MTH517 Course Project

Finding Relation between Currency Exchange Rates over a Period of Time

Abhas Goyal 18817013
Shubham Indoliya 190830
Lakshay Rastogi 180378

1) Abstract

The present article compares the performance of two well-known time series models, viz., Auto-Regressive (AR) and Vector Auto-Regressive (VAR). We illustrate the differences between these models by using a dataset on exchange rates, and analyze which model among the 2 is able to fit the data in a better way and make more accurate predictions.

2) Introduction

In this project, we study 3 different exchange rates, viz., **USD-INR**, **JPY-INR** and **EUR-INR.** Exchange rate markets are some of the most heavily traded markets, with more than $3.98 TRILLION traded on the foreign exchange market each day. That's about 53 times more than the New York stock exchange, where a measly $74 billion changes hands daily. Some of the factors on which exchange rates depend include inflation rates, interest rates, government debt and the political stability and performance of a country.

Being able to make accurate predictions in the exchange rate market enables various stakeholders to minimize their risk and make informed decisions.

We try to fit time series data of exchange rates on 2 models, Auto-Regressive (AR) and Vector Auto-Regressive (VAR), and find which one performs in terms of generating more accurate predictions.

3) Concepts

3.1 TRAIN-TEST SPLIT

NEED: Forecasting is difficult! A test set is needed to estimate how well the model works on new data.

HOW: The test set is kept as large as the maximum forecast horizon. We use an 80:20 split in our analysis

3.2 METRIC FOR EVALUATING PREDICTIONS

ROOT MEAN SQUARE ERROR: Good or bad i.e. acceptability depends on the data and the problem being solved.

$$RMSE = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(y_{T+l} - \widehat{y_{T+l}})^2}$$

where,
T is the last observation period and l is the lag.

3.3 STATIONARITY

Defined as the absence of trend and seasonal component in the data - where mean, variance and covariance are not functions of time but of lag values itself.

3.4 AUGMENTED DICKIE FULLER TEST

One can never be certain about the stationarity/non-stationarity of the given time series just by looking at the data. Augmented Dickie Fuller Test is used to quantitatively check for the stationarity of the time series.
For this test the null hypothesis stated that $\phi = 1$ (this is also called a unit test), which states that the data is non-stationary.
The test returns many statistics, but we focus on the p-value. A small p-value ($p < 0.05$) indicates strong evidence against the null hypothesis, and thus choose that the data is stationary.

3.5 DIFFERENCING

This method eliminates non-stationarity from the data. By every step of differencing we lose one row of data, and therefore the size of the data should be kept in mind while performing differencing. In our case we had sufficient data hence losing one row did not pose a problem.

3.6 INVERSE TRANSFORMATION

Since we have differenced the original time series, the forecasted values in our models also represent the difference values. We need to roll back these forecasts to the original form to get our predictions.

To roll back a first-order difference we take the most recent value on the training side of the original series, i.e., the latest training time point and add it to a cumulative sum, also called the prefix sum of the forecasted values.

3.7 MODELS

1. AR(p) Auto-Regressive Model: A regression model that utilizes the dependent relationship between a current observation and observations over a previous period.

This is run against a set of lagged values of order P, described as follows

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots\ldots\ldots + \phi_p y_{t-p} + \varepsilon_t$$

where,
c is a constant,
ϕ_1, ϕ_2 are lag coefficients up to order p,
ε_t is drawn from a white noise process.

Generally, the higher order you use the more accurate your results are going to be because you're using more historical information but there will be a limit because if you go too far back then it may lead to overfitting and our model may base it's predictions on the noise present in the data which is actually not very relevant.

2. VAR(p), Vector Auto-Regressive Model: Used when there's a relationship or an influence factor within two or more variables. A K-dimensional VAR model of order p, denoted VAR(p), considers each variable y_k in the system.

For example, The system of equations for a 2-dimensional VAR(1) model is:

$$y_{1,t} = c_1 + \phi_{11,1} y_{1,t-1} + \phi_{12,1} y_{2,t-1} + \varepsilon_{1,t}$$
$$y_{2,t} = c_2 + \phi_{21,1} y_{1,t-1} + \phi_{22,1} y_{2,t-1} + \varepsilon_{2,t}$$

where,
the coefficient $\phi_{ii,l}$ captures the influence of the lth lag of variable y_i on itself,
the coefficient $\phi_{ij,l}$ captures the influence of the *l*th lag of variable y_j on y_i,
and $\varepsilon_{1,t}$ and $\varepsilon_{2,t}$ are drawn from white noise processes that may be correlated

3.8 CHOOSING P

1D GRID SEARCH: Involves running a bunch of
models on different values of p and then comparing the models on some evaluation metric - like the one we have used for this project, the AIC score. We pick the value for our hyperparameter "p" corresponding to the lowest AIC score.

