vietnam-er-forcast-arima

April 1, 2024

[1]:
```
import pandas as pd
import numpy as np
```

[15]:
```
ER = pd.read_csv("C:\\Users\\sayan maitra\\Downloads\\Vietnam_Exchange_Rate.
 ↪csv", index_col = "Year", parse_dates= True)
ER.head()
```

[15]:
```
             USD_VND_ER
Year
1986-01-01    22.936728
1987-01-01    78.953316
1988-01-01   611.646087
1989-01-01  4501.686529
1990-01-01  6537.604686
```

[16]:
```
ER.shape
```

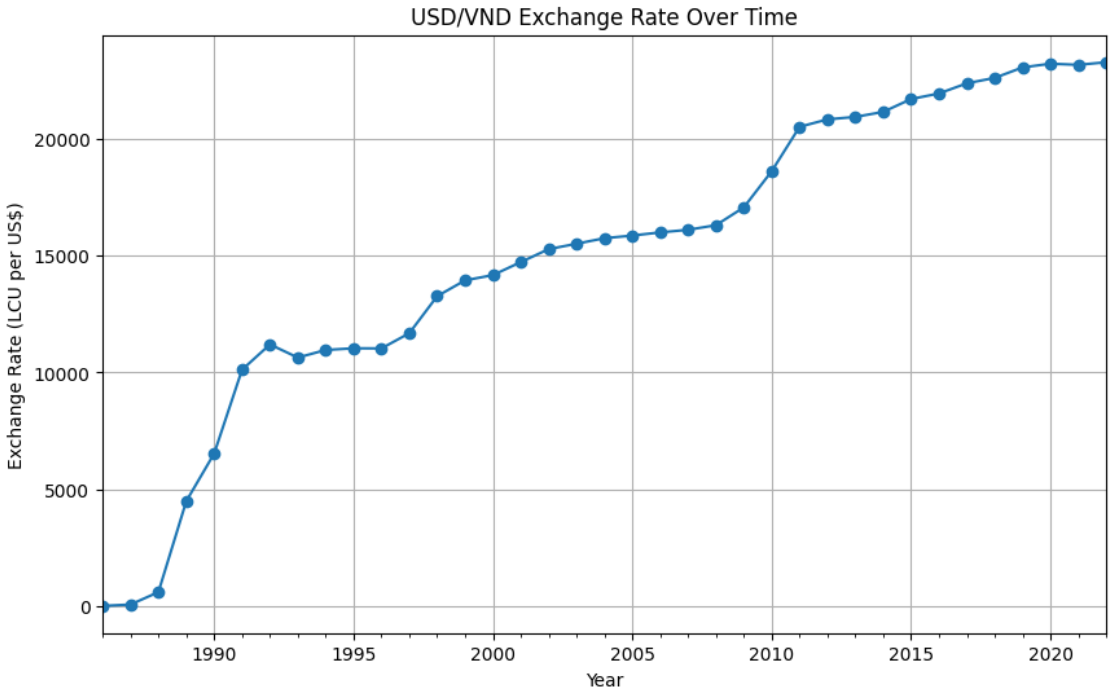
[16]:
```
(37, 1)
```

[17]:
```
ER.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 37 entries, 1986-01-01 to 2022-01-01
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   USD_VND_ER  37 non-null     float64
dtypes: float64(1)
memory usage: 592.0 bytes
```

[11]:
```
import matplotlib.pyplot as plt

ER['USD_VND_ER'].plot(figsize=(10, 6), marker='o')
plt.title('USD/VND Exchange Rate Over Time')
plt.xlabel('Year')
plt.ylabel('Exchange Rate (LCU per US$)')
plt.grid(True)
plt.show()
```



[28]:
```
from statsmodels.tsa.stattools import adfuller

def ad_test(dataset):
    dftest = adfuller(dataset, autolag = "AIC")
    print("1. ADF : ", dftest[0])
    print("2. P-Value : ", dftest[1])
    print("3. Num of Lags : ", dftest[2])
    print("4. Num of Observations Used For ADF Regression and Critical Values␣
 ↪Calculation : ", dftest[3])
    print("5. Critical Values : ")
    for key, val in dftest[4].items():
        print("\t",key,": ", val)
```

[29]:
```
ad_test(ER["USD_VND_ER"])
```

```
1. ADF :  -3.288609035273156
2. P-Value :  0.01539539279105131
3. Num of Lags :  2
4. Num of Observations Used For ADF Regression and Critical Values Calculation :
34
5. Critical Values :
         1% :  -3.639224104416853
         5% :  -2.9512301791166293
         10% :  -2.614446989619377
```

