# Time Series Forecasting

## With the usage of ARIMA Model

Kiran R Pande (3rd Year, BTech, NSUT Delhi)
Soham Jain (3rd Year, BTech, NSUT Delhi)

# Introduction to Time Series Forecasting

A time series is a sequence where a metric is recorded over regular time intervals.

Depending on the frequency, a time series can be of yearly (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: weather), hourly (ex: stocks price), minutes (ex: inbound calls in a call center) and even seconds wise (ex: web traffic)

Now forecasting a time series can be broadly divided into two types.

If you use only the previous values of the time series to predict its future values, it is called Univariate Time Series Forecasting

And if you use predictors other than the series (a.k.a exogenous variables) to forecast it is called Multi Variate Time Series Forecasting.

# A Brief about the ARIMA Model

ARIMA, short for ‘**Auto Regressive Integrated Moving Average’** is actually a class of models that ‘explains’ a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.

An ARIMA model is characterized by 3 terms: p, d, q
where,
**p** is the order of the Autoregressive term
**q** is the order of the Moving Average term
**d** is the number of differencing required to make the time series stationary

These terms are further explained in their dedicated slides

If a time series, has seasonal patterns, then you need to add seasonal terms and it becomes SARIMA, short for ‘Seasonal ARIMA’. More on that once we finish ARIMA.

# Steps to be followed

Running AD Fuller Test

Transforming the data

Values of p,d,q

Fitting the Model

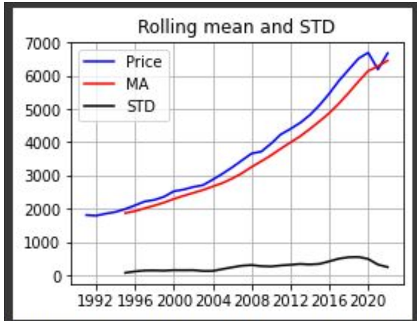
Forecasting

Plotting the Results

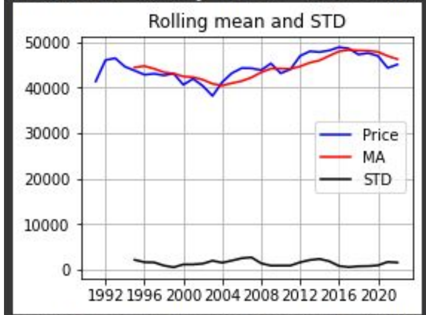
# Ad Fuller Test

Augmented Dickey Fuller test (ADF Test) is a common statistical test used to test whether a given Time series is stationary or not.

```
#Perform Dickey fuller test
print('Results of dickey fuller test {name}'.format
(name = ticker[i]))
dftest = adfuller(hm, autolag = 'AIC')
dfoutput = pd.Series(dftest[0:4], index =
['Test Statistics', 'P Value', 'Lags Used',
'Number of observations used'])
for key,value in dftest[4].items():
dfoutput['Critical Value (%s)'%key] = value
```




Results of dickey fuller test ECONOMICS:INGDPPCP



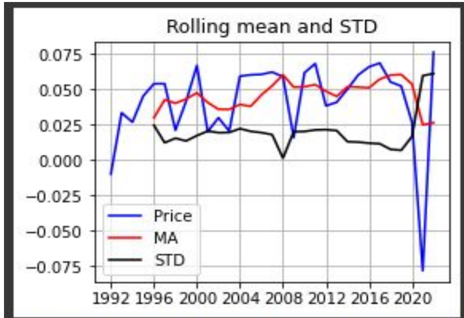

```
Results of dickey fuller test ECONOMICS:SAGDPPCP
                    Test Statistics   P Value  Lags Used  \
ECONOMICS:INGDPPCP         1.498749  0.997520        0.0
ECONOMICS:SAGDPPCP        -1.623561  0.470863        8.0

                    Number of observations used  Critical Value (1%)  \
ECONOMICS:INGDPPCP                         31.0            -3.661429
ECONOMICS:SAGDPPCP                         23.0            -3.752928

                    Critical Value (5%)  Critical Value (10%)
ECONOMICS:INGDPPCP            -2.960525             -2.619319
ECONOMICS:SAGDPPCP            -2.998500             -2.638967
```

