

```
In [ ] : !pip install statsmodels --upgrade

Collecting statsmodels
  Downloading https://files.pythonhosted.org/packages/be/4c/9e2435ca6645d6bafa2b5bb11f6e365b2b934a2f7e9d6e339d67138926d/statsmodels-0.12.1-cp36-cp38m-manylinux1_x86_64.whl (9.5MB)
    |#####| 9.5MB 6.0MB/s
  Requirement already satisfied, skipping upgrade: patsy>=0.5 in /usr/local/lib/python3.6/dist-packages (from statsmodels) (0.5.1)
  Requirement already satisfied, skipping upgrade: scipy>=1.1 in /usr/local/lib/python3.6/dist-packages (from statsmodels) (1.4.1)
  Requirement already satisfied, skipping upgrade: pandas>=0.21 in /usr/local/lib/python3.6/dist-packages (from statsmodels) (1.1.5)
  Requirement already satisfied, skipping upgrade: numpy>=1.15 in /usr/local/lib/python3.6/dist-packages (from statsmodels) (1.19.4)
  Requirement already satisfied, skipping upgrade: python-dateutil<=2.7.3 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21->statsmodels) (2.8.1)
  Installing collected packages: statsmodels
    Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
    Successfully installed statsmodels-0.12.1
```

```
In [ ] : import pandas as pd
import numpy as np
from matplotlib import pyplot
from statsmodels.tsa.ar_model import AutoReg
```

```
In [ ] : from google.colab import files
uploaded = files.upload()

Choose files to be uploaded:
Upload files is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
Saving Covid_Pakistan.csv to Covid_Pakistan.csv

In [ ] : data=pd.read_csv('Covid_Pakistan.csv')
data
```

```
Out [ ] :
      date  total_cases  new_cases  total_deaths  new_deaths  total_tests  new_tests
0  31/12/2019           0           0           0           0           0           0.0
1  01/01/2020           0           0           0           0           0           0.0
2  02/01/2020           0           0           0           0           0           0.0
3  03/01/2020           0           0           0           0           0           0.0
4  04/01/2020           0           0           0           0           0           0.0
...      ...
327 22/11/2020       374173       2665       7662       59  5180026       38623.0
328 23/11/2020       376929       2756       7696       34  5180955       36929.0
329 24/11/2020       379883       2954       7744       48  5256120       39165.0
330 25/11/2020       382892       3009       7803       59  5297703       41583.0
331 26/11/2020       386198       3206       7843       40  5343702       46999.0

332 rows x 7 columns
```

```
In [ ] : cases=pd.DataFrame(data,columns=['total_cases'])
cases
```

```
Out [ ] :
      total_cases
0           0
1           0
2           0
3           0
4           0
...      ...
327       374173
328       376929
329       379883
330       382892
331       386198

332 rows x 1 columns
```

```
In [ ] : x=cases.values
y=data['date'].values
```

```
In [ ] : from statsmodels.tsa.stattools import adfuller
df_test=adfuller(cases['total_cases'],autolag='AIC')
print("1. ADF : ",df_test[0])
print("2. P-Value : ",df_test[1])
print("3. Num Of Lags : ",df_test[2])
print("4. Num Of Observations Used For ADF Regression : ",df_test[3])
print("5. Critical Values : ",df_test[4])
for key, val in df_test[4].items():
    print("\t",key," : ",val)
```

```
1. ADF : 0.10768516499141628
2. P-Value : 0.9665819959543044
3. Num Of Lags : 12
4. Num Of Observations Used : 319
5. Critical Values : {'1%' : -3.4510167751522642, '5%' : -2.87064334231426, '10%' : -2.5716201744283174}
1% : -3.4510167751522642
5% : -2.87064334231426
10% : -2.5716201744283174
```

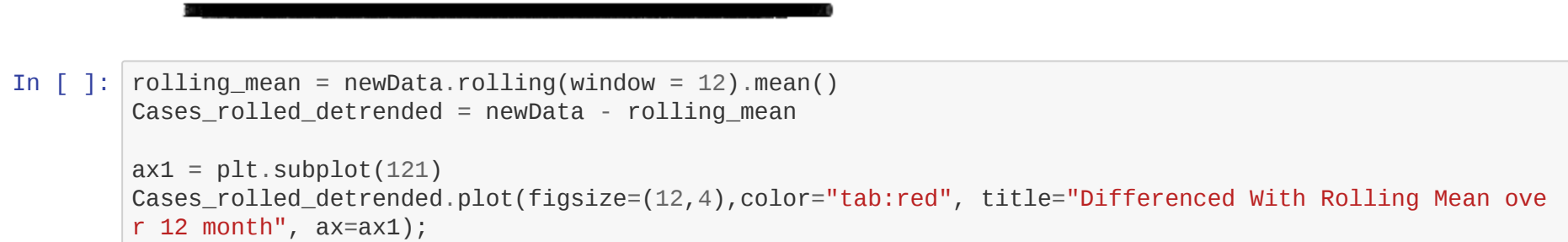
```
In [ ] : import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
In [ ] : newData=pd.DataFrame(X,Y)

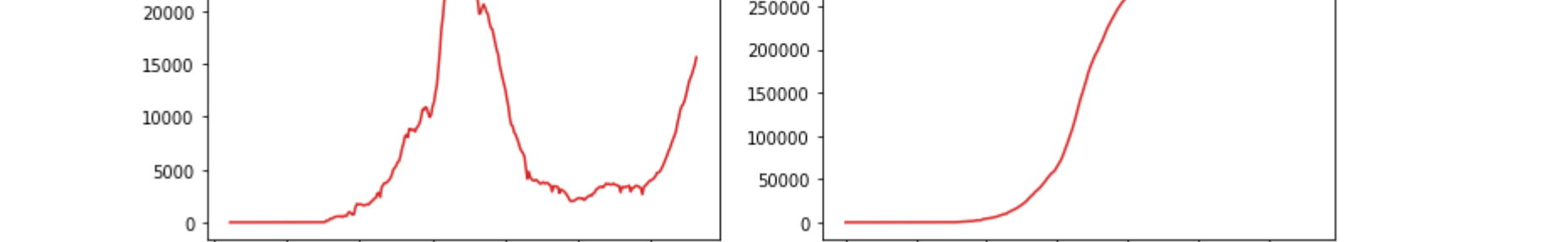
newData=newData.rename(columns={'0':'Cases'})
```

```
In [ ] : logged_newData = newData["Cases"].apply(lambda x : np.log(x))

ax1 = plt.subplot(121)
logged_newData.plot(figsize=(12,4), color="tab:red", title="Log Transformed Values", ax=ax1);
ax2 = plt.subplot(122)
cases.plot(color="tab:red", title="Original Values", ax=ax2);
```

