FORECASTING GDP OF INDIA AND ITS VARIOUS FACTORS

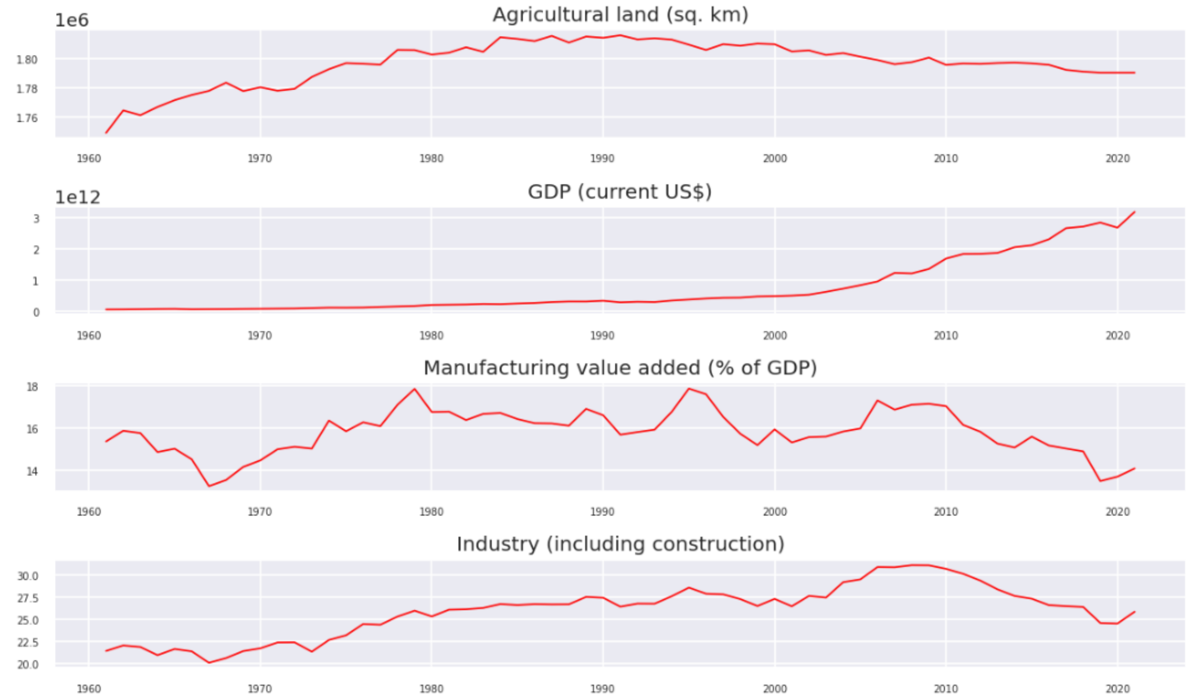
Sreeja Paul(J016)

Hemansi Kevadiya(J051)