# Transforming the data

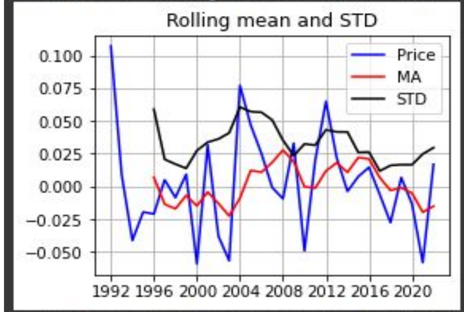
As we can see in the slide before the P value is significantly high which denotes that the data is not stationary.

To make the data stationary we have to transform the data in such a way that the moving average and standard deviation becomes linear or close to linear. For this we have taken the logarithm of the data set.

By doing this the P value decreases drastially and the values comes closer to 0, this means that our data is now stationary and now the model can be fitted.




```
Results of dickey fuller test ECONOMICS:INGDPPCP
```



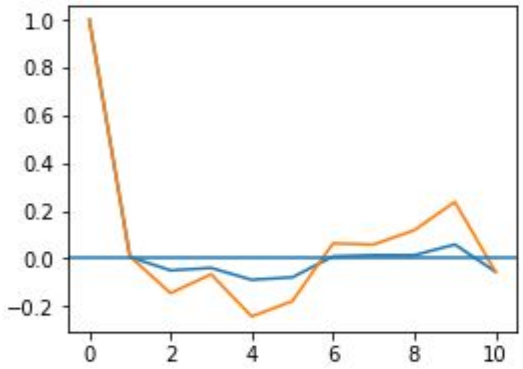

```
Results of dickey fuller test ECONOMICS:SAGDPPCP
                    Test Statistics   P Value  Lags Used  \
ECONOMICS:INGDPPCP        -5.416422  0.000003        0.0
ECONOMICS:SAGDPPCP        -1.813374  0.373842        9.0

                    Number of observations used  Critical Value (1%)  \
ECONOMICS:INGDPPCP                         30.0            -3.669920
ECONOMICS:SAGDPPCP                         21.0            -3.788386

                    Critical Value (5%)  Critical Value (10%)
ECONOMICS:INGDPPCP            -2.964071             -2.621171
ECONOMICS:SAGDPPCP            -3.013098             -2.646397
```

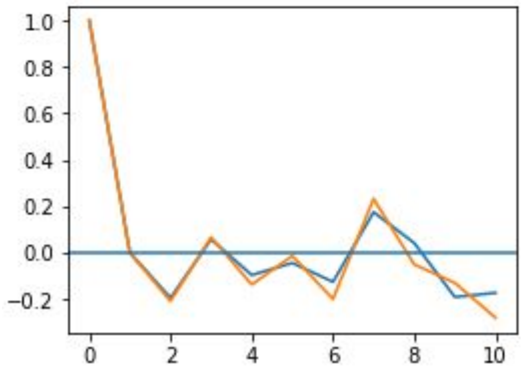
# Values of p,d,q

ACF (p term) The ACF tells how many MA terms are required to remove any autocorrelation in the stationarized series. Or 'p' is the order of the 'Auto Regressive' (AR) term. It refers to the number of lags of Y to be used as predictors

$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \alpha_3 Y_{t-3}$$

```
 from statsmodels.tsa.stattools import acf , pacf
 lag_acf = acf(df_logdiffshift.iloc[:, i], nlags=10)
Final_ACF = []
for z in range(len(ticker)):
 ER = AP.iloc[:,z].to_list()
 for i in ER:
   if i<0:
     Final_ACF.append(ER.index(i))
     break
```



# Values of p,d,q

PACF (q term) Partial autocorrelation can be imagined as the correlation between the series and its lag, after excluding the contributions from the intermediate lags. So, PACF sort of conveys the pure correlation between a lag and the series. That way, you will know if that lag is needed in the AR term or not. Partial autocorrelation of lag (k) of a series is the coefficient of that lag in the autoregression equation of Y.

