FORECAST EXCHANGE RATES

Anirudh M V
Punith M K
Anushka Anil Mhaskar
Pawan Raju Katwe
Harmanjot Singh

BUSINESS OBJECTIVE

- Data provided is related to USD/INR Exchange rates. The objective is to understand the underlying structure in the given dataset and come up with a suitable forecasting model which can effectively forecast USD/INR exchange rate for the next 30 days.

- This forecasting model will be used by exporting and importing companies to understand the currency movements and accordingly set their revenue expectations.

DATA

- The data was converted to .csv format.
- It had observation_date and DEXINUS columns.
- The observation date was converted to datetime format using “prase_date”

```
inrusd = pd.read_csv('Dataset.csv',parse_dates=["observation_date"])
inrusd.head()
```

	observation_date	DEXINUS
0	1973-01-02	8.02
1	1973-01-03	8.02
2	1973-01-04	8.00
3	1973-01-05	8.01
4	1973-01-08	8.00

DATA PREPROCESSING (EDA)

- There were 12649 values in the data set.
- The data also had multiple missing null values, which were 494, it's nearly 5% of the data, so we could not ignore it by dropping them. Instead We used Forward fill method to filling the missing values and then converted the rate into float

```
data.isnull().sum()
```

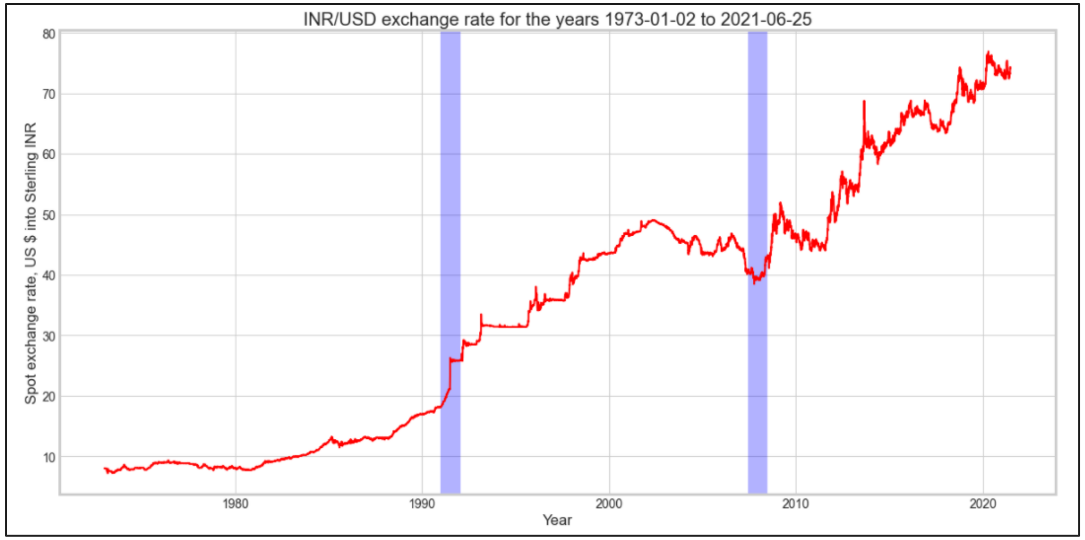
```
date       0
rate     494
dtype: int64
```