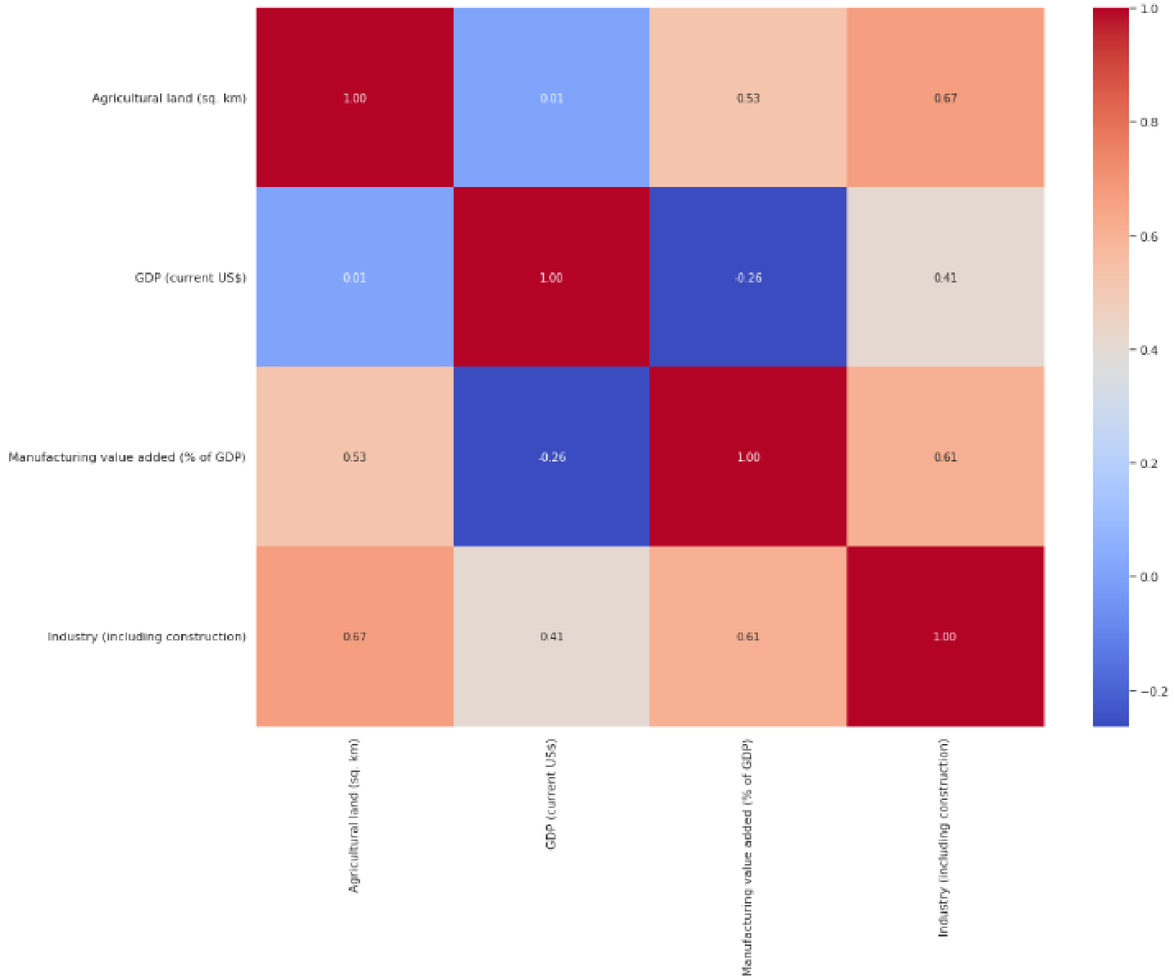
Anuj Garg(J052)

Labdhi Joshi(J064)

Time series analysis is an approach to forecasting commonly used in business to produce and improve point forecasts where regression falls short. Time series forecasting is increasingly in demand due to its ability to predict events based solely on previously observed data of the given event. GDP measures the monetary value of final goods and services—that is, those that are bought by the final user—produced in a country in a given period of time(say a quarter or a year).The three largest economies in the world as measured by nominal GDP are the United States, China, and Japan. Economic growth and prosperity are impacted by a wide array of factors, namely investment in workforce education, production output (as determined by investment in physical capital), natural resources, and entrepreneurship. The economies of the U.S., China, and Japan all have a unique combination of these factors that have led to economic growth over time, as outlined below.



CORRELATION HEATMAP:



TRANSFORMATION

Transformation is a method of transforming a time series dataset. It can be used to remove the series dependence on time, so-called temporal dependence. This includes structures like trends and seasonality. We had to perform transformation till order 3 for making the time series stationary.

ADF TEST

In statistics and econometrics, an augmented Dickey–Fuller test (ADF) tests the null hypothesis that a unit root is present in a time series sample. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity. It is an augmented version of the Dickey–Fuller test for a larger and more complicated set of time series models. The augmented Dickey–Fuller (ADF) statistic, used in the test, is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.

TO TEST IF THE TIME SERIES IS STATIONARY OR NO?

In [14]:

```
def adf_test(series,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """
    print(f'Augmented Dickey-Fuller Test: {title}')
    result = adfuller(series.dropna(),autolag='AIC') # .dropna() handles differenced data
    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)
    for key,val in result[4].items():
        out[f'critical value ({key})']=val
    print(out.to_string())          # .to_string() removes the line "dtype: float64"
    if result[1] <= 0.05:
        print("Strong evidence against the null hypothesis")
        print("Reject the null hypothesis")
        print("Data has no unit root and is stationary")
    else:
        print("Weak evidence against the null hypothesis")
        print("Fail to reject the null hypothesis")
        print("Data has a unit root and is non-stationary")
```

```
adf_test(df['Agricultural land (sq. km)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic      -3.103007
p-value                  0.026331
# lags used             11.000000
# observations          49.000000
critical value (1%)     -3.571472
critical value (5%)     -2.922629
critical value (10%)    -2.599336
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```
1 adf_test(df['GDP (current US$)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic        2.111950
p-value                   0.998802
# lags used              10.000000
# observations           50.000000
critical value (1%)      -3.568486
critical value (5%)      -2.921360
critical value (10%)     -2.598662
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data has a unit root and is non-stationary
```

```
1 adf_test(df['Manufacturing value added (% of GDP)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -1.990633
p-value                   0.290695
# lags used               0.000000
# observations           60.000000
critical value (1%)      -3.544369
critical value (5%)      -2.911073
critical value (10%)     -2.593190
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data has a unit root and is non-stationary
```

```
1 adf_test(df['Industry (including construction)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -1.602416
p-value                   0.482380
# lags used               0.000000
# observations           60.000000
critical value (1%)      -3.544369
critical value (5%)      -2.911073
critical value (10%)     -2.593190
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data has a unit root and is non-stationary
```