3.9 AIC SCORE

The AIC evaluates a collection of models and estimates the quality of each model relative to the others. Penalties are provided for the number of parameters used in an effort to thwart over-fitting. The lower the AIC, the better the model should be at forecasting.

Let k be the number of estimated parameters in the model. Let $\widehat{L}$ be the maximum value of the likelihood function for the model. Then the AIC value of the model is:

$$AIC = 2k - 2ln(\widehat{L})$$

4) Dataset

We used https://www.google.com/finance/ for getting the data of the currency rates for weekly rates for the last years. We used the weekly rates for the last 7 years.

5) Workflow

We perform our analysis and forecasting in Python 3.0. The libraries we use are pandas, matplotlib and statsmodels. Statsmodels has functions to fit the AR and VAR models which we use in the project.

1) Import the relevant tools and libraries

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.api import VAR
from statsmodels.tsa.ar_model import AR,ARResults,AutoReg
import warnings
warnings.filterwarnings("ignore")
```

2) Import Data from the Dataset

```
rates = pd.read_csv('usdinr.csv')
rates = rates[3493:5682] # Decide the amount of data we need.
```

```
rates = rates.iloc[::7,:] # Getting weekly data
rates.info() # to check for any null values present
```

3) Scale the data so that it can be visualized properly

```
scaled_rates = rates.copy(deep=True)
for i in range(2,len(scaled_rates)-2):
    for j in range(1,4):
        scaled_rates.iloc[i,j] = sum(scaled_rates.iloc[(i-2):(i+3),j])/5
for col in scaled_rates:
    if col != 'Date':
        scaled_rates[col] = scaled_rates[col]*100/scaled_rates[col].mean()
```

4) Plot the scaled dataset to visualize the data:

```
plt.figure(figsize = (16,10))
plt.title("Scaled Exchange Rates")
scaled_rates['usdinr'].plot(legend=True)
scaled_rates['eurinr'].plot(legend=True)
scaled_rates['jpyinr'].plot(legend=True)
plt.show()
```

5) Defining the function which performs the Augmented Dicky-Fuller Test, and tells about the stationarity of the time series

```
def aug_df(series):
    result = adfuller(series.dropna(), autolag='AIC')

    print('ADF Statistic: %f' % result[0])

    print('p-value: %f' % result[1])

    print('Critical Values:')