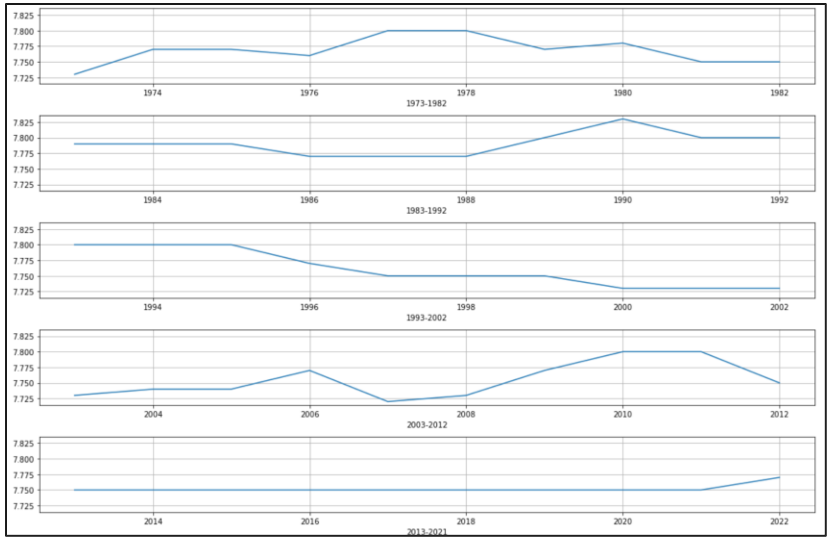
```
#transformation of values to float
data['rate'] = pd.to_numeric(data['rate'], downcast="float")
```

```
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    12649 non-null  datetime64[ns]
 1   rate    12649 non-null  float32
dtypes: datetime64[ns](1), float32(1)
memory usage: 247.1 KB
```

DATA VISUALIZATION

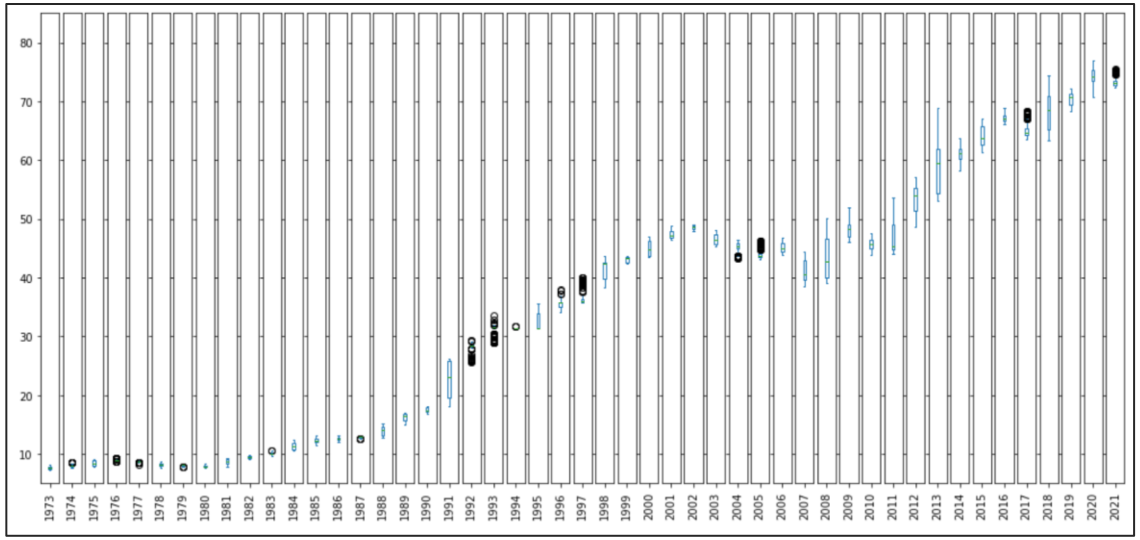
- Line plot of entire data
- Line plot of 10 years intervals