AFTER 3rd ORDER Of DIFFERENCING

From the result that we got We can clearly see that all the features are stationary leaving GDP as P value for GDP is still less than 0.05.

So, Now We apply 3rd order of Transformation on the entire data

We have performed 3rd Order of Transformation followed by ADF Test to check if the Time Series has become stationary or not

```python
df_transformed = df_transformed.diff().dropna()
```

```python
adf_test(df_transformed['Agricultural land (sq. km)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.336356
p-value                   0.000005
# lags used              11.000000
# observations           46.000000
critical value (1%)      -3.581258
critical value (5%)      -2.926785
critical value (10%)     -2.601541
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```python
adf_test(df_transformed['GDP (current US$)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -4.797248
p-value                   0.000055
# lags used              11.000000
# observations           46.000000
critical value (1%)      -3.581258
critical value (5%)      -2.926785
critical value (10%)     -2.601541
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```python
adf_test(df_transformed['Manufacturing value added (% of GDP)'])
```

```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.605125
p-value                   0.000001
# lags used               6.000000
# observations           51.000000
critical value (1%)      -3.565624
critical value (5%)      -2.920142
critical value (10%)     -2.598015
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

```python
adf_test(df_transformed['Industry (including construction)'])
```

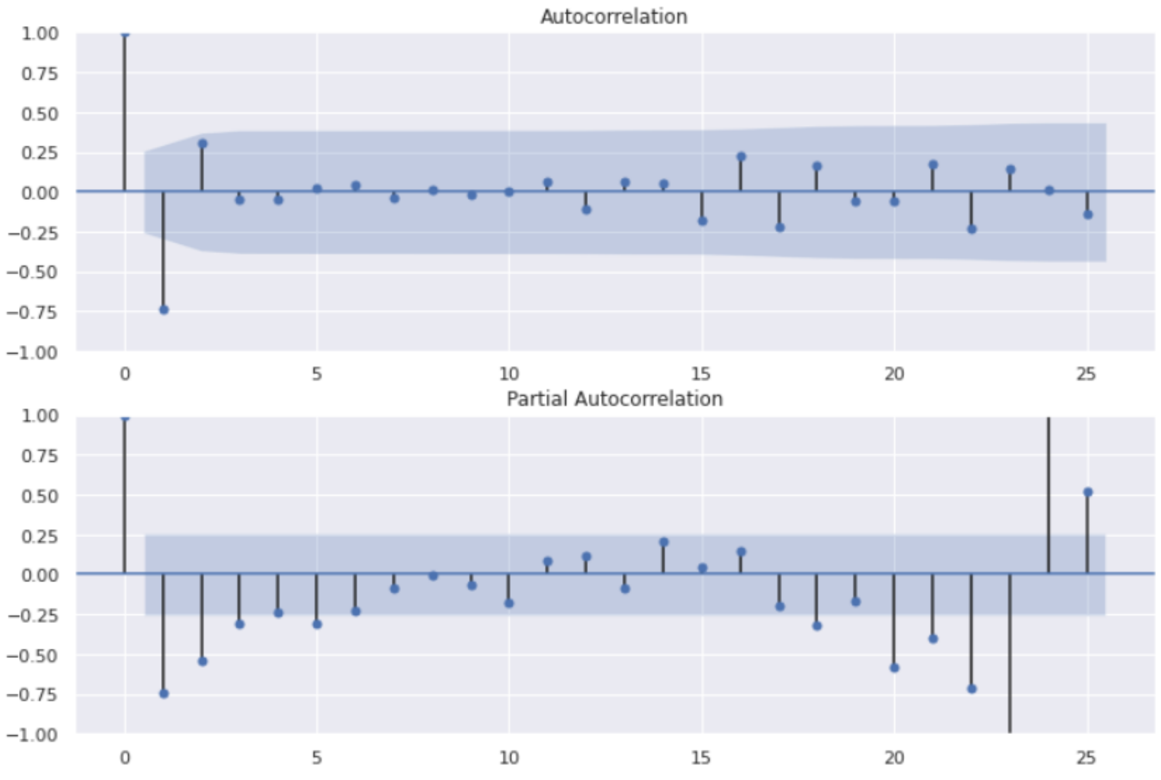
```
Augmented Dickey-Fuller Test:
ADF test statistic       -5.151497
p-value                   0.000011
# lags used               6.000000
# observations           51.000000
critical value (1%)      -3.565624
critical value (5%)      -2.920142
critical value (10%)     -2.598015
Strong evidence against the null hypothesis
Reject the null hypothesis
Data has no unit root and is stationary
```

ACF

Autocorrelation analysis is an important step in the Exploratory Data Analysis of time series forecasting. The autocorrelation analysis helps detect patterns and check for randomness. It’s especially important when you intend to use an autoregressive–moving-average (ARMA) model for forecasting because it helps to determine its parameters. The analysis involves looking at the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.

PACF

In the estimation of the ARMA parameter spectrum, the AR parameters are first estimated, and then the MA parameters are estimated based on these AR parameters. The spectral estimates of the ARMA model are then obtained. The parameter estimation of the MA model is therefore often calculated as a process of ARMA parameter spectrum association. It is used in mechanical parts like gears to form fault diagnosis and analysis since it can process separate sinusoidal signal frequencies.



ARIMA MODEL

We use this algorithm to get (p,q) values.