    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
    if result[0] < result[4]["5%"]:
        print ("Reject Ho - Time Series is Stationary")
    else:
        print ("Failed to Reject Ho - Time Series is Non-Stationary")
    print(result)
```

6) Next we perform the Augmented Dicky-Fuller Test, twice. First without differencing and second with differencing. In the first test, we get the series' as non-stationary. On applying differencing and then performing the test once more, we get the time series' as stationary.

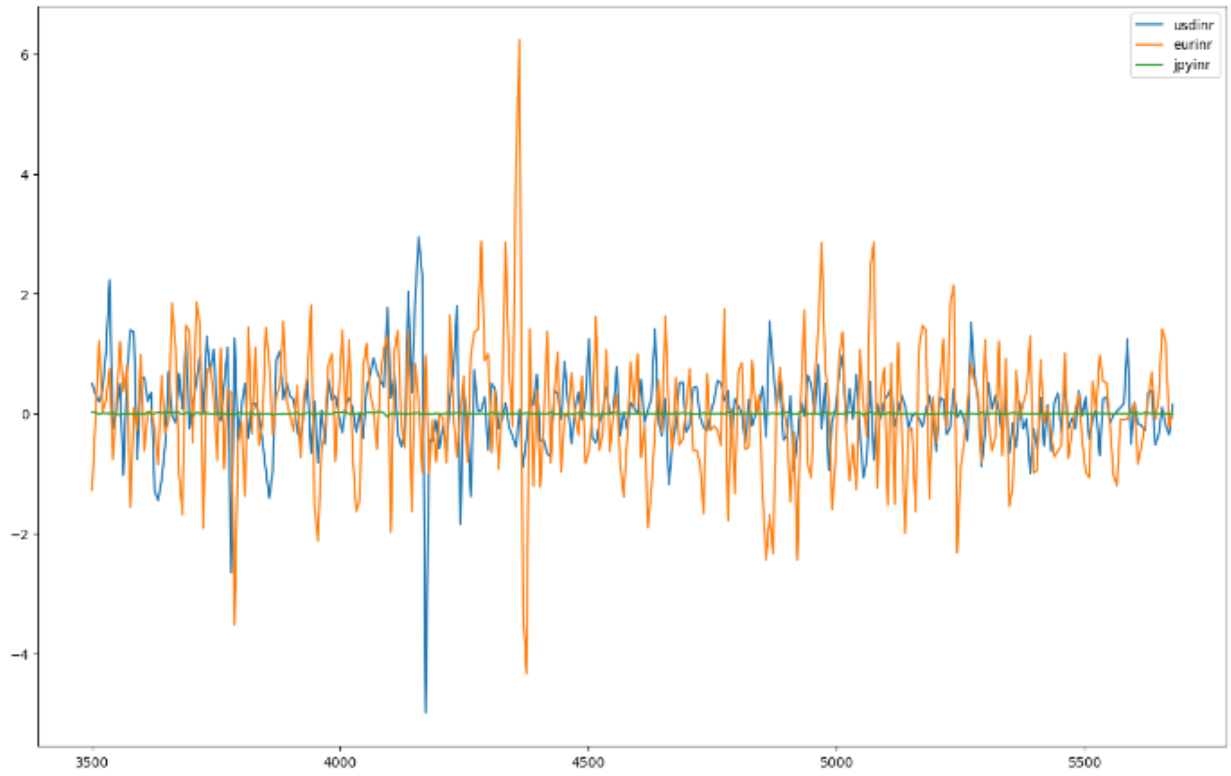
```
aug_df(rates['usdinr'])
aug_df(rates['jpyinr'])
aug_df(rates['eurinr'])
```

```
withoutdate = rates.drop('Date',axis=1)
rates_1 = withoutdate.diff()
aug_df(rates_1['usdinr'])
aug_df(rates_1['jpyinr'])
aug_df(rates_1['eurinr'])
```

7) Plotting the time series after differencing

```
plt.figure(figsize = (16,10))
rates_1['usdinr'].plot(legend=True)
rates_1['eurinr'].plot(legend=True)
```

```
rates_1['jpyinr'].plot(legend=True)
plt.show()
```



8) Splitting the dataset into train and test sets

```
train,test =
rates_1[:int(0.8*len(rates_1))],rates_1[int(-0.2*len(rates_1)):]
```

9) Finding the appropriate order of the VAR model to be fitted. Intuitively we felt that the rate of a particular week should atleast be a funciton of the last 8 observations i.e. atleast 2 month. Therefore the intention was to choose the value of p that would give the best AIC value where p was greater than 8 and less than 24 (that is 6 months).

```
for p in range(8,25):
    model = VAR(train)
    results = model.fit(p)
    print('Order =', p)
    print('AIC : ', results.aic)
```

We get the least AIC value at order 11, and this is what we use to fit our model

```
Order = 8
AIC :  -8.796588888357341
Order = 9
AIC :  -8.753387975165147
Order = 10
AIC :  -8.734451761512064
Order = 11
AIC :  -9.121925618460745
Order = 12
AIC :  -9.075881942857587
Order = 13
AIC :  -9.033655426199363
Order = 14
AIC :  -9.014543600688027
Order = 15
AIC :  -8.97238165374125
Order = 16
AIC :  -8.925253848888142
Order = 17
AIC :  -8.955638670944039
Order = 18
AIC :  -8.90622867655607
Order = 19
AIC :  -8.87117496850735
Order = 20
AIC :  -8.837114499092426
Order = 21
AIC :  -8.814347427451883
Order = 22
AIC :  -8.77410613111307
Order = 23
AIC :  -8.726697601941897
Order = 24
AIC :  -8.702234388187442
```

10) Fitting VAR model of order 11, and getting the predicted values for the exchange rates after that rolling back the differencing(by using cumulative sum)