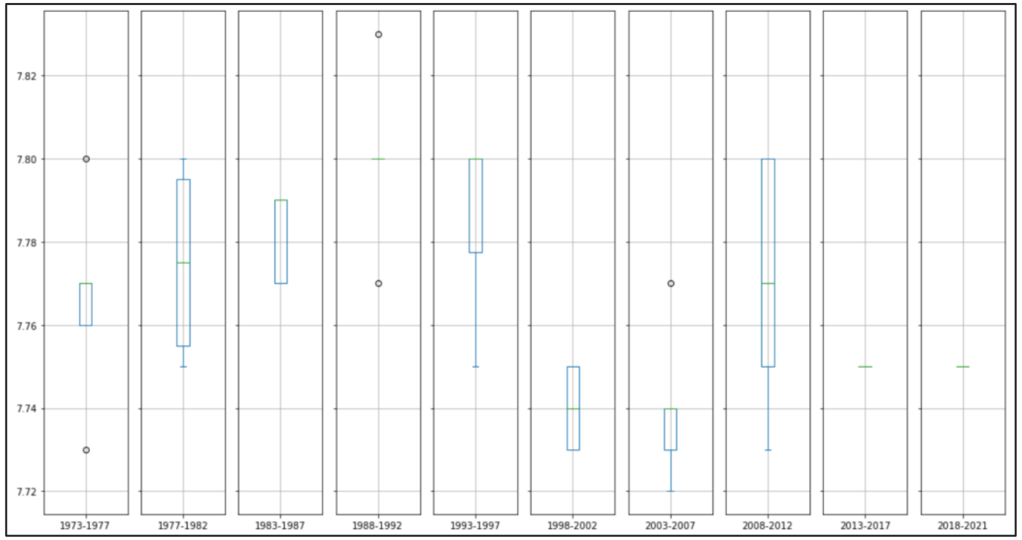




DATA VISUALIZATION

- Box plot of entire data
- Box plot of 5 years intervals





STATIONARITY

- Stationarity is an important concept in the field of time series analysis with tremendous influence on how the data is perceived and predicted.
- Since the data is not stationary we had to perform transformation use Difference and Logarithmic transformations.

	Time Series	ADF Stats	P Value
0	30 days	-0.442700	0.902722
1	6 Months	-2.004302	0.284690
2	1 year	-2.663459	0.080578
3	5 years	-0.980875	0.760176
4	10 years	-2.216786	0.200209

	Time Series on Differeation	ADF Stats	P Value
0	30 days	-4.925721	3.106929e-05
1	6 Months	-11.952723	4.279486e-22
2	1 year	-17.495970	4.411273e-30
3	5 years	-27.304792	0.000000e+00
4	10 years	-21.900816	0.000000e+00

	Time Series on Log Function	ADF Stats	P Value
0	30 days	-0.442667	0.902728
1	6 Months	-2.007642	0.283233
2	1 year	-2.657349	0.081714
3	5 years	-27.304792	0.000000
4	10 years	-2.670302	0.079320
5	Entire Data	-0.795877	0.820330

From the above Augmented Dickey-Fuller test. We can see that 1 year of data has p-value of less than 0.05 in Log Transformation and ADF transformation on normal data.

MODEL BUILDING

The Preprocessing and Stationarity test is completed and now the Time Series model building is the next step for forecasting results.

- ARIMA
- PyCaret
- FB PROPHET
- XGBOOST
- LSTM

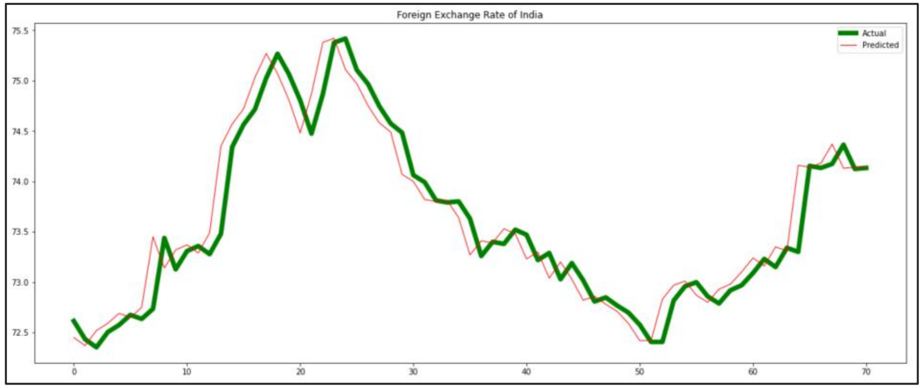
This are models were used for building time series forecasting.