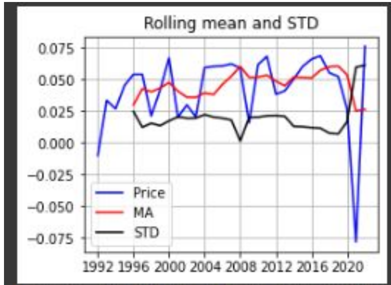
$$Y_t = \alpha_0 + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \alpha_3 Y_{t-3}$$

```
  from statsmodels.tsa.stattools import acf , pacf
  lag_pacf = pacf(df_logdiffshift.iloc[:, i], nlags=10, method = 'ols')
Final_PACF = []
for z in range(len(ticker)):
  EN = PP.iloc[:,z].to_list()
  for i in EN:
    if i<0:
      Final_PACF.append(EN.index(i))
      break
```

# Values of p,d,q

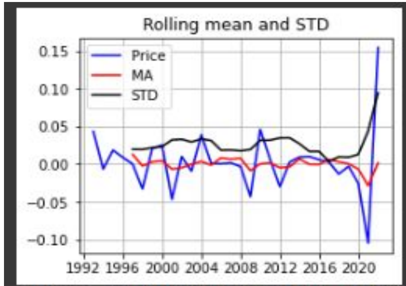
D is the number of differencing required to make the time series stationary. The value of d, therefore, is the minimum number of differencing needed to make the series stationary. And if the time series is already stationary, then d = 0.

```
df_logdiffshift = dflogscale - dflogscale.shift()
df_logdiffshift.shift()
df_logdiffshift.dropna(inplace = True)
```




```
Results of dickey fuller test ECONOMICS:INGDPPCP
                    Test Statistics   P Value  Lags Used  \
ECONOMICS:INGDPPCP        -5.416422  0.000003        0.0

                    Number of observations used  Critical Value (1%)  \
ECONOMICS:INGDPPCP                         30.0             -3.66992

                    Critical Value (5%)  Critical Value (10%)
ECONOMICS:INGDPPCP            -2.964071             -2.621171
```

```
df_logdiffshift2 = df_logdiffshift -
df_logdiffshift2.dropna(inplace = True)
```




```
Results of dickey fuller test ECONOMICS:INGDPPCP
                    Test Statistics       P Value  Lags Used  \
ECONOMICS:INGDPPCP        -7.433387  6.274680e-11        0.0

                    Number of observations used  Critical Value (1%)  \
ECONOMICS:INGDPPCP                         29.0             -3.67906

                    Critical Value (5%)  Critical Value (10%)
ECONOMICS:INGDPPCP            -2.967882             -2.623158
```

# Fitting the model

```python
from statsmodels.tsa.arima_model import ARIMA
for i in range(len(ticker)):
    p = int(Final_ACF[i])
    q = int(Final_PACF[i])
    d = int(lag)
    isd = dflogscale.iloc[:,i].to_frame()
    model = ARIMA(isd, order = (p, d, q))
    results_AR = model.fit()
    e = int(Forecast_Years)
    tf = results_AR.forecast(e)
    tf = np.exp(tf[0])
    hsd.append(tf)
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:     D.ECONOMICS:INGDPPCP   No. Observations:                   31
Model:                   ARIMA(1, 1, 1)   Log Likelihood                  66.181
Method:                         css-mle   S.D. of innovations              0.028
Date:                  Wed, 12 Oct 2022   AIC                           -124.362
Time:                          15:20:54   BIC                           -118.626
Sample:                      12-31-1991   HQIC                          -122.492
                           - 12-31-2021
==============================================================================================
                                coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                         0.0441      0.002     17.823      0.000       0.039       0.049
ar.L1.D.ECONOMICS:INGDPPCP    0.8226      0.149      5.537      0.000       0.531       1.114
ma.L1.D.ECONOMICS:INGDPPCP   -1.0000      0.087    -11.446      0.000      -1.171      -0.829
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.2156           +0.0000j            1.2156            0.0000
MA.1            1.0000           +0.0000j            1.0000            0.0000
-----------------------------------------------------------------------------
```

The model summary reveals a lot of information. The table in the middle is the coefficients table where the values under 'coef' are the weights of the respective terms.

```python
print(results_AR.summary())
```