```
p = 11
n = len(test)
results = model.fit(11)
```

```
results.summary()
```

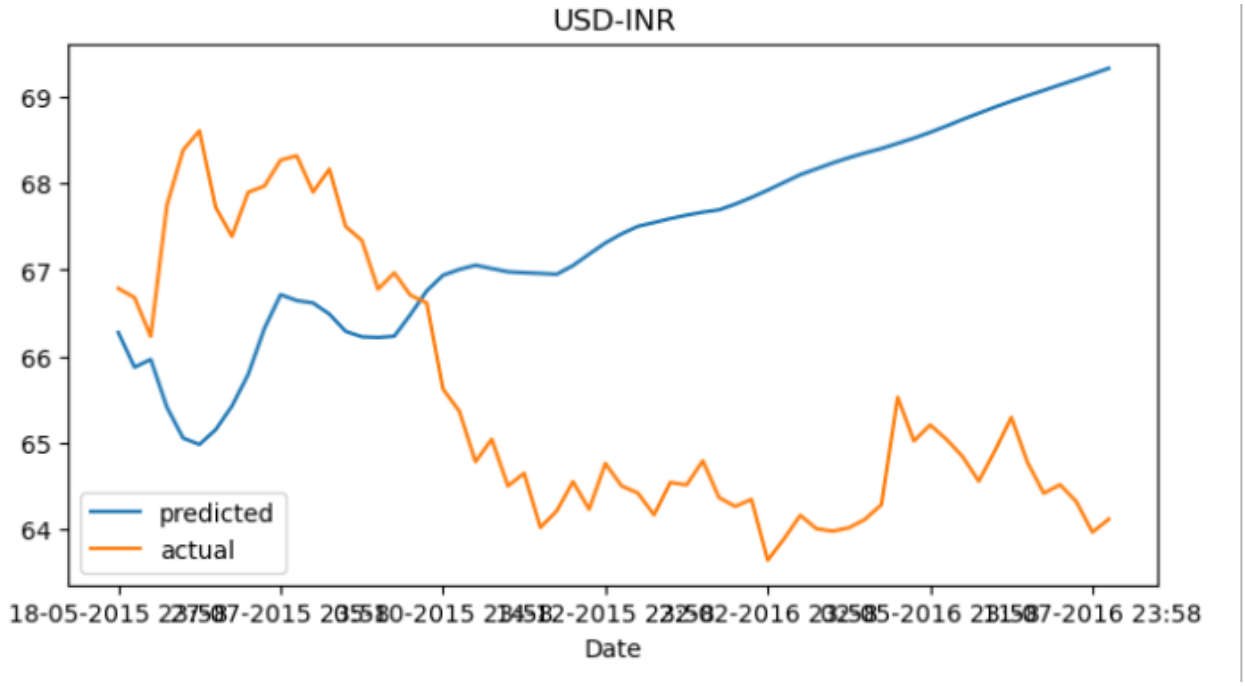
```
idx = pd.date_range(rates.iloc[-n]['Date'],periods=n,freq='7D')
z = results.forecast(y=train.values[-p:],steps=n)
pred_rates = pd.DataFrame(z,idx,columns=['usdinr','jpyinr','eurinr'])
pred_rates['usdinr'] = rates['usdinr'].iloc[-n-1]+pred['usdinr'].cumsum()
pred_rates['jpyinr'] = rates['jpyinr'].iloc[-n-1]+pred['jpyinr'].cumsum()