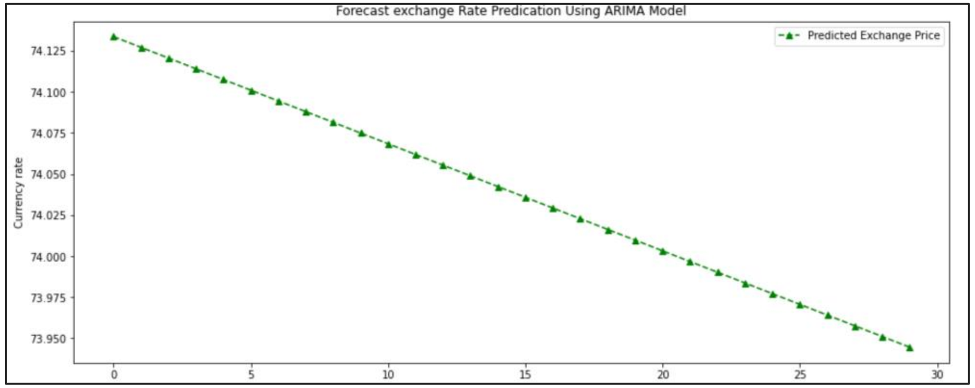
ARIMA MODEL

Since the ADF test had significant level of less than 0.05 for 1 year we used one year of data for ARIMA

- SARIMAX method was used and the best score was (0,1,0).
- But the 30 days future prediction was having downward trend.

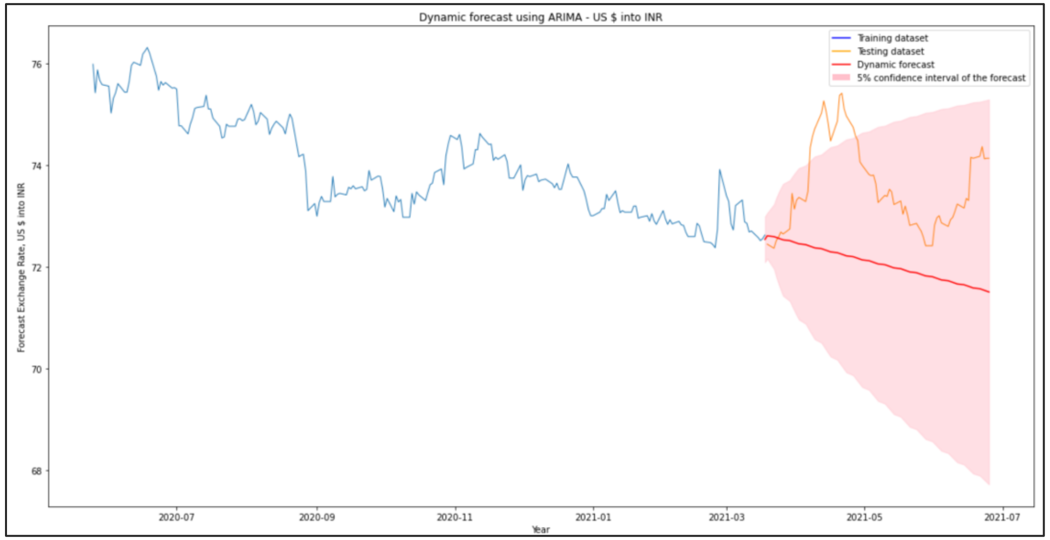

```
Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.544 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  214
Model:               SARIMAX(0, 1, 0)   Log Likelihood                  10.963
Date:                Fri, 13 Aug 2021   AIC                            -19.927
Time:                        18:59:36   BIC                            -16.565
Sample:                             0   HQIC                           -18.568
                                - 214
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0528      0.003     17.595      0.000       0.047       0.059
===================================================================================
Ljung-Box (L1) (Q):                   2.29   Jarque-Bera (JB):               138.37
Prob(Q):                              0.13   Prob(JB):                         0.00
Heteroskedasticity (H):               0.84   Skew:                             0.25
Prob(H) (two-sided):                  0.46   Kurtosis:                         6.92
===================================================================================
```