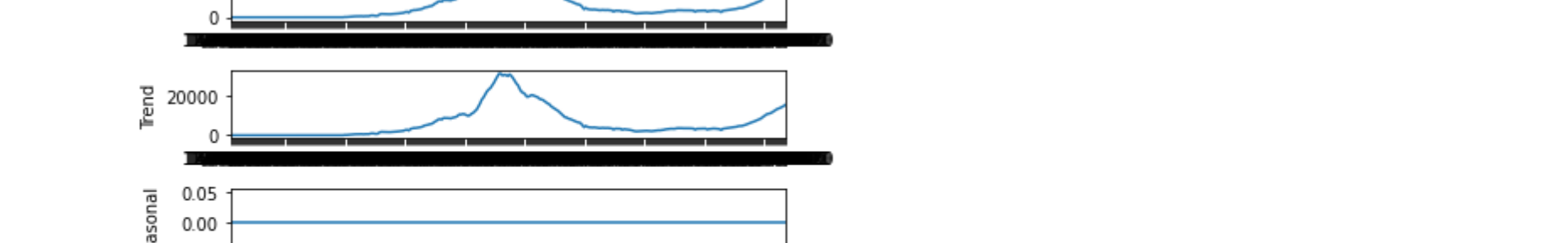


```
In [ ] : decompose_result =seasonal_decompose(newData,period=1)
decompose_result.plot();
```

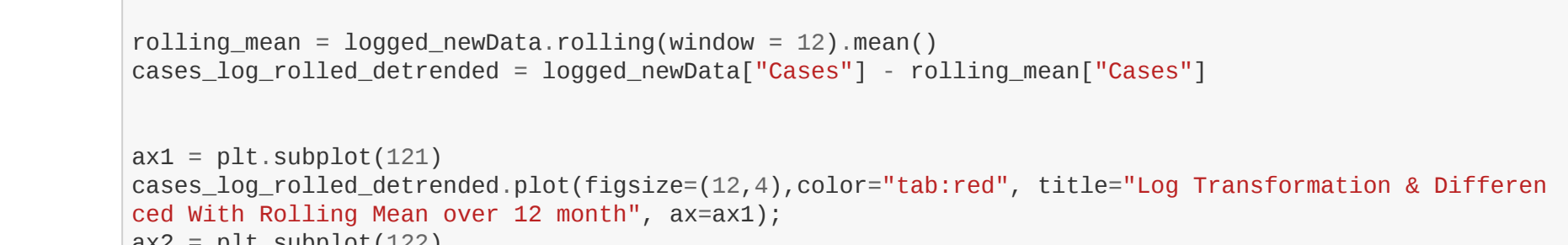


```
In [ ] : powered_newData = newData["Cases"].apply(lambda x : x ** 0.5)

ax1 = plt.subplot(121)
powered_newData.plot(figsize=(12,4), color="tab:red", title="Powered Transformed Values", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```

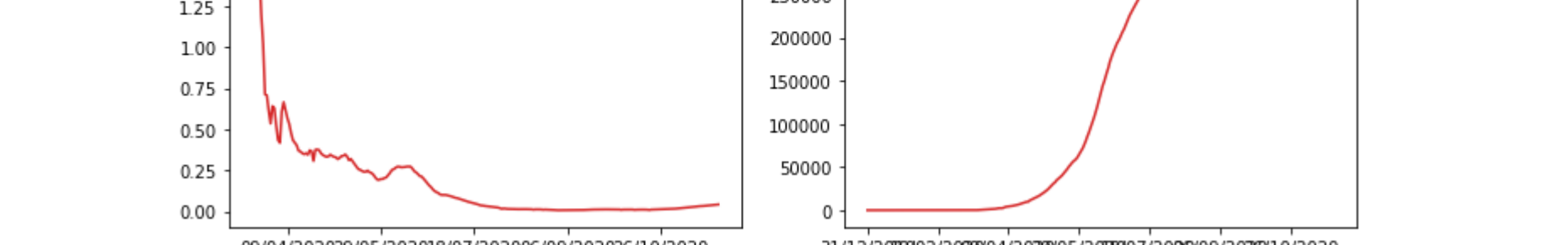


```
In [ ] : decompose_result = seasonal_decompose(powered_newData,period=1)
decompose_result.plot();
```



```
In [ ] : rolling_mean = newData.rolling(window = 12).mean()
cases_rolled_detrended = newData - rolling_mean

ax1 = plt.subplot(121)
cases_rolled_detrended.plot(figsize=(12,4),color="tab:red", title="Differenced With Rolling Mean over 12 month", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



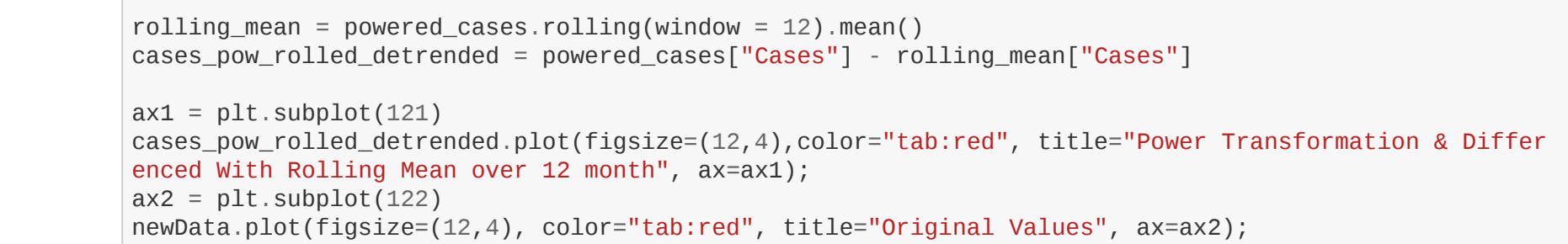
```
In [ ] : decompose_result = seasonal_decompose(cases_rolled_detrended.dropna(),period=1)
decompose_result.plot();
```