pred_rates['eurinr'] = rates['eurinr'].iloc[-n-1]+pred['eurinr'].cumsum()
```

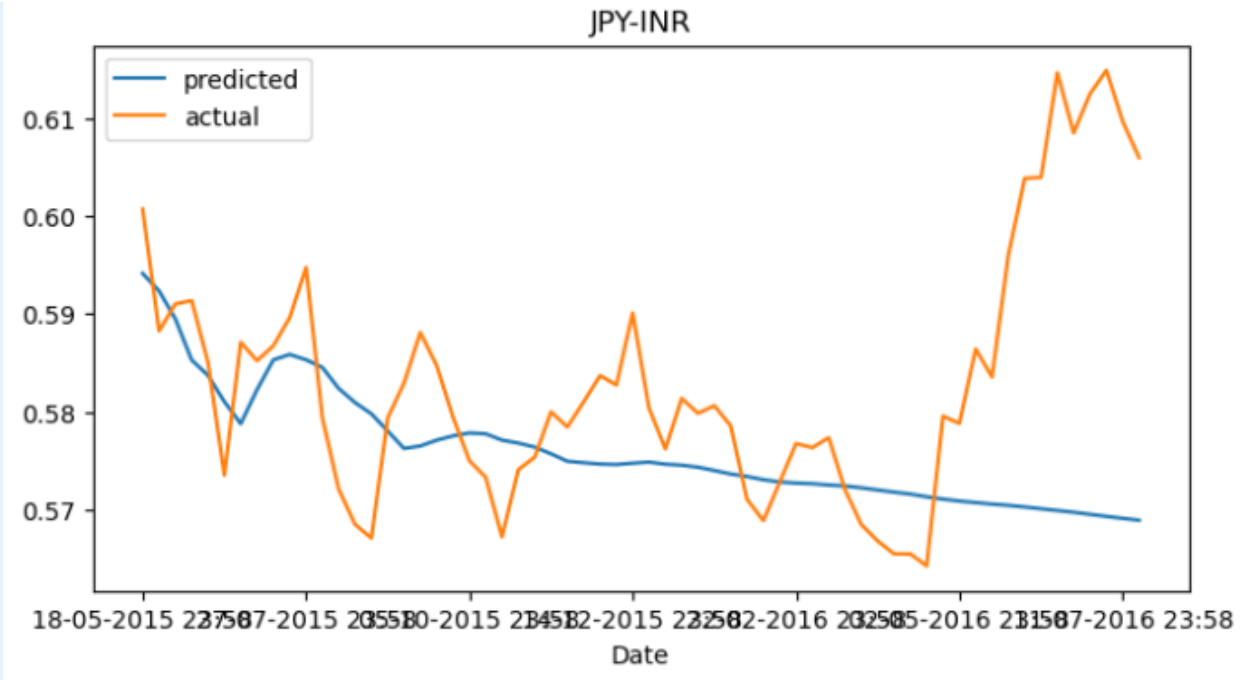
11) Plotting the predicted vs actual exchange rates for the 3 time series' for the VAR model

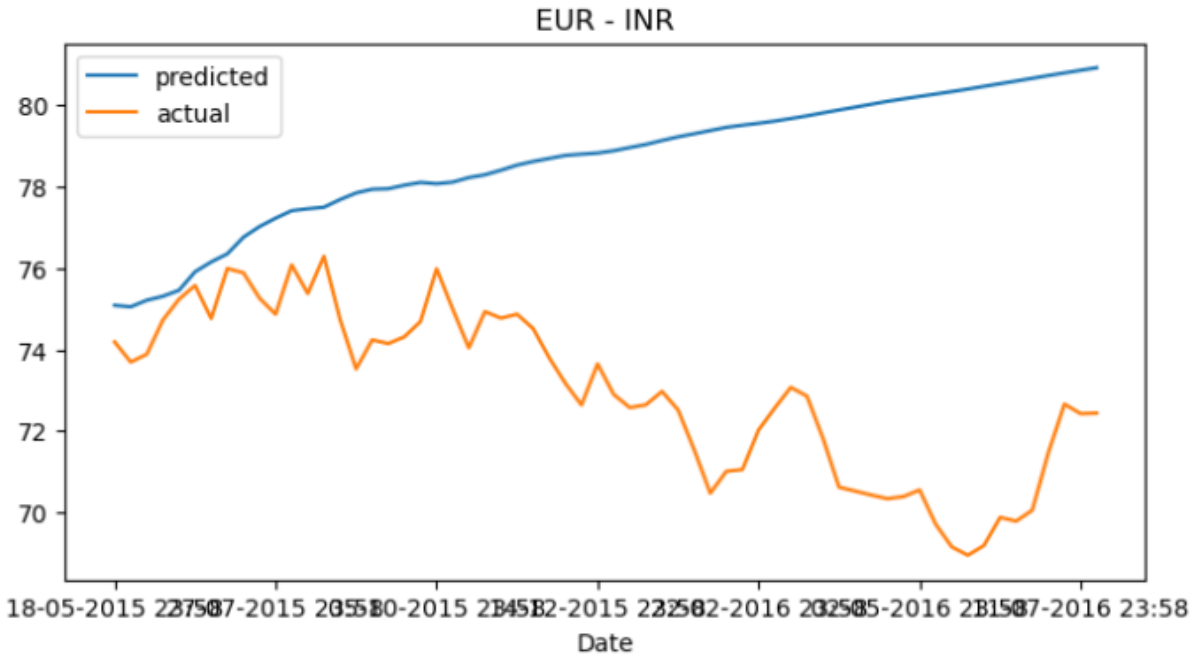
```
plt.figure(figsize=(8,4))
plt.title("USD-INR")
pred['usdinr'].plot(legend=True,label="predicted")
rates['usdinr'][-n-1:].plot(legend=True,label="actual")
plt.show()
```

```
plt.figure(figsize=(8,4))
plt.title("JPY-INR")
pred['jpyinr'].plot(legend=True,label="predicted")
rates['jpyinr'][-n-1:].plot(legend=True,label="actual")
plt.show()
```

```
plt.figure(figsize=(8,4))
plt.title('EUR - INR')
pred['eurinr'].plot(legend=True,label='predicted')
rates['eurinr'][-n-1:].plot(legend=True,label="actual")
plt.show()
```


USD-INR
69
68
67
66
65
64
predicted
actual
18-05-2015 23:58
07-2015 23:58
30-10-2015 23:58
02-2016 23:58
05-2016 23:58
07-2016 23:58
Date


JPY-INR
predicted
actual
0.61
0.60
0.59
0.58
0.57
18-05-2015 23:58
07-2015 23:58
30-10-2015 23:58
02-2016 23:58
05-2016 23:58
07-2016 23:58
Date



12) Now, fitting the AR model

```
model = AutoReg(train['usdinr'],lags=p).fit()
usdinr = model.forecast(n)

model = AutoReg(train['jpyinr'],lags=p).fit()
jpyinr = model.forecast(n)

model = AutoReg(train['eurinr'],lags=p).fit()
eurinr = model.forecast(n)
```

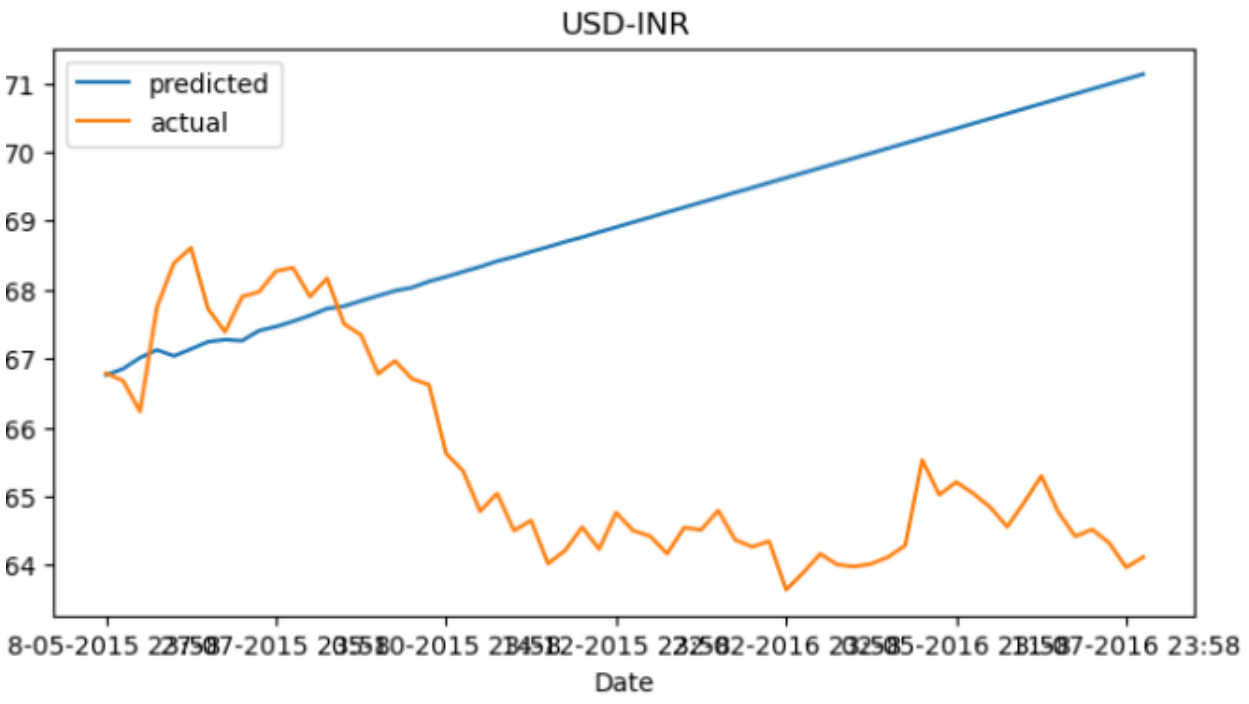
13) Getting the predicted values from the AR model and plotting them

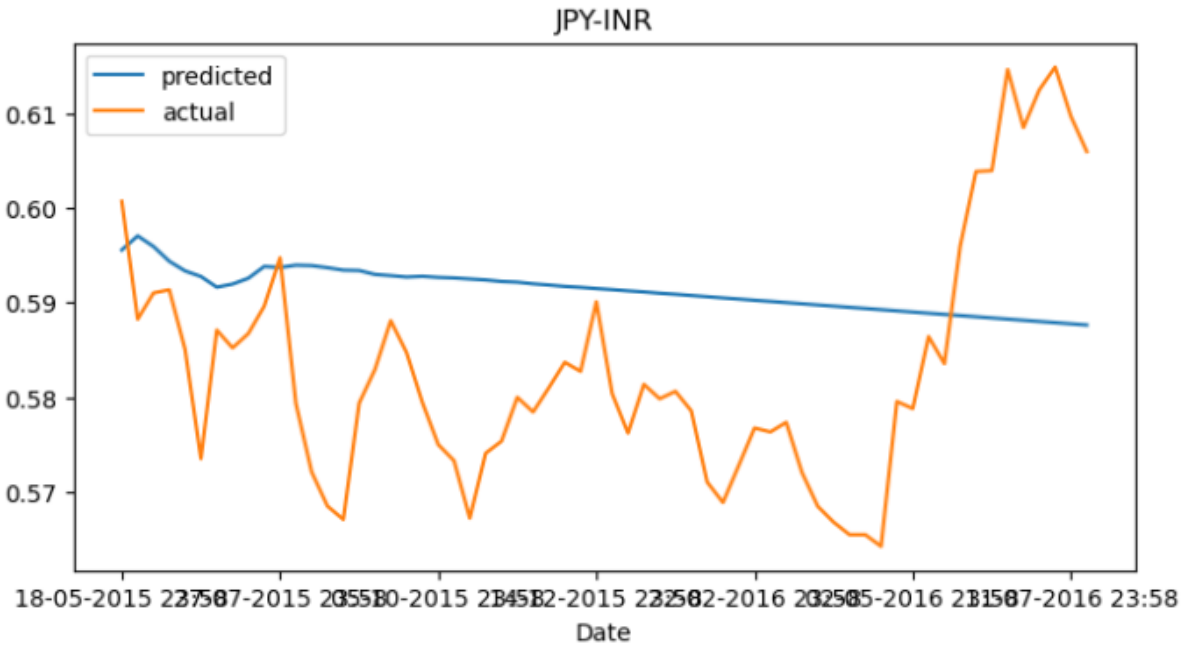
```
idx = pd.date_range(rates.iloc[-n]['Date'],periods=n,freq='7D')
pred = pd.DataFrame({'usdinr' : usdinr,
                     'jpyinr' : jpyinr,
                     'eurinr' : eurinr})
pred_rates = pd.DataFrame({'usdinr' : [],
                           'jpyinr' : [],
                           'eurinr' : []})
pred_rates['usdinr'] = rates['usdinr'].iloc[-n-1] + pred['usdinr'].cumsum()
pred_rates['jpyinr'] = rates['jpyinr'].iloc[-n-1] + pred['jpyinr'].cumsum()
pred_rates['eurinr'] = rates['eurinr'].iloc[-n-1] + pred['eurinr'].cumsum()
```

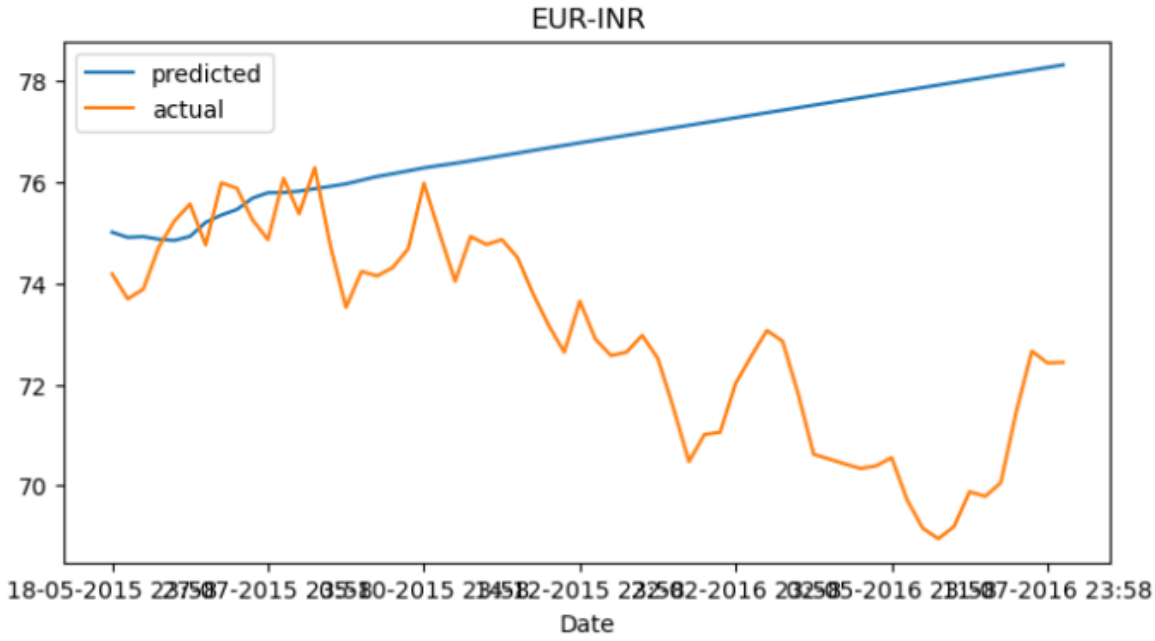
```
plt.figure(figsize=(8,4))
plt.title("USD-INR")
pred_rates['usdinr'].plot(legend=True,label="predicted")
rates['usdinr'][-n:].plot(legend=True,label="actual")
plt.show()
```