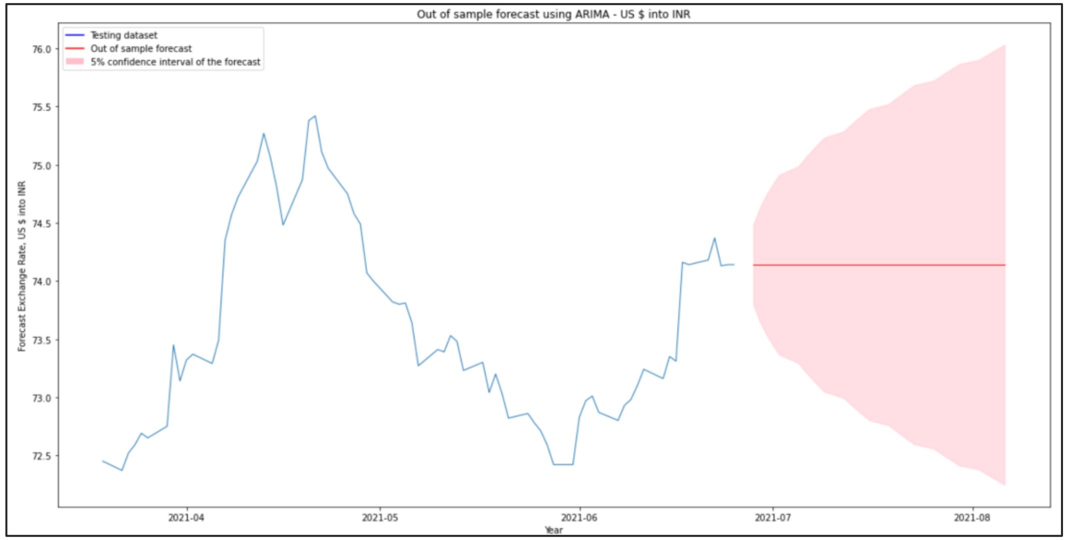




ARIMA MODEL

Further we used Dynamic forecasting and Out of sample forecasting, but the results were not as expected.

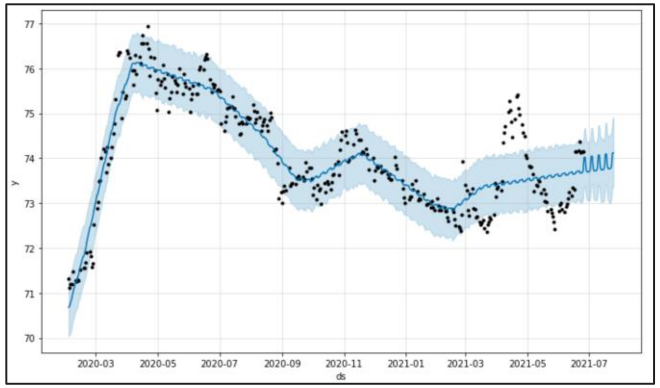




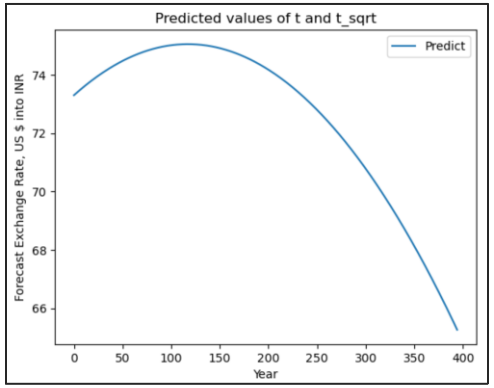
FB PROPHET MODEL

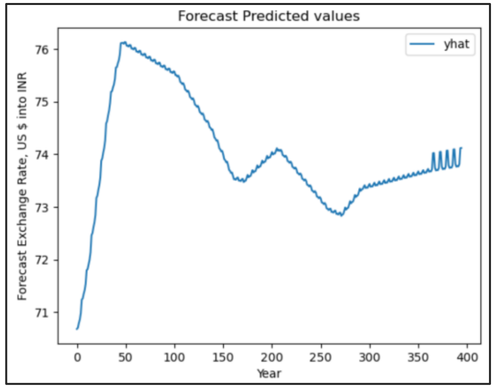
Further we used Dynamic forecasting and Out of sample forecasting, but the results were not as expected.