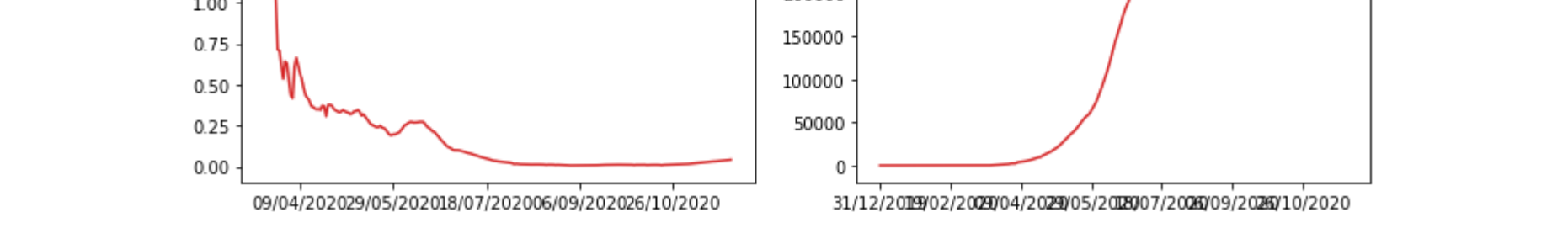
```
In [ ] : logged_newData = pd.DataFrame(newData["Cases"]).apply(lambda x : np.log(x))

rolling_mean = logged_newData.rolling(window = 12).mean()
cases_log_rolled_detrended = logged_newData["Cases"] - rolling_mean["Cases"]

ax1 = plt.subplot(121)
cases_log_rolled_detrended.plot(figsize=(12,4),color="tab:red", title="Log Transformation & Differenced With Rolling Mean over 12 month", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



```
In [ ] : decompose_result = seasonal_decompose(cases_log_rolled_detrended.dropna(),period=1)
decompose_result.plot();
```



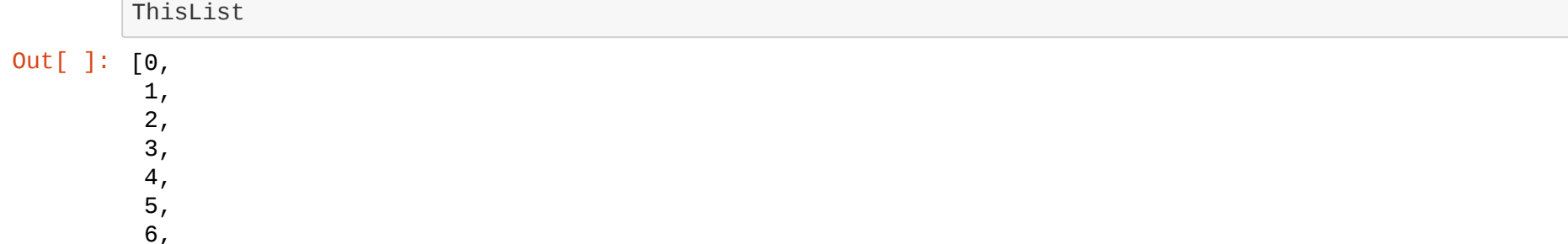
```
In [ ] : powered_cases = pd.DataFrame(newData["Cases"]).apply(lambda x : np.log(x))

rolling_mean = powered_cases.rolling(window = 12).mean()
cases_pow_rolled_detrended = powered_cases["Cases"] - rolling_mean["Cases"]

ax1 = plt.subplot(121)
cases_pow_rolled_detrended.plot(figsize=(12,4),color="tab:red", title="Power Transformation & Differenced With Rolling Mean over 12 month", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



```
In [ ] : decompose_result = seasonal_decompose(cases_pow_rolled_detrended.dropna(),period=1)
decompose_result.plot();
```



```
In [ ] : ThisList=[]
for i in range(newData.shape[0]):
    ThisList.append(i)
ThisList
```

```
Out [ ] :
0,
1,
2,
3,
4,
5,
6,
7,
8,
9,
10,
11,
12,
13,
14,
15,
16,
17,
18,
19,
20,
21,
22,
23,
24,
25,
26,
27,
28,
29,
30,
31,
32,
33,
34,
35,
36,
37,
38,
39,
40,
41,
42,
43,
44,
45,
46,
47,
48,
49,
50,
51,
52,
53,
54,
55,
56,
57,
58,
59,
60,
61,
62,
63,
64,
65,
66,
67,
68,
69,
70,
71,
72,
73,
74,
75,
76,
77,
78,
79,
80,
81,
82,
83,
84,
85,
86,
87,
88,
89,
90,
91,
92,
93,
94,
95,
96,
97,
98,
99,
100,
101,
102,
103,
104,
105,
106,
107,
108,
109,
110,
111,
112,
113,
114,
115,
116,
117,
118,
119,
120,
121,
122,
123,
124,
125,
126,
127,
128,
129,
130,
131,
132,
133,
134,
135,
136,
137,
138,
139,
140,
141,
142,
143,
144,
145,
146,
147,
148,
149,
150,
151,
152,
153,
154,
155,
156,
157,
158,
159,
160,
161,
162,
163,
164,
165,
166,
167,
168,
169,
170,
171,
172,
173,
174,
175,
176,
177,
178,
179,
180,
181,
182,
183,
184,
185,
186,
187,
188,
189,
190,
191,
192,
193,
194,
195,
196,
197,
198,
199,
200,
201,
202,
203,
204,
205,
206,
207,
208,
209,
210,
211,
212,
213,
214,
215,
216,
217,
218,
219,
220,
221,
222,
223,
224,
225,
226,
227,
228,
229,
230,
231,
232,
233,
234,
235,
236,
237,
238,
239,
240,
241,
242,
243,
244,
245,
246,
247,
248,
249,
250,
251,
252,
253,
254,
255,
256,
257,
258,
259,
260,
261,
262,
263,
264,
265,
266,
267,
268,
269,
270,
271,
272,
273,
274,
275,
276,
277,
278,
279,
280,
281,
282,
283,
284,
285,
286,
287,
288,
289,
290,
291,
292,
293,
294,
295,
296,
297,
298,
299,
300,
301,
302,
303,
304,
305,
306,
307,
308,
309,
310,
311,
312,
313,
314,
315,
316,
317,
318,
319,
320,
321,
322,
323,
324,
325,
326,
327,
328,
329,
330,
331,
332
```

```
In [ ] : from statsmodels.regression.linear_model import OLS
least_squares = OLS(newData["Cases"].values , ThisList)
result = least_squares.fit()
```

```
fit = pd.Series(result.predict(ThisList), index = newData.index)
cases_ols_detrended= newData["Cases"] - fit

ax1 = plt.subplot(121)
cases_ols_detrended.plot(figsize=(12,4), color="tab:red", title="Linear Regression Fit", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