```
plt.figure(figsize=(8,4))
plt.title("JPY-INR")
pred_rates['jpyinr'].plot(legend=True,label="predicted")
rates['jpyinr'][-n:].plot(legend=True,label="actual")
plt.show()
```

```
plt.figure(figsize=(8,4))
plt.title("EUR-INR")
pred_rates['eurinr'].plot(legend=True,label="predicted")
rates['eurinr'][-n:].plot(legend=True,label="actual")
plt.show()
```







14) Defining the function which calculates the RMSE values for the models

```
from statsmodels.tools.eval_measures import rmse
varusdrmse = rmse(rates['usdinr'][-n:], pred['usdinr'])
varjpyrmse = rmse(rates['jpyinr'][-n:], pred['jpyinr'])
vareurrmse = rmse(rates['eurinr'][-n:], pred['eurinr'])
print('RMSE1-',varusdrmse,', RMSE2-',varjpyrmse,', RMSE3-',vareurrmse)
```

6) Results

We calculate the RMSE values for both the models

```
arusdrmse = rmse(rates['usdinr'][-n:], pred_rates['usdinr'])
arjpyrmse = rmse(rates['jpyinr'][-n:], pred_rates['jpyinr'])
areurrmse = rmse(rates['eurinr'][-n:], pred_rates['eurinr'])
varusdrmse = rmse(rates['usdinr'][-n:], pred_rates['usdinr'])
varjpyrmse = rmse(rates['jpyinr'][-n:], pred_rates['jpyinr'])
vareurrmse = rmse(rates['eurinr'][-n:], pred_rates['eurinr'])
```

1) For VAR:

```
RMSE1- 3.1218209971201154 , RMSE2- 0.016080805631104245 , RMSE3- 6.644683018526006
```

2) For AR:

```
RMSE1- 4.3338725755554 , RMSE2- 0.015797594119178863 , RMSE3- 4.765630370182896
```

RMSE values from prediction using VAR	**RMSE values from prediction using AR**
For usdinr = -3.12	For usdinr = -4.33
For jpyinr = 0.01	For jpyinr = 0.01
For eurinr = 6.64	For eurinr = -4.76

7) Conclusion

a) Lack of complexity in the models.

One can notice that the predictions made using the VAR and AR models are influences by the trends shown at the end of the training data. This is what we think is the reason why the model is not able to capture changes seen in the testing data far off from the end of the training data. A possible solution would be to consider an even larger p value to capture the past trend to a larger extent.

b) Not much difference in RMSE values for VAR and AR

One can notice that given we assume that our model is reasonable complex, and using a larger p value would result in overfitting, there isn't much difference in accuracy for exchange rates using AR than prediction using VAR model. As VAR model uses interdependence between series to forecast hence we can say that these series don't have a strong interdependence between them. However this conclusion itself requires further analysis since the exchange rates were all with respect to that of INR, and any changes in exchange rate of INR with respect to one currency should have a significant effect on its exchange rate another currency.