	ds	yhat	yhat_lower	yhat_upper
0	2020-02-03	70.678959	70.019891	71.374013
1	2020-02-04	70.700681	70.071201	71.357483
2	2020-02-05	70.788927	70.098784	71.439732
3	2020-02-06	70.858844	70.198977	71.504412
4	2020-02-07	70.978305	70.342177	71.598076
...	...	...	...	...
390	2021-07-21	73.774837	73.003813	74.518690
391	2021-07-22	73.768438	73.004637	74.537560
392	2021-07-23	73.811582	73.130657	74.585434
393	2021-07-24	74.116381	73.368988	74.867256
394	2021-07-25	74.119790	73.359234	74.911034



	Model	Values
0	rmse_Mult_sea	2.049318
1	rmse_add_sea_quad	1.960477
2	rmse_add_sea	2.049318
3	rmse_Quad	4.061458
4	rmse_Expt	69.150687
5	rmselin	0.874936





PYCARET MODEL

PyCaret is an open-source **low-code** machine learning library in Python that aims to reduce the time needed for experimenting with different machine learning models. It helps Data Scientist to perform any experiments end-to-end quickly and more efficiently.
PyCaret is a wrapper around many ML models and frameworks such as XGBoost, Scikit-learn, and many more.

```
best_pycaret = compare_models(sort = 'MAE')
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
xgboost	Extreme Gradient Boosting	8.6123	118.6715	9.8533	-2.8061	0.2004	0.1589	0.2567
lightgbm	Light Gradient Boosting Machine	8.6298	118.2656	9.8434	-2.8045	0.1998	0.1592	0.0333
catboost	CatBoost Regressor	8.6457	118.5746	9.8600	-2.8361	0.1998	0.1595	0.7700
gbr	Gradient Boosting Regressor	8.6559	117.4471	9.8269	-2.8899	0.1971	0.1592	0.1033
et	Extra Trees Regressor	8.7825	124.0813	10.0119	-2.9237	0.2039	0.1614	0.3433
dt	Decision Tree Regressor	8.8500	125.9306	10.0796	-2.9881	0.2052	0.1626	0.0200
rf	Random Forest Regressor	8.8641	125.9187	10.0829	-2.9912	0.2052	0.1628	0.3433
knn	K Neighbors Regressor	9.0842	130.1866	10.2498	-3.1327	0.2089	0.1669	1.2200
ada	AdaBoost Regressor	9.8182	158.7806	10.9703	-3.8418	0.2238	0.1772	0.0333
par	Passive Aggressive Regressor	20.6155	484.3937	21.2012	-17.7566	0.5595	0.4134	1.1433
huber	Huber Regressor	22.7901	568.8867	23.3323	-23.7536	0.6187	0.4548	0.9900
llar	Lasso Least Angle Regression	23.3042	596.6053	23.8353	-24.7923	0.6316	0.4630	1.9867

```
prediction_holdout = predict_model(best_pycaret);
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Extreme Gradient Boosting	3.5277	13.6845	3.6993	-11.2634	0.3298	0.4188

PyCaret described XGBOOST is the best model compared to other models.