```
In [ ] : logged_newData_diff = logged_newData - logged_newData.shift()

logged_newData_diff.plot(figsize=(12,4), color="tab:red", title="Log-Transformed & Differenced Time-Series", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



```
In [ ] : logged_newData_diff=logged_newData_diff.replace([np.inf, -np.inf], np.nan)
logged_newData_diff=logged_newData_diff.dropna()
```

```
In [ ] : df_test = adfuller(logged_newData_diff["Cases"].values, autolag = 'AIC')

print("1. ADF : ",df_test[0])
print("2. P-Value : ",df_test[1])
print("3. Num Of Lags : ",df_test[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation : ",df_test[3])
print("5. Critical Values : ")
for key, val in df_test[4].items():
    print("\t",key," : ",val)
```

```
1. ADF : -5.887239947861143
2. P-Value : 2.98164559473934e-07
3. Num Of Lags : 15
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 249
5. Critical Values : {'1%' : -3.4568861317725864, '5%' : -2.8732185133016057, '10%' : -2.5729936189738876}
1% : -3.4568861317725864
5% : -2.8732185133016057
10% : -2.5729936189738876
```

```
In [ ] : powered_newData_diff = powered_newData - powered_newData.shift()

ax1 = plt.subplot(121)
powered_newData_diff.plot(figsize=(12,4), color="tab:red", title="Power-Transformed & Differenced Time-Series", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```



```
In [ ] : df_test = adfuller(powered_newData_diff.dropna().values, autolag = 'AIC')

print("1. ADF : ",df_test[0])
print("2. P-Value : ",df_test[1])
print("3. Num Of Lags : ",df_test[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation : ",df_test[3])
print("5. Critical Values : ")
for key, val in df_test[4].items():
    print("\t",key," : ",val)
```

```
1. ADF : -1.694355228641399
2. P-Value : 0.434082315460836394
3. Num Of Lags : 10
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 320
5. Critical Values : {'1%' : -3.456951662284933, '5%' : -2.8761475789808784, '10%' : -2.571664931640625}
1% : -3.456951662284933
5% : -2.8761475789808784
10% : -2.571664931640625
```

```
In [ ] : cases_rolled_detrended_diff = Cases_rolled_detrended - Cases_rolled_detrended.shift()

ax1 = plt.subplot(121)
cases_rolled_detrended_diff.plot(figsize=(8,4), color="tab:red", title="Rolled & Differenced Time-Series", ax=ax1);
ax2 = plt.subplot(122)
newData.plot(figsize=(12,4), color="tab:red", title="Original Values", ax=ax2);
```




```
In [ ]: std_err = std_err + (std_err - std_err.mean())**2
print("AIC:", df_test[0])
print("BIC:", df_test[1])
print("3. Num Of Lags :", df_test[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation :", df_test[3])
print("5. Critical Values :", df_test[4])
for key, val in df_test[4].items():
    print("%10s, %10s, %10s" % (key, val, val))

1. ADF : -3.59635852681393
2. P-value : 0.00583182362791723
3. Num Of Lags : 8
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 311
5. Critical Values :
    1% : -3.451552879535732
    5% : -2.870876756338407
    10% : -2.57145666091128
```

```
In [ ]: print(cases_rolled_detrended_diff.dropna())

Cases
12/01/2020    0.000000
13/01/2020    0.000000
14/01/2020    0.000000
15/01/2020    0.000000
16/01/2020    0.000000
22/11/2020    356.916667
23/11/2020    360.583333
24/11/2020    403.003222
25/11/2020    459.333333
26/11/2020    661.250000