```python
from pmdarima.arima import auto_arima

arima_model=auto_arima(datanew,start_p=1,d=1,start_q=1,
                       max_p=5,max_q=5,max_d=5,m=12,
                       start_P=0,D=1,start_Q=0,max_P=5,max_D=5,max_Q=5,
                       seasonal=True,
                       trace=True,
                       error_action="ignore",
                       suppress_warnings=True,
                       stepwise=True,n_fits=50)
```

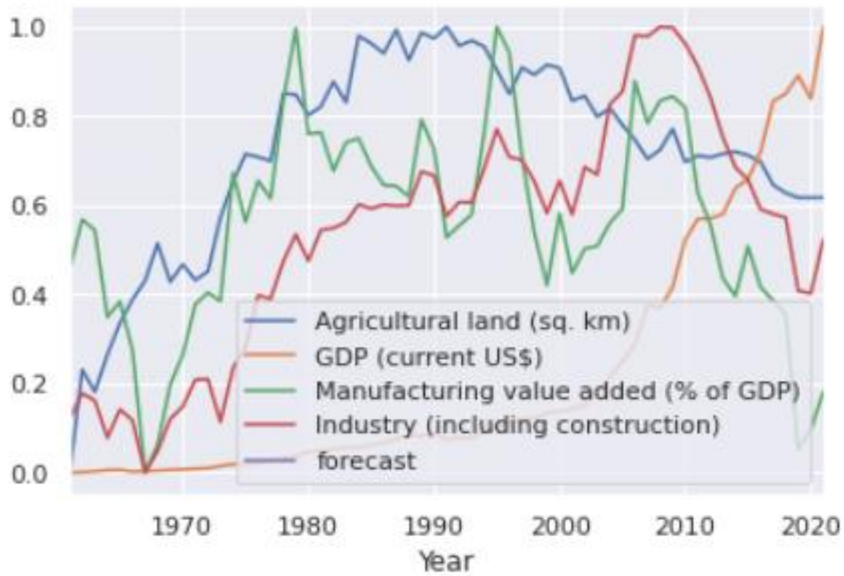
```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=inf, Time=0.50 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=280.478, Time=0.05 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=223.293, Time=0.17 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=0.60 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=237.598, Time=0.04 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=221.418, Time=0.57 sec
 ARIMA(1,1,0)(3,1,0)[12]             : AIC=222.886, Time=0.64 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=223.072, Time=0.42 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=222.730, Time=0.15 sec
 ARIMA(1,1,0)(3,1,1)[12]             : AIC=inf, Time=2.73 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=266.744, Time=0.23 sec
 ARIMA(2,1,0)(2,1,0)[12]             : AIC=191.385, Time=0.31 sec
 ARIMA(2,1,0)(1,1,0)[12]             : AIC=190.914, Time=0.13 sec
 ARIMA(2,1,0)(0,1,0)[12]             : AIC=204.351, Time=0.06 sec
 ARIMA(2,1,0)(1,1,1)[12]             : AIC=191.687, Time=0.24 sec
 ARIMA(2,1,0)(0,1,1)[12]             : AIC=192.329, Time=0.21 sec
 ARIMA(2,1,0)(2,1,1)[12]             : AIC=inf, Time=1.66 sec
 ARIMA(3,1,0)(1,1,0)[12]             : AIC=181.627, Time=0.15 sec
 ARIMA(3,1,0)(0,1,0)[12]             : AIC=193.132, Time=0.06 sec
 ARIMA(3,1,0)(2,1,0)[12]             : AIC=182.503, Time=0.58 sec
 ARIMA(3,1,0)(1,1,1)[12]             : AIC=182.513, Time=0.52 sec
 ARIMA(3,1,0)(0,1,1)[12]             : AIC=182.238, Time=0.47 sec
 ARIMA(3,1,0)(2,1,1)[12]             : AIC=184.492, Time=1.48 sec
 ARIMA(4,1,0)(1,1,0)[12]             : AIC=179.526, Time=0.50 sec
 ARIMA(4,1,0)(0,1,0)[12]             : AIC=191.044, Time=0.17 sec
 ARIMA(4,1,0)(2,1,0)[12]             : AIC=180.361, Time=0.82 sec
 ARIMA(4,1,0)(1,1,1)[12]             : AIC=180.535, Time=0.61 sec
 ARIMA(4,1,0)(0,1,1)[12]             : AIC=180.197, Time=0.55 sec
 ARIMA(4,1,0)(2,1,1)[12]             : AIC=182.311, Time=2.04 sec
 ARIMA(5,1,0)(1,1,0)[12]             : AIC=179.645, Time=0.62 sec
 ARIMA(4,1,1)(1,1,0)[12]             : AIC=inf, Time=2.25 sec
 ARIMA(3,1,1)(1,1,0)[12]             : AIC=inf, Time=1.77 sec
 ARIMA(5,1,1)(1,1,0)[12]             : AIC=inf, Time=2.11 sec
 ARIMA(4,1,0)(1,1,0)[12] intercept   : AIC=181.174, Time=0.60 sec