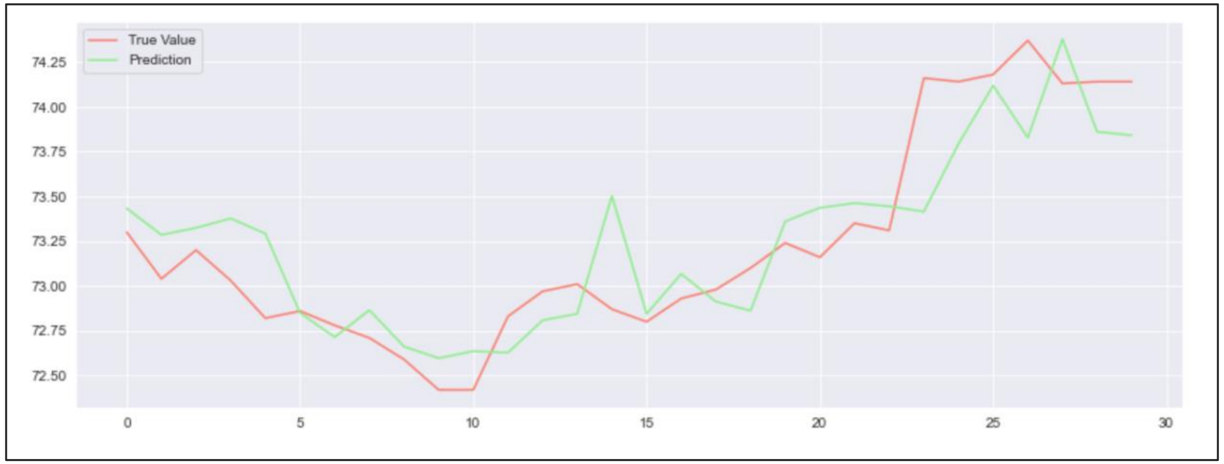
XGBOOST MODEL

XGBoost stands for “Extreme Gradient Boosting”. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. It provides a parallel tree boosting to solve many data science problems in a fast and accurate way.

```
#bayesian hyper parameter tuning
#define the params
def xgb_evaluate(max_depth, gamma, colsample_bytree):
    params = {'eval_metric': 'rmse',
              'max_depth': int(max_depth),
              'subsample': 0.8,
              'eta': 0.1,
              'gamma': gamma,
              'colsample_bytree': colsample_bytree}

    cv_result = xgb.cv(params, dtrain, num_boost_round=250, nfold=3)
    return -1.0 * cv_result['test-rmse-mean'].iloc[-1]
```

```
#run optimizer
xgb_bo = BayesianOptimization(xgb_evaluate, {'max_depth': (3, 7),
                                             'gamma': (0, 1),
                                             'colsample_bytree': (0.3, 0.9)})
#define iter points
xgb_bo.maximize(init_points=10, n_iter=15, acq='ei')
```



The XGBOOST Model prediction were not upto the expectations and the rmse was 0.81 so we denied proceeding further on this model

LSTM MODEL

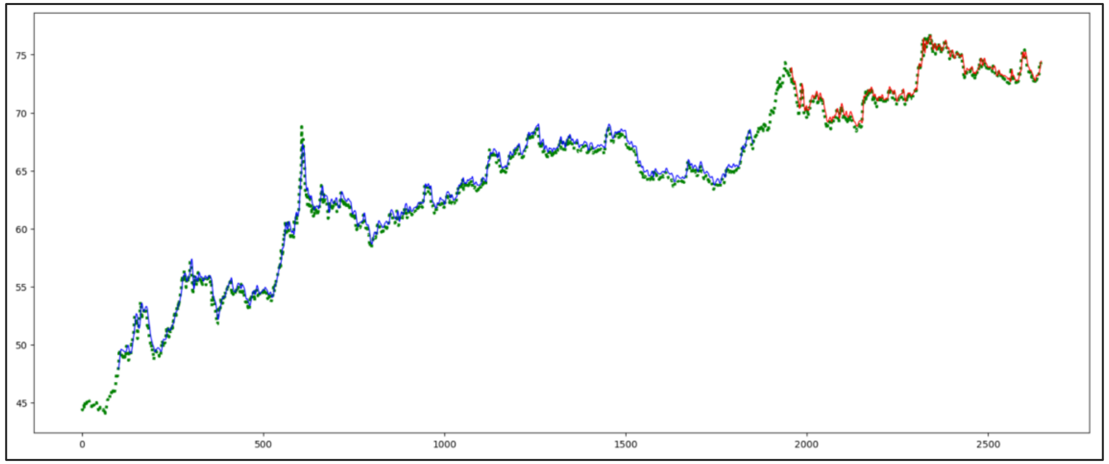
Long short-term memory (LSTM) units or blocks are part of a recurrent neural network structure. Recurrent neural networks are made to utilize certain types of artificial memory processes that can help these artificial intelligence programs to more effectively imitate human thought.

