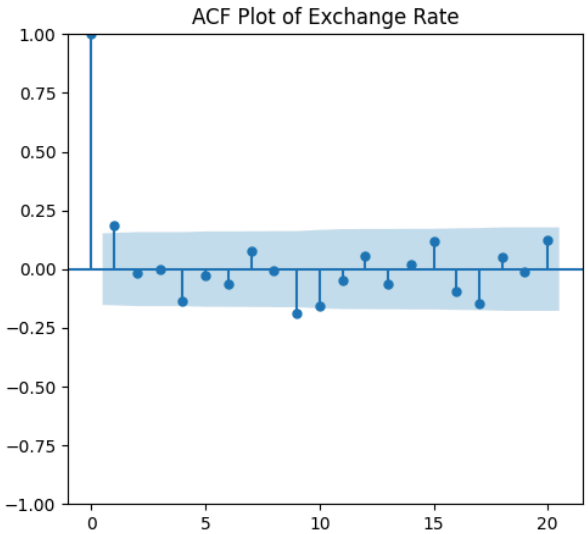
Exchange Rate Forecasting using ARIMA and VAR model

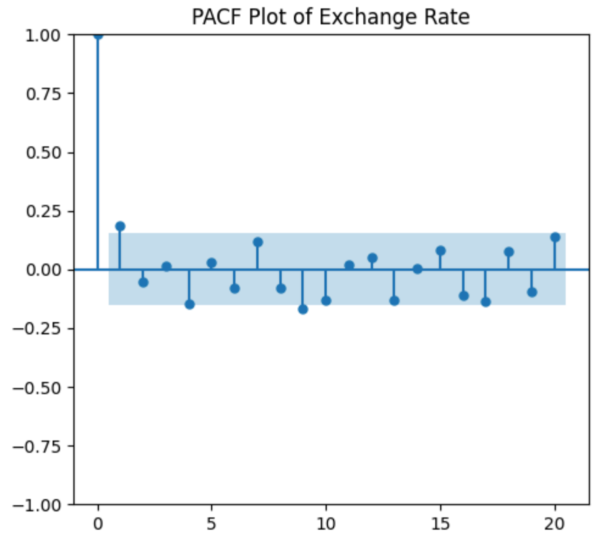
In [53]:
```
target = data_diff2['Exchange_rate(INR)_diff1']

max_lag = 20

acf_vals = acf(target, nlags=max_lag)
pacf_vals = pacf(target, nlags=max_lag)

lags = np.arange(max_lag + 1)
acf_pacf_df = pd.DataFrame({'Lag': lags, 'ACF': acf_vals, 'PACF': pacf_vals}

print(acf_pacf_df)
```

```
    Lag       ACF      PACF
0     0  1.000000  1.000000
1     1  0.185743  0.186868
2     2 -0.017349 -0.054379
3     3 -0.000726  0.013388
4     4 -0.138381 -0.150880
5     5 -0.027534  0.030680
6     6 -0.063334 -0.082890
7     7  0.075644  0.121626
8     8 -0.008141 -0.084714
9     9 -0.187792 -0.177888
10   10 -0.158788 -0.140292
11   11 -0.048477  0.023880
12   12  0.054084  0.053127
13   13 -0.064529 -0.141109
14   14  0.018420  0.002500
15   15  0.117945  0.091958
16   16 -0.092701 -0.122034
17   17 -0.144055 -0.156961
18   18  0.050773  0.091360
19   19 -0.013493 -0.108182
20   20  0.121381  0.158380
```