# Forecasting

We will use the following code snippet to Forecast the Economic Growth of India obtained in a table form

```
hsd = []
e = int(Forecast_Years)
  tf = results_AR.forecast(e)
  tf = np.exp(tf[0])
  hsd.append(tf)
```

| Years | ECONOMICS:INGDPPCP | ECONOMICS:SAGDPPCP |
|---|---|---|
| 2022 | 7026.589989 | 45076.437847 |
| 2023 | 7377.675262 | 45187.432025 |
| 2024 | 7735.405362 | 45290.319639 |
| 2025 | 8103.484302 | 45380.192358 |
| 2026 | 8484.321420 | 45483.597097 |
| ... | ... | ... |
| 2096 | 194799.635187 | 52853.479101 |
| 2097 | 203710.861154 | 52967.029587 |
| 2098 | 213029.736490 | 53080.824025 |
| 2099 | 222774.909359 | 53194.862940 |
| 2100 | 232965.880997 | 53309.146856 |

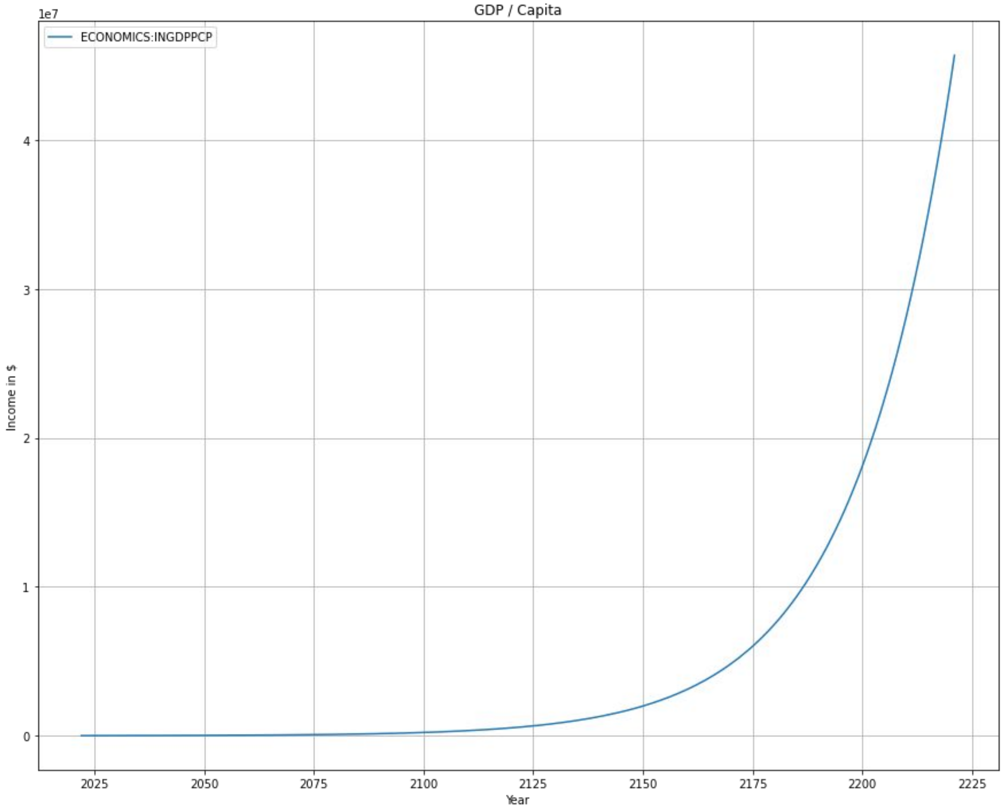
79 rows × 2 columns

# Plotting the results

Plotting the growth trajectory of India's GDP

Growth for a period of 200 years with a lag of 1

```
SG.plot(label = 'ticker')
plt.title('GDP / Capita')
plt.xlabel('Year')
plt.ylabel('Income in $')
plt.legend(loc= 'upper left')
plt.grid(True)
plt.show()
```



Thanks for a patient hearing,

Hope you enjoyed it as much as we did while working over it!

References-


- Google Images
- Wikipedia
- Machine Learning plus
- Geeks for Geeks