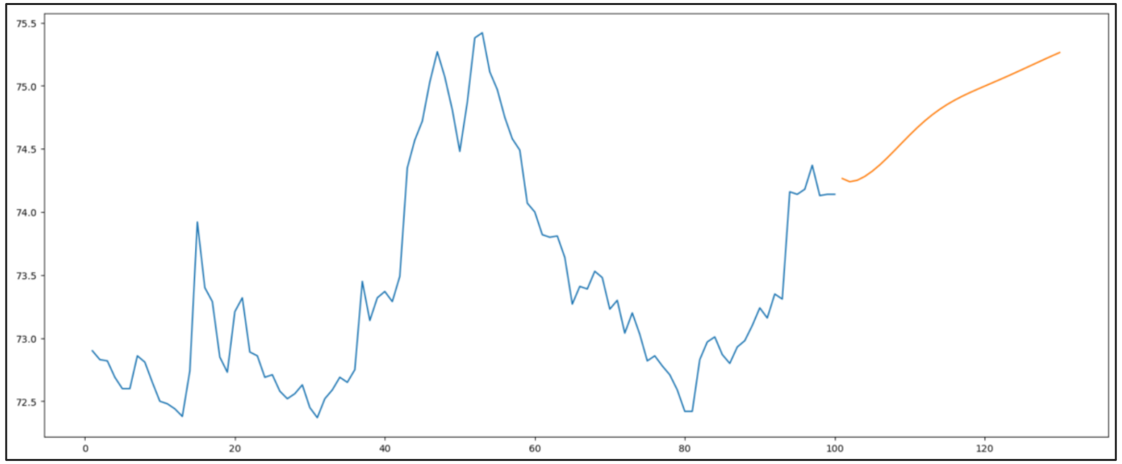
Steps:

- MinMax Scaler
- Splitting Data into 70% for Training and 30% for Testing
- Time step 100 for transformation of x_train, y_train, x_test, y_test
- Building Model
- Fitting Model with ephos = 100
- Predicting train and test data
- Inverse transformation of train and test data
- Calculating RMSE Scores

LSTM MODEL

- Visualization of Train and test on actual data
- Visualization on future 30 day.





From the above visualization we can say that the LSTM model is providing us best results with upward trend.

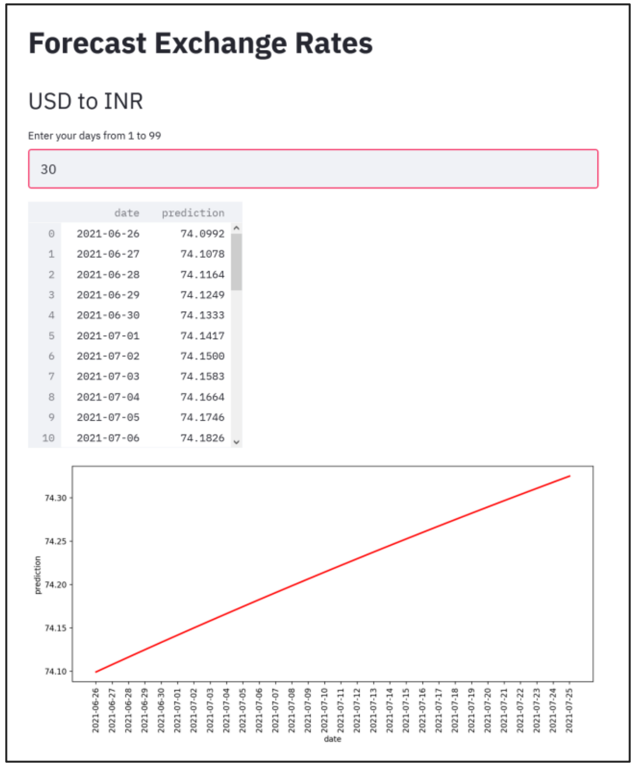
- RMSE = 0.0691

RMSE

MODELS	RMSE
ARIMA	0.2503
FB PROPHET	0.8745
XGBOOST	0.2984
LSTM	0.0691

DEPLOYMENT

The Model building and Evaluation process are completed now we have deployed the code for LSTM model using Streamlit.



Thank you