Best model:  ARIMA(4,1,0)(1,1,0)[12]
Total fit time: 24.156 seconds
```

ARIMA MODEL FORECAST

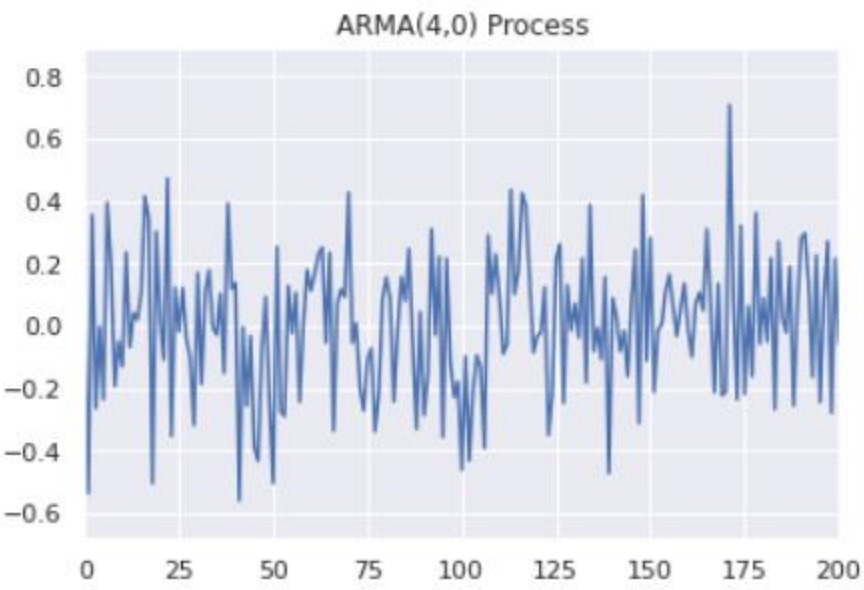
```
df_norm = (df - df.min())/(df.max()- df.min())
df_norm.plot()
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x7f708cca5050>
```



ARMA MODEL

ARMA is a model of forecasting in which the methods of autoregression (AR) analysis and moving average (MA) are both applied to time-series data that is well behaved. In ARMA it is assumed that the time series is stationary and when it fluctuates, it does so uniformly around a particular time.



AR MODEL

AR model is commonly used in current spectrum estimation.

MA Model

It is a commonly used model in modern spectrum estimation and is also one of the methods of model parametric spectrum analysis. In the estimation of the ARMA parameter spectrum, the AR parameters are first estimated, and then the MA parameters are estimated based on these AR parameters. The spectral estimates of the ARMA model are then obtained. The parameter estimation of the MA model is therefore often calculated as a process of ARMA parameter spectrum association. It is used in mechanical parts like gears to form fault diagnosis and analysis since it can process separate sinusoidal signal frequencies.

VECTOR AUTOREGRESSIVE MODELS VAR(p) MODELS

VAR models (vector autoregressive models) are used for multivariate time series. The structure is that each variable is a linear function of past lags of itself and past lags of the other variables.

In general, for a VAR(p) model, the first p lags of each variable in the system would be used as regression predictors for each variable.

VAR models are a specific case of more general VARMA models. VARMA models for multivariate time series include the VAR structure above along with moving average terms for each variable. More generally yet, these are special cases of ARMAX models that allow for the addition of other predictors that are outside the multivariate set of principal interest.
For example, we could use a VAR model to show how real GDP is a function of policy rate and how policy rate is, in turn, a function of real GDP.
VAR models are traditionally widely used in finance and econometrics.

ORDERS:

```
ORDER1
AIC67.3076302502678

ORDER2
AIC66.43113174389815

ORDER3
AIC66.30747056978653

ORDER4
AIC66.6185382794587

ORDER5
AIC65.96030617524256

ORDER6
AIC65.14674274601077

ORDER7
AIC63.58713321567597

ORDER8
AIC56.468874564235506
```

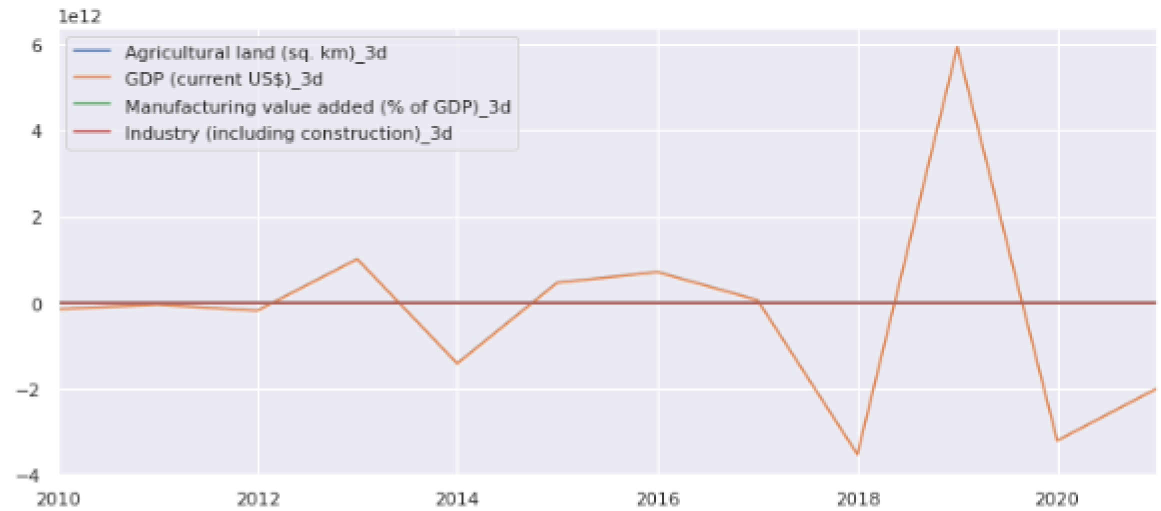
RESULT SUMMARY:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Fri, 04, Nov, 2022
Time:                     14:19:22
--------------------------------------------------------------------
No. of Equations:         4.00000    BIC:                    62.1573
Nobs:                     38.0000    HQIC:                   58.4928
Log likelihood:          -1156.59    FPE:                1.30638e+26
AIC:                      56.4689    Det(Omega_mle):     1.07194e+25
--------------------------------------------------------------------
Results for equation Agricultural land (sq. km)
==============================================================================================================
                                              coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------------------------------------
const                                         -138.931657       540.727312           -0.257           0.797
L1.Agricultural land (sq. km)                   -1.735541         0.227362           -7.633           0.000
L1.GDP (current US$)                             0.000000         0.000000            0.601           0.548
```

FORECASTING:

```
1  df_forecast.plot(figsize=(12,5))
```

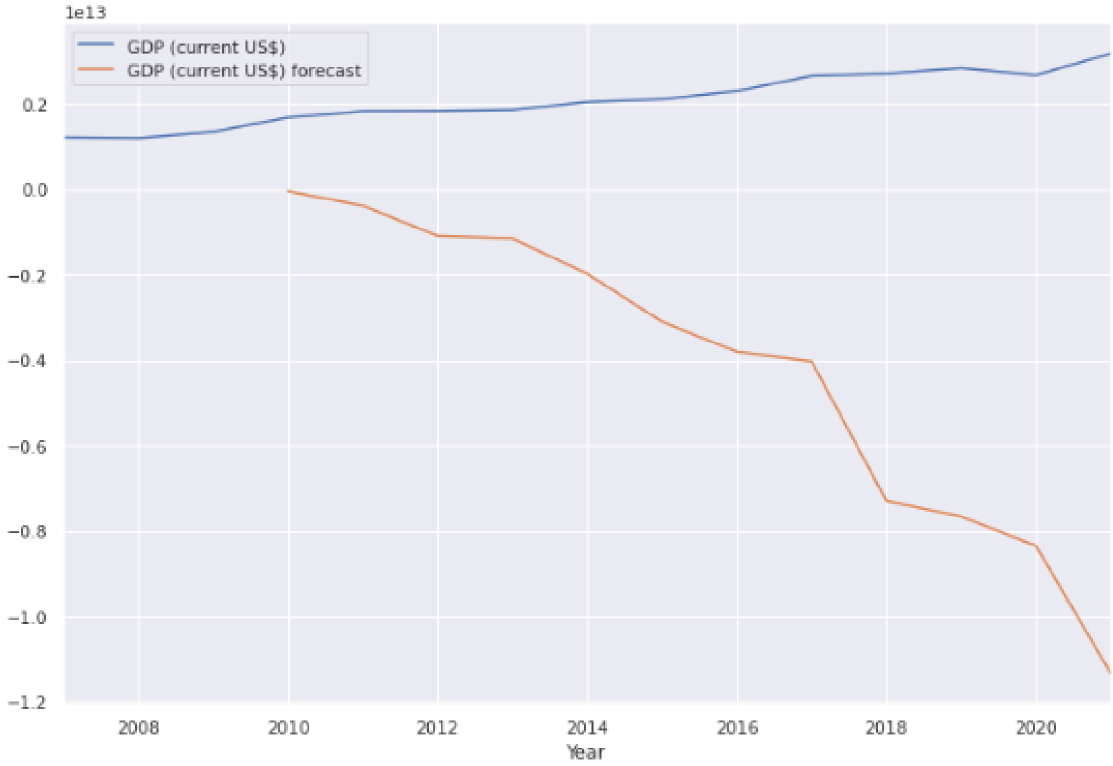
<matplotlib.axes._subplots.AxesSubplot at 0x7f708c9583d0>



FOR GDP:

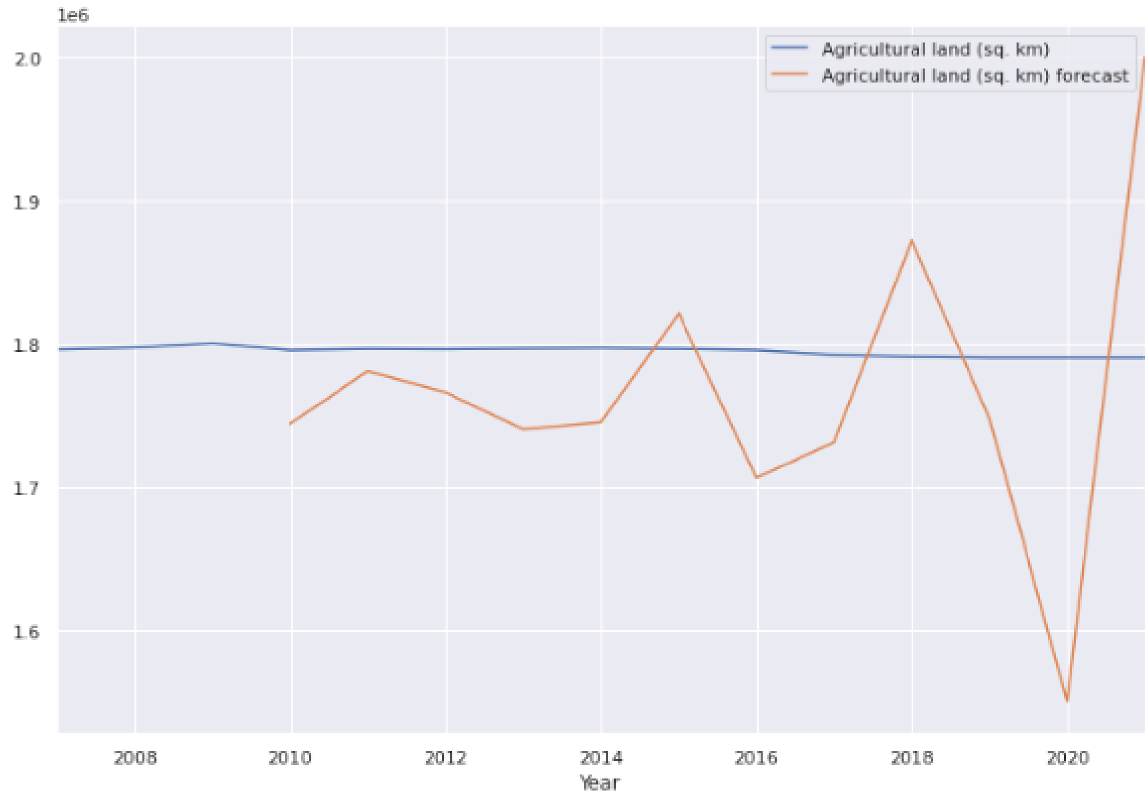
```
test_range['GDP (current US$)'].plot(figsize=(12,8),legend=True)
df_forecast['GDP (current US$) forecast'].plot(legend=True)
```

]: <matplotlib.axes._subplots.AxesSubplot at 0x7f708ca17590>



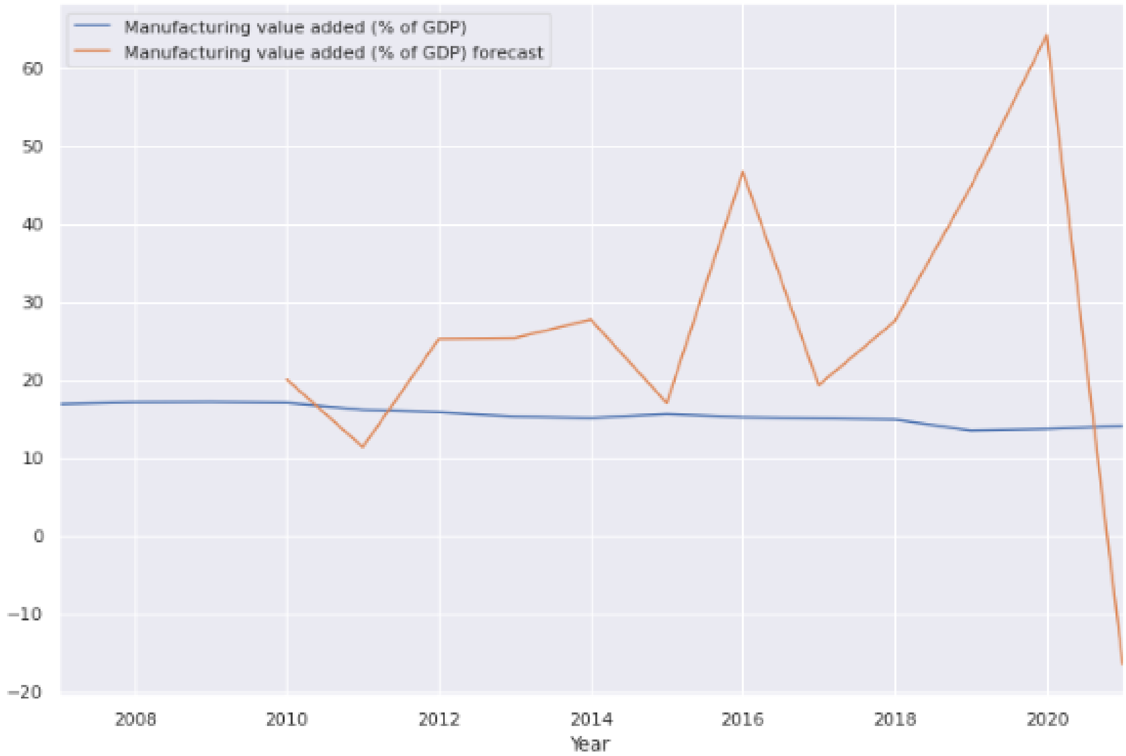
```
test_range['Agricultural land (sq. km)'].plot(figsize=(12,8),legend=True
df_forecast['Agricultural land (sq. km) forecast'].plot(legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f708cd081d0>
```



```
1  test_range['Manufacturing value added (% of GDP)'].plot(figsize=(12,8),l
2  df_forecast['Manufacturing value added (% of GDP) forecast'].plot(legend
```

]: <matplotlib.axes._subplots.AxesSubplot at 0x7f708ccc1d90>



```
1  test_range['Industry (including construction)'].plot(figsize=(12,8),lege
2  df_forecast['Industry (including construction) forecast'].plot(legend=Tr
```

]: <matplotlib.axes._subplots.AxesSubplot at 0x7f708c8d53d0>