[320 rows x 1 columns]
```

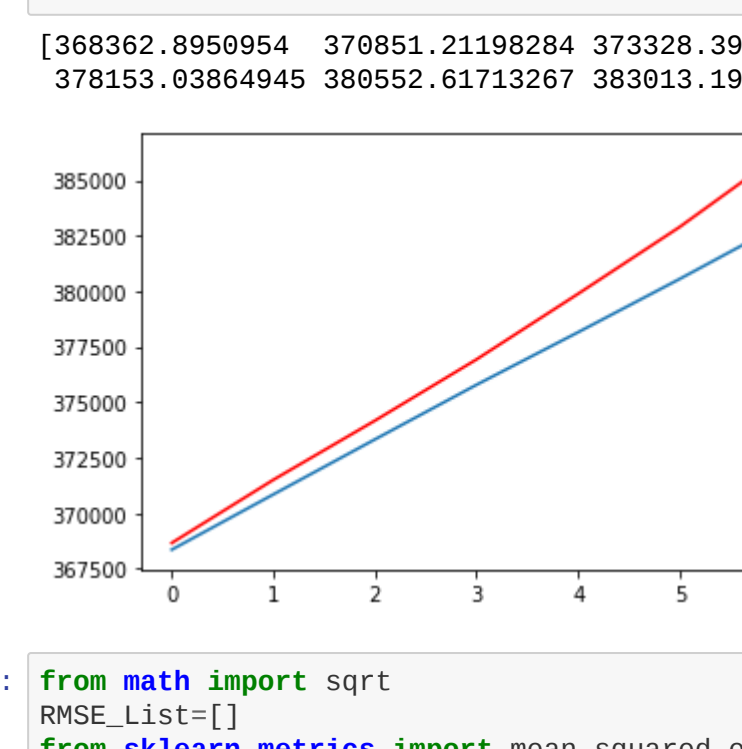
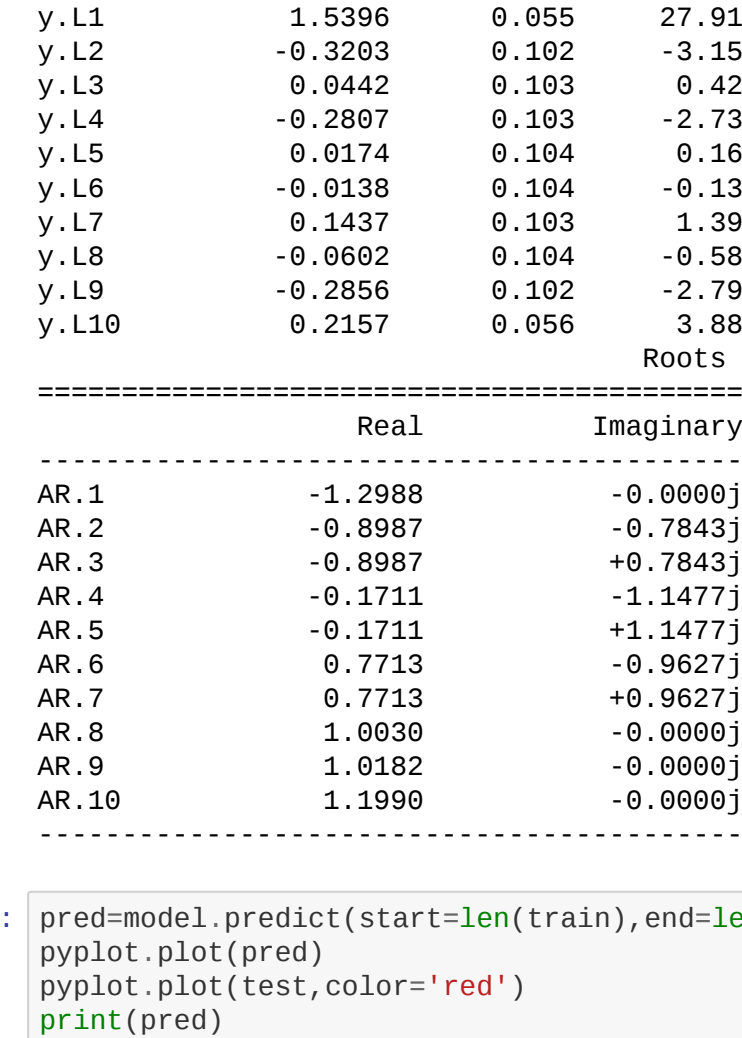
```
In [ ]: cases_rolled_detrended_diff=cases_rolled_detrended_diff.dropna()
newData=cases_rolled_detrended_diff
```

1.Auto Regression

```
In [ ]: from statsmodels.tsa.stattools import adfuller
df_test=adfuller(newData['Cases'],autolag='AIC')
print("AIC:", df_test[0])
print("BIC:", df_test[1])
print("3. Num Of Lags :", df_test[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation :", df_test[3])
print("5. Critical Values :", df_test[4])
for key, val in df_test[4].items():
    print("%10s, %10s, %10s" % (key, val, val))

1. ADF : -3.59635852681393
2. P-value : 0.00583182362791723
3. Num Of Lags : 8
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 311
5. Critical Values :
    1% : -3.451552879535732
    5% : -2.870876756338407
    10% : -2.57145666091128
```

```
In [ ]: from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
pacf=plot_pacf(newData, lags=25)
acf=plot_acf(newData, lags=25)
```



```
In [ ]: train=X[1:len(X)-7]
test=X[len(X)-7:]

In [ ]: model=AutoReg(train, lags=7).fit()
```

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/ar_model.py:252: FutureWarning: The parameter names will change after 0.12 is released. Set old_names to False to use the new names now. Set old_names to True to use the old names.

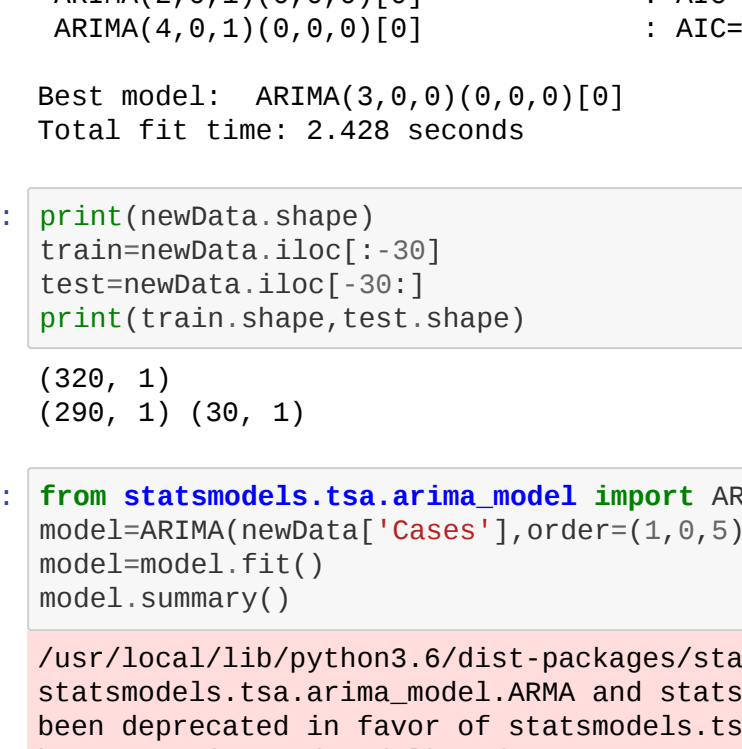
FutureWarning,

```
In [ ]: print(model.summary())