[31]:
```
from pmdarima import auto_arima
import warnings
warnings.filterwarnings("ignore")
```

[33]:
```
stepwise_fit = auto_arima(ER["USD_VND_ER"], trace= True,
                          suppress_warnings=True)
stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.64 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=597.926, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=591.966, Time=0.02 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=593.505, Time=0.21 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=610.213, Time=0.02 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=593.962, Time=0.06 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=593.937, Time=0.30 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=594.853, Time=0.42 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=594.190, Time=0.05 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0] intercept
Total fit time: 1.756 seconds
```

[33]:

Dep. Variable:	y	**No. Observations:**	37
Model:	SARIMAX(1, 1, 0)	**Log Likelihood**	-292.983
Date:	Mon, 01 Apr 2024	**AIC**	591.966
Time:	02:49:14	**BIC**	596.716
Sample:	01-01-1986	**HQIC**	593.624
	- 01-01-2022		
Covariance Type:	opg		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
intercept	368.5316	340.819	1.081	0.280	-299.461	1036.525
ar.L1	0.4366	0.187	2.331	0.020	0.070	0.804
sigma2	6.976e+05	1.41e+05	4.964	0.000	4.22e+05	9.73e+05

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	87.97
Prob(Q):	0.90	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.12	**Skew:**	2.25
Prob(H) (two-sided):	0.00	**Kurtosis:**	9.19

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[178]:
```
print(ER.shape)
train=ER.iloc[:-11]
test=ER.iloc[-13:]
print(train.shape,test.shape)
```

```
(37, 1)
```

```
(26, 1) (13, 1)
```

[179]:
```
from statsmodels.tsa.arima.model import ARIMA
model=ARIMA(train['USD_VND_ER'],order=(1,1,0))
model=model.fit()
model.summary()
```

[179]:

Dep. Variable:	USD_VND_ER	**No. Observations:**	26
Model:	ARIMA(1, 1, 0)	**Log Likelihood**	-208.924
Date:	Mon, 01 Apr 2024	**AIC**	421.848
Time:	03:16:56	**BIC**	424.286
Sample:	01-01-1986 - 01-01-2011	**HQIC**	422.524
Covariance Type:	opg		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
ar.L1	0.6599	0.155	4.246	0.000	0.355	0.964
sigma2	1.058e+06	1.92e+05	5.524	0.000	6.83e+05	1.43e+06

Ljung-Box (L1) (Q):	0.82	**Jarque-Bera (JB):**	17.48
Prob(Q):	0.36	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.10	**Skew:**	1.37
Prob(H) (two-sided):	0.00	**Kurtosis:**	6.05

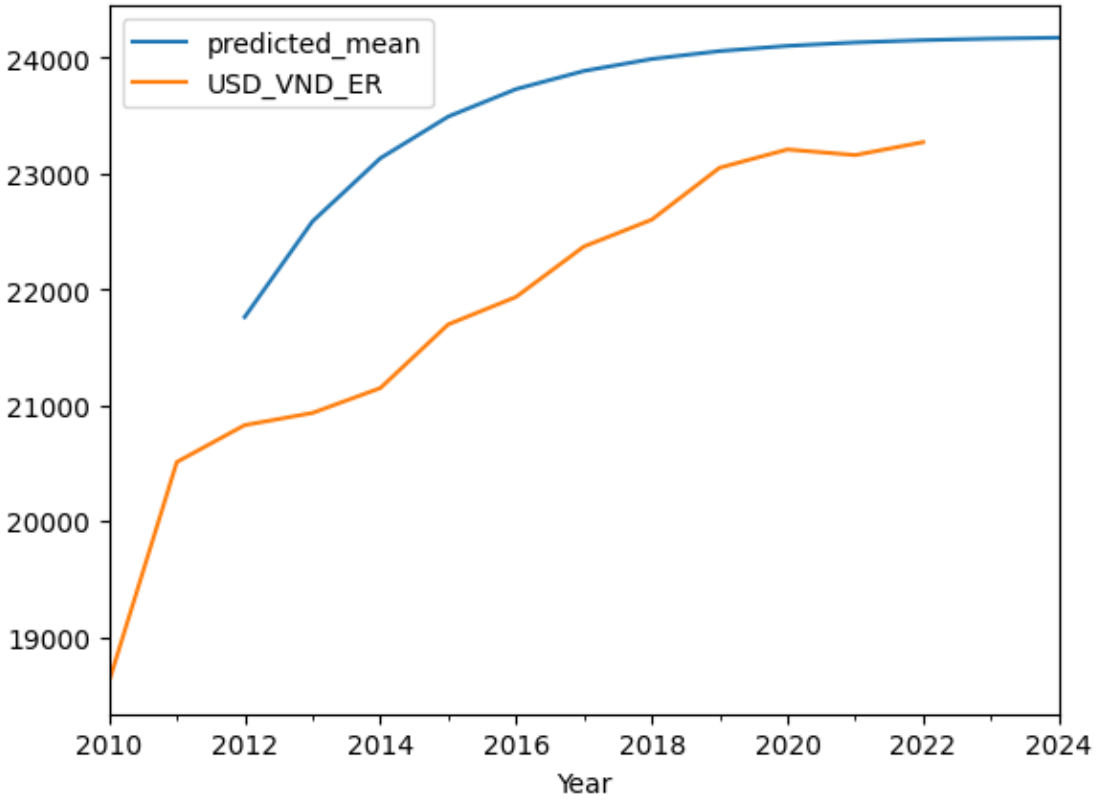
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