In [54]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
plot_acf(data_diff2['Exchange_rate(INR)_diff1'], lags=20, ax=axes[0])
axes[0].set_title('ACF Plot of Exchange Rate')

plot_pacf(data_diff2['Exchange_rate(INR)_diff1'], lags=20, ax=axes[1])
axes[1].set_title('PACF Plot of Exchange Rate')
```

Out[54]:
```
Text(0.5, 1.0, 'PACF Plot of Exchange Rate')
```

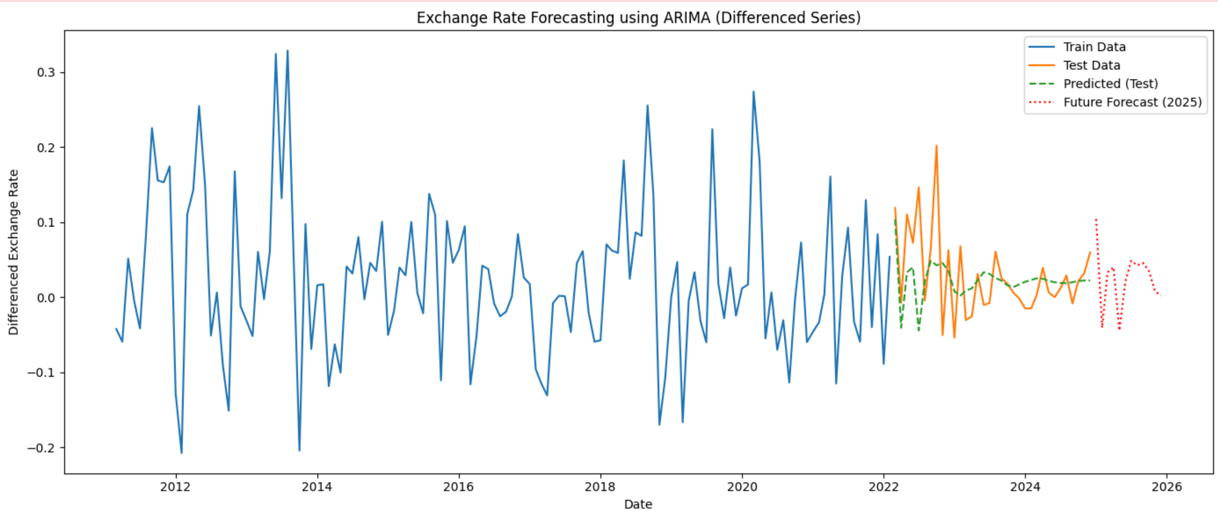




In [56]:

```
model = ARIMA(train_data['Exchange_rate(INR)_diff1'], order=(4, 0, 9))
fitted_model = model.fit()

# Forecast on test period
test_forecast = fitted_model.forecast(steps=len(test_data))

# Forecast into the future
future_steps = 12
future_forecast = fitted_model.forecast(steps=future_steps)

# Create datetime index for future forecasts
last_date = test_data.index[-1]
future_index = pd.date_range(start=last_date + pd.DateOffset(months=1), peri

# Plotting
plt.figure(figsize=(14,6))
plt.plot(train_data.index, train_data['Exchange_rate(INR)_diff1'], label='Tr
plt.plot(test_data.index, test_data['Exchange_rate(INR)_diff1'], label='Test
plt.plot(test_data.index, test_forecast, label='Predicted (Test)', linestyle
plt.plot(future_index, future_forecast, label='Future Forecast (2025)', line
plt.legend()
plt.title("Exchange Rate Forecasting using ARIMA (Differenced Series)")
plt.xlabel("Date")
plt.ylabel("Differenced Exchange Rate")
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/base/model.py:607: Conve
rgenceWarning: Maximum Likelihood optimization failed to converge. Check mle
_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```



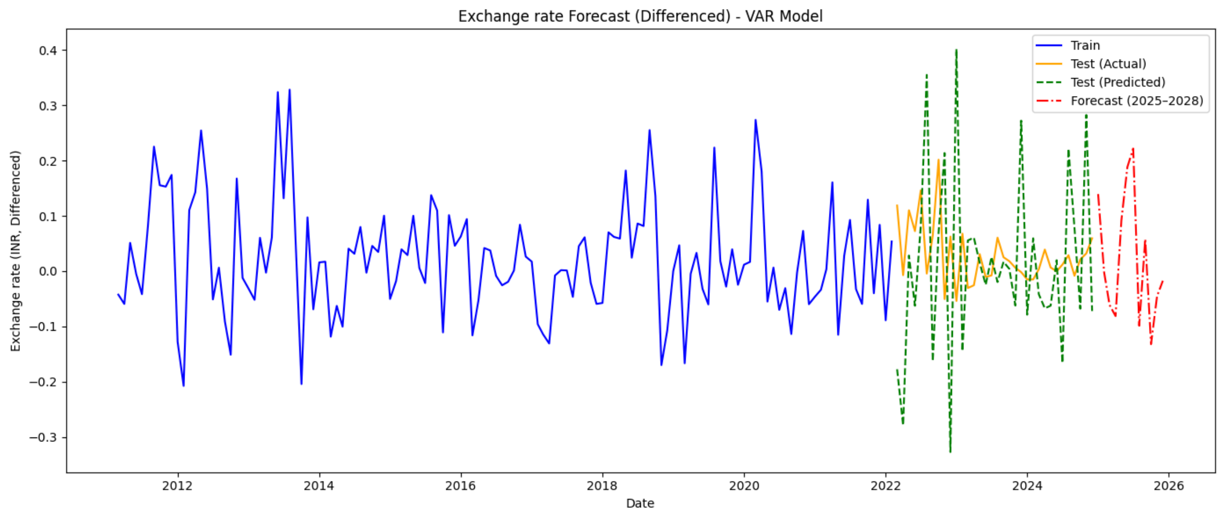
In [57]:
```
rmse = np.sqrt(mean_squared_error(test_data['Exchange_rate(INR)_diff1'], tes
mape = mean_absolute_percentage_error(test_data['Exchange_rate(INR)_diff1'],
aic = fitted_model.aic
n = fitted_model.nobs
k = fitted_model.df_model

# AICc formula
aicc = aic + (2 * k * (k + 1)) / (n - k - 1)

print(f"RMSE: {rmse}")
print(f"MAPE: {mape * 100:.2f}%")
print(f"AIC: {fitted_model.aic}")
print(f"BIC: {fitted_model.bic}")
print(f"AICc: {aicc}")
```

```
RMSE: 0.055218911584419966
MAPE: 382.17%
AIC: -224.19878423069198
BIC: -180.95675539189642
AICc: -220.06085319620922
```

In [58]:
```
model = VAR(train_data)
lag_order = model.select_order(maxlags=8)
selected_lag = lag_order.aic
```

```
fitted_model = model.fit(selected_lag)

# Forecast on test_data
n_test = len(test_data)
test_forecast = fitted_model.forecast(train_data.values[-selected_lag:], ste
test_forecast_df = pd.DataFrame(test_forecast, index=test_data.index, column

# Refit model on full data for future forecasting
full_data = pd.concat([train_data, test_data])
model_full = VAR(full_data)
fitted_full_model = model_full.fit(selected_lag)

# Forecast future values
future_steps = 12
last_input = full_data.values[-selected_lag:]
future_forecast = fitted_full_model.forecast(last_input, steps=future_steps)

# Create datetime index for future forecast
last_date = full_data.index[-1]
future_index = pd.date_range(start=last_date + pd.DateOffset(months=1), peri
future_df = pd.DataFrame(future_forecast, index=future_index, columns=train_

# Plotting
plt.figure(figsize=(14, 6))
plt.plot(train_data['Exchange_rate(INR)_diff1'], label='Train', color='blue'
plt.plot(test_data['Exchange_rate(INR)_diff1'], label='Test (Actual)', color
plt.plot(test_forecast_df['Exchange_rate(INR)_diff1'], label='Test (Predicte
plt.plot(future_df['Exchange_rate(INR)_diff1'], label='Forecast (2025–2028)'
plt.title('Exchange rate Forecast (Differenced) - VAR Model')
plt.xlabel('Date')
plt.ylabel('Exchange rate (INR, Differenced)')
plt.legend()
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
```



In [59]:
```
rmse = np.sqrt(mean_squared_error(test_data['Exchange_rate(INR)_diff1'], tes
mape = mean_absolute_percentage_error(test_data['Exchange_rate(INR)_diff1'],

# AICc calculation (corrected AIC)
n = fitted_model.nobs
k = fitted_model.df_model
aic = fitted_model.aic
aicc = aic + (2 * k * (k + 1)) / (n - k - 1)

# Print results
print(f"RMSE: {rmse}")
print(f"MAPE: {mape:.2f}%")
print(f"AIC  : {fitted_model.aic}")
print(f"BIC  : {fitted_model.bic}")
print(f"AICc : {aicc}")
```

```
RMSE: 0.1847356386615379
MAPE: 13.98%
AIC  : -74.47821197311349
BIC  : -43.432370545283845
AICc : 1162.1884546935532
```

In [61]:
```
def granger_causality_matrix(data, target_col, variables, max_lag=8):
    results = {}
    for var in variables:
        if var == target_col:
            continue
        p_values = []
        for lag in range(1, max_lag + 1):
            test_result = grangercausalitytests(data[[target_col, var]], max
            p_val = test_result[lag][0]['ssr_ftest'][1]
            p_values.append(round(p_val, 4))
        results[var] = p_values
    df_results = pd.DataFrame(results, index=[f'Lag {i}' for i in range(1, m
    return df_results.T

target = 'Exchange_rate(INR)_diff1'
variables = data_diff2.columns
granger_table = granger_causality_matrix(data_diff2, target_col=target, vari
```

```python
# Displaying the table
print("Granger Causality p-values (testing if column causes Exchange rate):"
print(granger_table)
```

```
Granger Causality p-values (testing if column causes Exchange rate):
                                Lag 1   Lag 2   Lag 3   Lag 4   Lag 5   Lag
6  \
percentage_change              0.0149  0.0424  0.0765  0.0097  0.0106  0.005
0
Repo_rate_diff1                0.9114  0.6768  0.8590  0.7235  0.7671  0.852
7
US_Fed_rate_diff1              0.9763  0.5875  0.7245  0.8300  0.6270  0.685
3
CPI_India_diff2                0.4933  0.6820  0.8597  0.8949  0.8580  0.949
2
CPI_US_diff2                   0.9413  0.8246  0.6780  0.7707  0.7909  0.816
0
Forex_reserves($)_diff1        0.0181  0.0502  0.0368  0.0467  0.0722  0.041
7
Nifty_50_diff1                 0.0007  0.0011  0.0013  0.0052  0.0097  0.019
5
S&P_500_diff1                  0.0013  0.0062  0.0074  0.0142  0.0276  0.037
6
Crude_Oil($ per Barrel)_diff1  0.7549  0.7687  0.8955  0.2230  0.2104  0.188
3
DXY_Index_diff1                0.0037  0.0143  0.0071  0.0207  0.0130  0.000
7
Gold_price(USD/Ounce)_diff1    0.1177  0.0982  0.1097  0.1614  0.2551  0.194
3
Trade_Balance(INR)_diff2       0.2223  0.4648  0.5349  0.6732  0.7777  0.895
0

                                Lag 7   Lag 8
percentage_change              0.0114  0.0186
Repo_rate_diff1                0.6998  0.7755
US_Fed_rate_diff1              0.3963  0.5491
CPI_India_diff2                0.8585  0.9302
CPI_US_diff2                   0.9592  0.9460
Forex_reserves($)_diff1        0.0620  0.0974
Nifty_50_diff1                 0.0387  0.0366
S&P_500_diff1                  0.0637  0.0638
Crude_Oil($ per Barrel)_diff1  0.2172  0.3715
DXY_Index_diff1                0.0007  0.0012
Gold_price(USD/Ounce)_diff1    0.0535  0.0448
Trade_Balance(INR)_diff2       0.8395  0.2636
```

Final Interpreation of future Exchange Rate

According to AICc, we can see the ARIMA model has less Value than the VAR Model. Hence We choose ARIMA model as the best model in this case.

The Granger causality test shows that percentage change, forex reserves, Nifty 50, S&P 500, and the DXY Index significantly influence the exchange rate (INR), with consistently low p-values across multiple lags. Gold prices also show a delayed impact, being marginally significant at lag 8. In contrast, variables like the repo rate, US Fed rate, CPI (India and US), crude oil, and trade balance do not exhibit significant short-term predictive power for the exchange rate, as their p-values remain high across all lags.

```
In [ ]:
```