=====
AutoReg Model Results
=====
Dep. Variable: y No. Observations: 325
Model: AutoReg(10) Log Likelihood: -2293.509
Method: Conditional MLE S.D. of innovations: 351.441
Date: Tue, 22 Dec 2020 AIC: 11.800
Time: 17:21:18 BIC: 11.943
Sample: 0 HQIC: 11.857
=====
coef std err z P>|z| [0.025 0.975]
-----
Intercept 45.8556 31.752 1.444 0.149 -16.378 108.089
y.L1 -1.2289 -0.0001 -0.0001 1.2889 -0.5089 0.975
y.L2 -1.5396 0.055 27.917 0.000 1.432 1.648
y.L3 -0.2393 0.102 -2.153 0.002 -0.519 -0.121
y.L4 0.4442 0.103 4.228 0.660 -0.158 0.246
y.L5 -0.2807 0.183 -2.730 0.006 -0.482 -0.079
y.L6 -0.6174 0.104 6.167 0.000 -0.187 0.222
y.L7 -0.8138 0.104 -8.132 0.000 -0.995 -0.632
y.L8 -0.4347 0.183 2.393 0.164 -0.958 0.346
y.L9 -0.0602 0.104 -0.581 0.561 -0.263 0.143
y.L10 -0.2556 0.102 -2.796 0.005 -0.486 -0.085
y.L11 0.2557 0.056 3.800 0.000 0.197 0.325
=====
Roots
=====
Real Imaginary Modulus Frequency
-----
AR.1 1.2289 -0.0001 1.2289 0.0000
AR.2 -0.8987 -0.7843 1.1929 -0.3588
AR.3 -0.8987 +0.7843 1.1929 -0.3588
AR.4 -0.1711 -1.1477 1.1694 -0.2735
AR.5 -0.1711 +1.1477 1.1694 0.2735
AR.6 0.7713 -0.9627 1.2336 -0.1425
AR.7 0.7713 +0.9627 1.2336 0.1425
AR.8 1.0030 -0.0000 1.0030 0.0000
AR.9 1.0182 -0.0000 1.0182 -0.0000
AR.10 1.1990 -0.0000 1.1990 -0.0000
=====
```

```
In [ ]: pred=model.predict(start=len(train),end=len(X)-1,dynamic=False)
pyplot.plot(pred)
plt.plot(test,color='red')
print(rmse)
```

1368362.88909504 370851.21180284 373328.30003901 375786.90346939
370153.03864945 380552.6131267 385913.19973262



```
In [ ]: from math import sqrt
rmse=sqrt(mean_squared_error(test,pred))
rmse_list.append(rmse)

In [ ]: print(rmse)

1738.1771879127213
```

```
In [ ]: pred_future=model.predict(start=len(X)-1,end=len(X)+7,dynamic=False)
print("Future Prediction for the next week")
print(pred_future)
print("No of prediction Made:",len(pred_future))
```

Future Prediction for the next week
[381814.21642183 380186.96811566 392545.94287645 394861.84468513
397150.59709582 399446.04177094 401707.82169941]
No of prediction Made: 7

2.ARIMA MODEL

```
In [ ]: pip install pmdarima

Collecting pmdarima
  Downloading https://files.pythonhosted.org/packages/c9/d7/61af189744963882f7c8b4b4efcc2fcec68a6cda9a43ebbbdd12b967/pmdarima-1.8.0-cp36-manylinux1_x86_64.whl (1.5MB)
Installing collected packages: pmdarima
  Successfully installed pmdarima-1.8.0
Requirement already satisfied: numpy=>1.7.3 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: joblib=>0.11 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: scipy=>1.3.2 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: pandas=>0.19 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: setuptools=>50.0.0,=>38.0.0 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: scikit-learn=>0.22 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: urllib=>0.11 in /usr/local/lib/python3.6/dist-packages (from pmdarima)
Requirement already satisfied: cython=>0.29.21
Found existing installation: Cython 0.29.21
Uninstalling Cython-0.29.21:
Successfully uninstalled Cython-0.29.21
Successfully installed Cython-0.29.17 pmdarima-1.8.0
```

```
In [ ]: from statsmodels.tsa.arima import arima
stepwise_fit = arima(newData['Cases'], trace=True,
suppress_warnings=True)
```

Performing stepwise search to minimize AIC
ARIMA(2,0,2)(0,0,0)[0] intercept : AIC=4191, Time=0.56 sec
ARIMA(0,0,0)(0,0,0)[0] intercept : AIC=4867.951, Time=0.02 sec
ARIMA(1,0,0)(0,0,0)[0] intercept : AIC=4658.991, Time=0.04 sec
ARIMA(0,0,1)(0,0,0)[0] intercept : AIC=4744.897, Time=0.15 sec
ARIMA(0,0,0)(0,0,0)[0] intercept : AIC=4869.209, Time=0.01 sec
ARIMA(2,0,0)(0,0,0)[0] intercept : AIC=4637.164, Time=0.08 sec
ARIMA(0,0,0)(0,0,0)[0] intercept : AIC=4622.401, Time=0.14 sec
ARIMA(1,0,0)(0,0,0)[0] intercept : AIC=4624.488, Time=0.15 sec
ARIMA(3,0,1)(0,0,0)[0] intercept : AIC=4624.488, Time=0.21 sec
ARIMA(2,0,1)(0,0,0)[0] intercept : AIC=4626.234, Time=0.38 sec
ARIMA(4,0,1)(0,0,0)[0] intercept : AIC=4626.427, Time=0.15 sec
ARIMA(2,0,0)(0,0,0)[0] intercept : AIC=4635.022, Time=0.04 sec
ARIMA(3,0,0)(0,0,0)[0] intercept : AIC=4620.746, Time=0.07 sec
ARIMA(2,0,0)(0,0,0)[0] intercept : AIC=4635.022, Time=0.04 sec
ARIMA(4,0,0)(0,0,0)[0] intercept : AIC=4622.752, Time=0.07 sec
ARIMA(3,0,1)(0,0,0)[0] intercept : AIC=4622.751, Time=0.09 sec
ARIMA(2,0,1)(0,0,0)[0] intercept : AIC=4624.554, Time=0.13 sec
ARIMA(4,0,1)(0,0,0)[0] intercept : AIC=4624.772, Time=0.08 sec
Best model: ARIMA(3,0,0)(0,0,0)[0]
Total fit time: 2.428 seconds

```
In [ ]: print(newData.shape)
train=newData.iloc[:30]
test=newData.iloc[30:]
print(train.shape, test.shape)
```

```
In [ ]: from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(newData['Cases'],order=(1,0,5))
model=model.fit()
model.summary()
```

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/arima_model.py:472: FutureWarning: statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have been deprecated in favor of statsmodels.tsa.arima_model.ARIMA (note the . between arima and model) and statsmodels.tsa.ARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima_model.ARIMA makes use of the statespace framework and is both well tested and maintained.

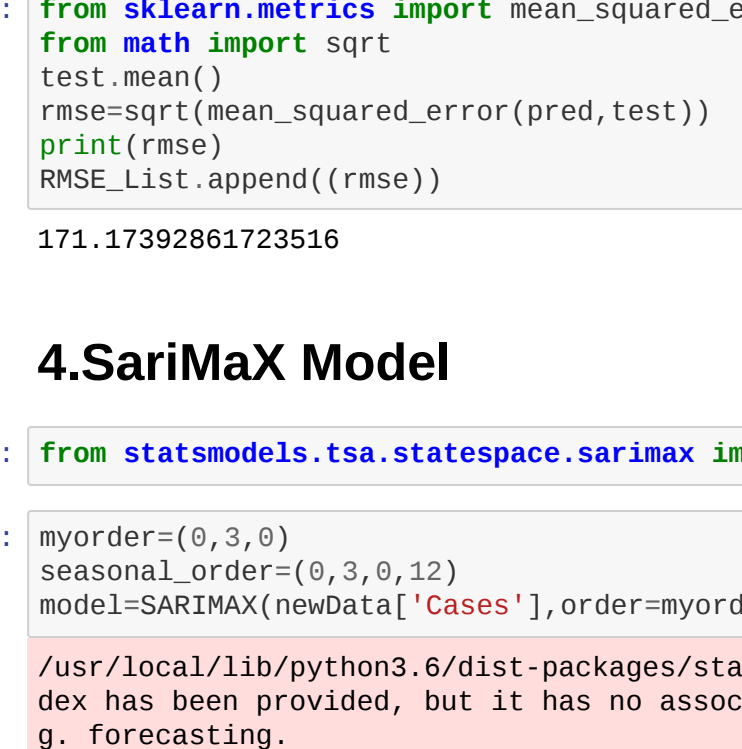
To silence this warning and continue using ARMA and ARIMA until they are removed, use:

```
import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
FutureWarning)
warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

/usr/local/lib/python3.6/dist-packages/statsmodels/base/tsa_model.py:583: ValueWarning: A date in dex has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/tsa_model.py:587: ValueWarning: A date in dex has been provided, but it is not monotonic and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in true divide
Z_mat.astype(complex), R_mat, T_mat)

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in multiply
**kwargs).imag/2, hess[1, j]
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:701: RuntimeWarning: invalid value encountered in exp
newparams = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:702: RuntimeWarning: invalid value encountered in true divide
tmp = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:548: HessianInversionWarning: Invert ing hessian failed, no use or cov_params available
available', hessianInversionWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:568: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
ConvergenceWarning)
return np.diag(-inv(hess))

```
Out [ ]: <matplotlib.legend.Legend at 0x7f5e592a0b8>
```



```
In [ ]: from sklearn.metrics import mean_squared_error
from math import sqrt
test.mean()
rmse=sqrt(mean_squared_error(pred,test))
print(rmse)
rmse_list.append(rmse)
```

167.68341645403106

3.ARMA MODEL

```
In [ ]: from statsmodels.tsa.arima_model import ARMA
model=ARMA(newData['Cases'],order=(4,1))
model=model.fit()
model.summary()
```

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/arima_model.py:472: FutureWarning: statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have been deprecated in favor of statsmodels.tsa.arima_model.ARIMA (note the . between arima and model) and statsmodels.tsa.ARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima_model.ARIMA makes use of the statespace framework and is both well tested and maintained.

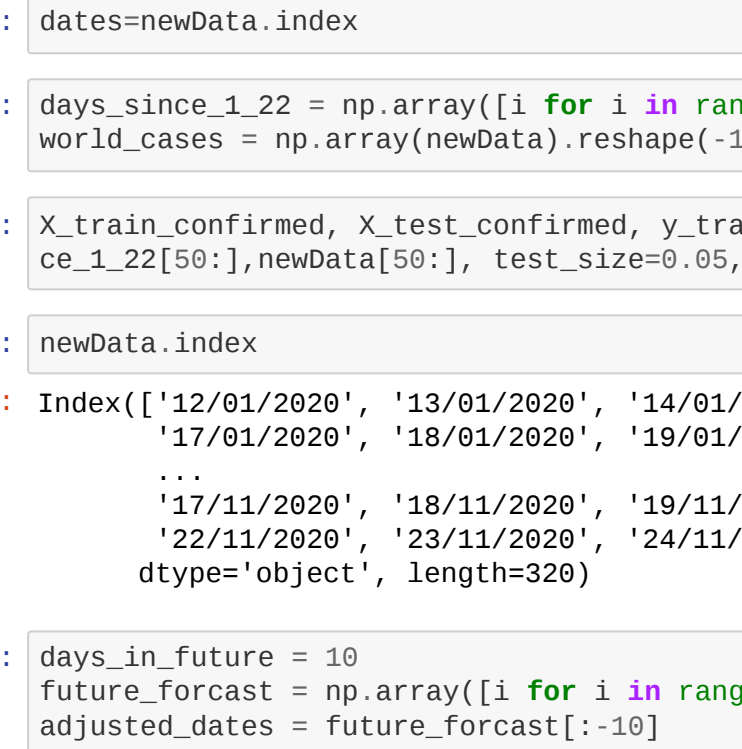
To silence this warning and continue using ARMA and ARIMA until they are removed, use:

```
import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
FutureWarning)
warnings.warn(ARMA_DEPRECATION_WARN, FutureWarning)
```

/usr/local/lib/python3.6/dist-packages/statsmodels/base/tsa_model.py:583: ValueWarning: A date in dex has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/tsa_model.py:587: ValueWarning: A date in dex has been provided, but it is not monotonic and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in true divide
Z_mat.astype(complex), R_mat, T_mat)

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in multiply
**kwargs).imag/2, hess[1, j]
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:701: RuntimeWarning: invalid value encountered in exp
newparams = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:702: RuntimeWarning: invalid value encountered in true divide
tmp = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:548: HessianInversionWarning: Invert ing hessian failed, no use or cov_params available
available', hessianInversionWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:568: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
ConvergenceWarning)
return np.diag(-inv(hess))

```
Out [ ]: <matplotlib.legend.Legend at 0x7f5e592a0b8>
```



```
In [ ]: from sklearn.metrics import mean_squared_error
from math import sqrt
test.mean()
rmse=sqrt(mean_squared_error(pred,test))
print(rmse)
rmse_list.append(rmse)
```

171.173928661723516

4.SariMAX Model

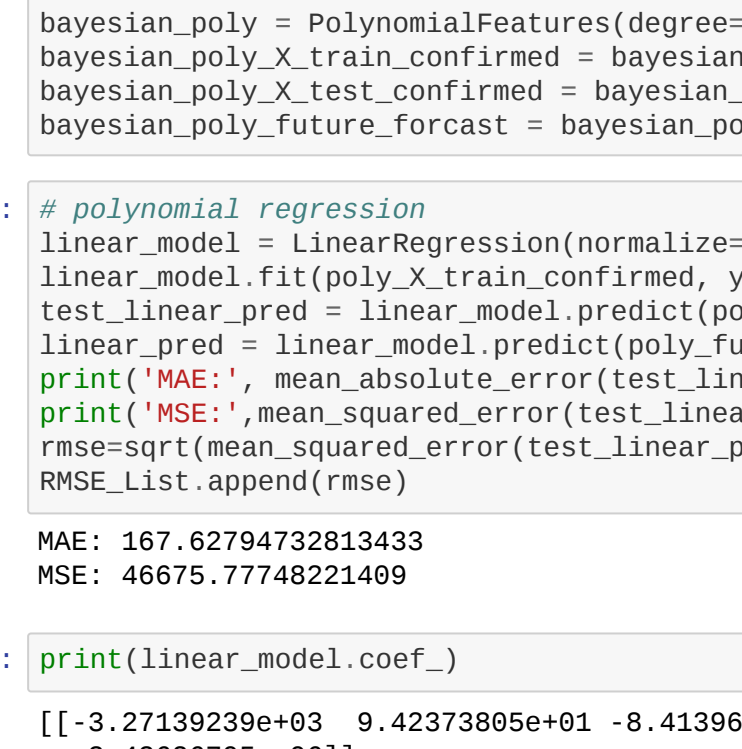
```
In [ ]: from statsmodels.tsa.statespace.sarimax import SARIMAX

In [ ]: myorder=(0,3,0)
seasonal_order=(0,3,0,12)
model=SARIMAX(newData['Cases'],order=myorder,seasonal_order=seasonal_order)
```

/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:583: ValueWarning: A date in dex has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:587: ValueWarning: A date in dex has been provided, but it is not monotonic and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:583: ValueWarning: A date in dex has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/base/tsa_model.py:587: ValueWarning: A date in dex has been provided, but it is not monotonic and so will be ignored when e.g. forecasting., ValueWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in true divide
Z_mat.astype(complex), R_mat, T_mat)

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/nudiff.py:243: RuntimeWarning: invalid value encountered in multiply
**kwargs).imag/2, hess[1, j]
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:701: RuntimeWarning: invalid value encountered in exp
newparams = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/tsa/tsatools.py:702: RuntimeWarning: invalid value encountered in true divide
tmp = ((1-np.exp(-params))/(1+np.exp(-params))).copy()
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:548: HessianInversionWarning: Invert ing hessian failed, no use or cov_params available
available', hessianInversionWarning)
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:568: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
ConvergenceWarning)
return np.diag(-inv(hess))

```
Out [ ]: <matplotlib.legend.Legend at 0x7f5e592a0b8>
```



```
In [ ]: from sklearn.metrics import mean_squared_error
from math import sqrt
test.mean()
rmse=sqrt(mean_squared_error(pred,test))
print(rmse)
rmse_list.append(rmse)
```

3098.1785217583993

5.SVM Prediction

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
import math
from sklearn.linear_model import LinearRegression, BayesianRidge
from sklearn.model_selection import RandomizedSearchCV, train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime
import operator
plt.style.use('fivethirtyeight')
matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
In [ ]: dates=newData.index

In [ ]: days_since_122 = np.array([i for i in range(len(dates))]).reshape(-1, 1)
world_cases = np.array(newData).reshape(-1, 1)
```

```
In [ ]: x_train_confirmed, x_test_confirmed, y_train_confirmed, y_test_confirmed = train_test_split(days_since_122, newData[50:], test_size=0.05, shuffle=False)
```

```
Out [ ]: newData.index

In [ ]: Index(['12/01/2020', '13/01/2020', '14/01/2020', '15/01/2020', '16/01/2020', '17/01/2020', '18/01/2020', '19/01/2020', '20/01/2020', '21/01/2020', '22/01/2020', '23/01/2020', '24/01/2020', '25/01/2020', '26/01/2020'],
dtype='object', length=320)
```

```
In [ ]: days_in_future = 30
future_forcast = np.array([i for i in range(len(dates)+days_in_future)]).reshape(-1, 1)
adjusted_dates = future_forcast[:-10]
```

```
In [ ]: # svm_confirmed = svm_search.best_estimator_
test_bayesian_pred = bayesian_confirmed.predict(bayesian_poly.X_test_confirmed)
bayesian_pred = bayesian_confirmed.predict(bayesian_poly_future_forcast)
print("MAE:", mean_absolute_error(test_bayesian_pred, y_test_confirmed))
print("RMSE:", mean_squared_error(svm_test_pred, y_test_confirmed))
rmse=sqrt(mean_squared_error(svm_test_pred, y_test_confirmed))
rmse_list.append(rmse)
print(rmse)
rmse_list.append(rmse)
```

MAE: 415.1951961903091
RMSE: 198238.899595713

```
Out [ ]: [matplotlib.lines.Line2D at 0x7f5e5c5b8d05]
```



```
In [ ]: plt.plot(svm_test_pred)
```

```
Out [ ]: [matplotlib.lines.Line2D at 0x7f5e5c5b8d05]
```