[180]:
```
start=len(train)
end=len(train)+len(test)-1
pred = model.predict(start=start,end=end,typ="levels")
print(pred)
```

```
2012-01-01    21761.451540
2013-01-01    22587.436997
2014-01-01    23132.496624
2015-01-01    23492.176077
2016-01-01    23729.524989
2017-01-01    23886.149186
2018-01-01    23989.503943
2019-01-01    24057.706725
2020-01-01    24102.713065
2021-01-01    24132.412304
2022-01-01    24152.010537
2023-01-01    24164.943218
2024-01-01    24173.477365
Freq: AS-JAN, Name: predicted_mean, dtype: float64
```

[181]:
```
pred.plot(legend=True)
test["USD_VND_ER"].plot(legend=True)
```

[181]: <Axes: xlabel='Year'>



[182]:
```
test["USD_VND_ER"].mean()
```

[182]: 21794.337956153842

[183]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse= sqrt(mean_squared_error(pred,test['USD_VND_ER']))
print(rmse)
```

1975.799513723094

[185]:
```
model2=ARIMA(ER["USD_VND_ER"],order=(1,1,0))
model2=model2.fit()
ER.tail()
```

[185]:

Year	USD_VND_ER
2018-01-01	22602.05000
2019-01-01	23050.24167

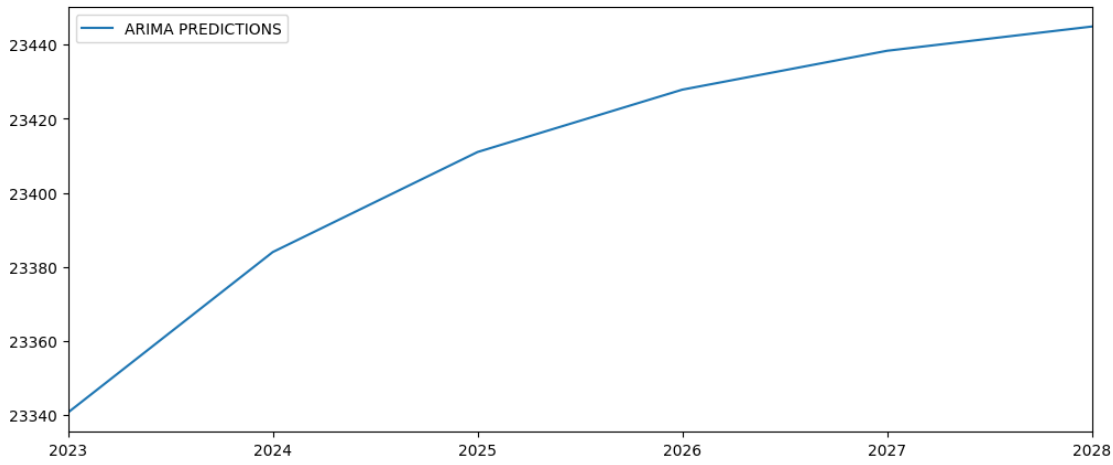
```
2020-01-01   23208.36833
2021-01-01   23159.78259
2022-01-01   23271.21250
```

[187]:
```
index_future_dates= pd.date_range(start="2022-01-01", end="2027-01-01")
pred=model2.predict(start=len(ER),end=len(ER)+5, typ = "levels").rename("ARIMA␣
↪PREDICTIONS")
print(pred)
```

```
2023-01-01    23340.683336
2024-01-01    23383.994840
2025-01-01    23410.997343
2026-01-01    23427.832019
2027-01-01    23438.327576
2028-01-01    23444.871018
Freq: AS-JAN, Name: ARIMA PREDICTIONS, dtype: float64
```

[188]:
```
pred.plot(figsize=(12,5),legend=True)
```

[188]: <Axes: >



[ ]: