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

```
Collecting statsmodels
  Downloading https://files.pythonhosted.org/packages/be/4c/9e2435ca6645d6bafa2b51bb11f0a365b28934a2ffe9d6e339d67130926d/statsmodels-0.12.1-cp36-cp36m-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: six in /usr/local/lib/python3.6/dist-packages (from patsy>=0.5->statsmodels) (1.15.0)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.21->statsmodels) (2018.9)
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 No file chosen

Upload widget 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	5216955	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	3306	7843	40	5343702	45999.0

332 rows × 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 × 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. No of Lags :",df_test[2])
print("4. No of Observation :",df_test[3])
print("5. Critical Values :",df_test[4])
for key,val in df_test[4].items():
  print("\t",key,": ",val)
```

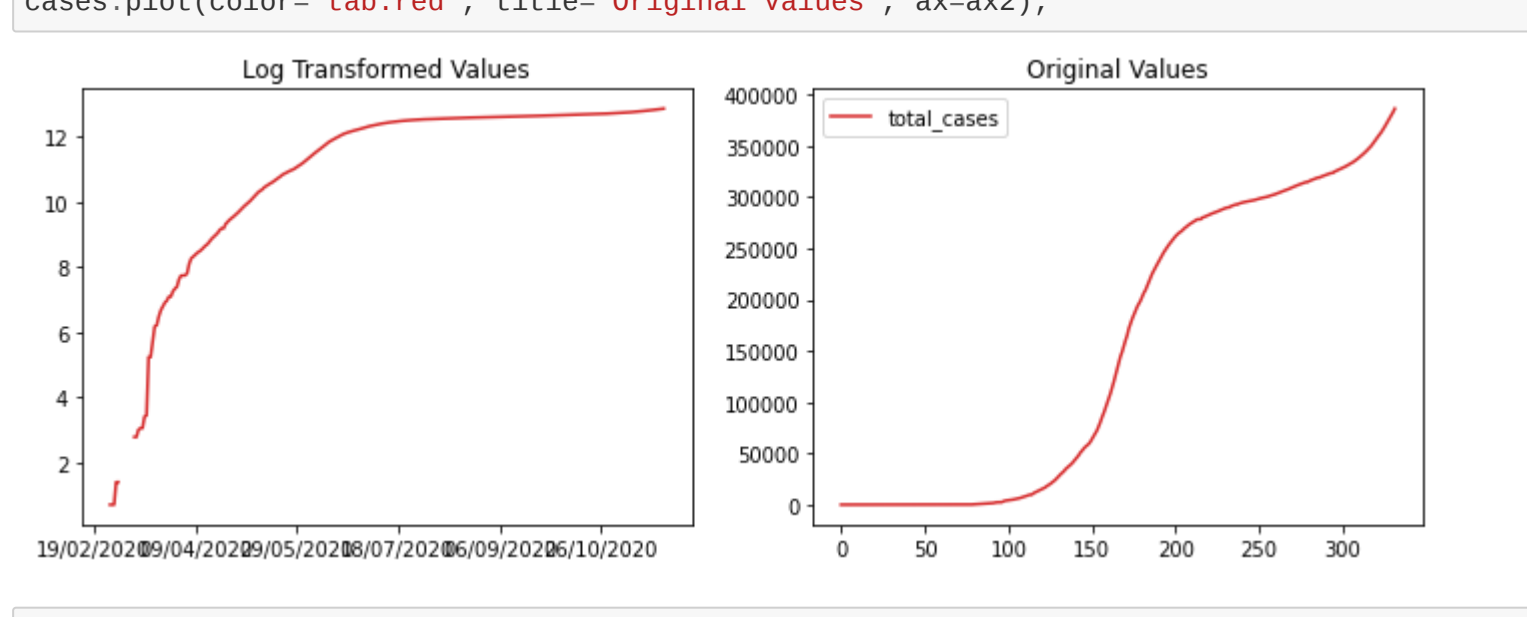
```
1. ADF : 0.10768516499141628
2. P-Value : 0.9665818958543044
3. No of Lags : 12
4. No of Observation : 319
5. Critical Values : {'1%': -3.4510167751522642, '5%': -2.87064334231426, '10%': -2.5716201744283174}
	 1% :  -3.4510167751522642
	 5% :  -2.87064334231426
	 10% :  -2.5716201744283174
```

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

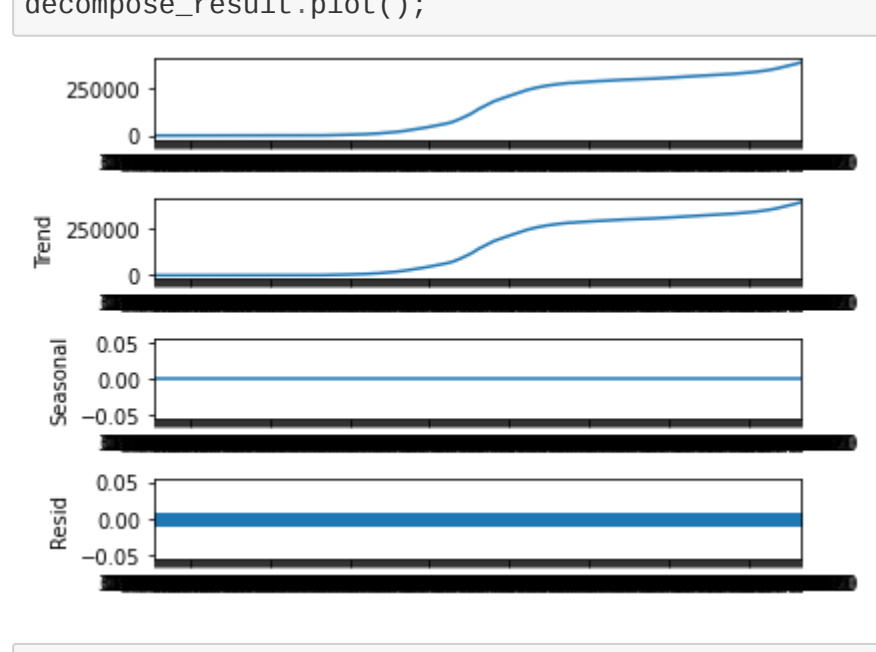
In [ ]:
```
newData=pd.DataFrame(X,Y)
```

In [ ]:
```
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);
```

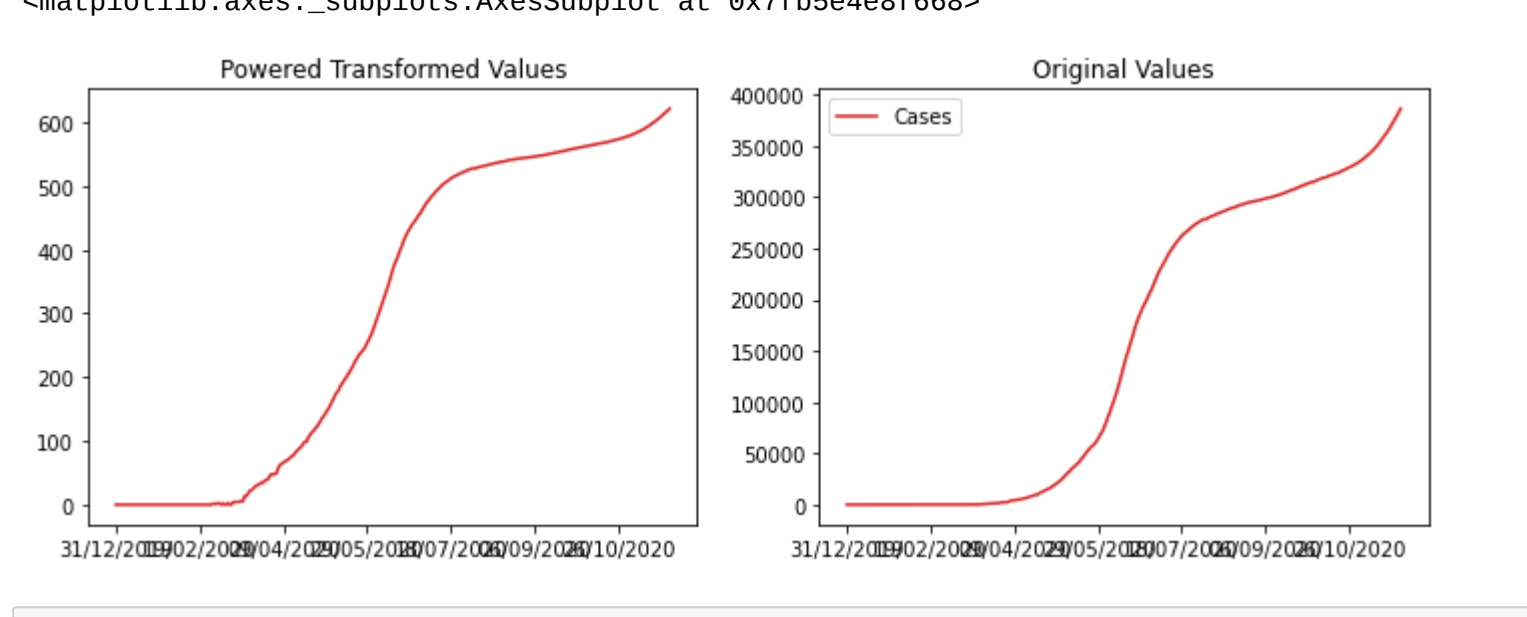


In [ ]:
```
decompose_result =seasonal_decompose(newData,period=1)
decompose_result.plot();
```

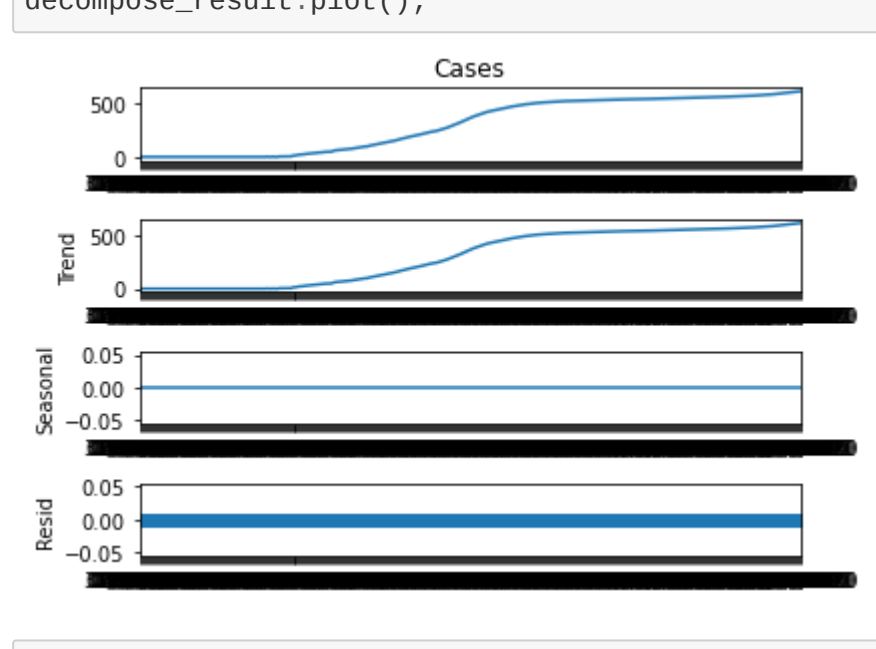


In [ ]:
```
powered_newData = newData["Cases"].apply(lambda x : x ** 0.5)

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)
```

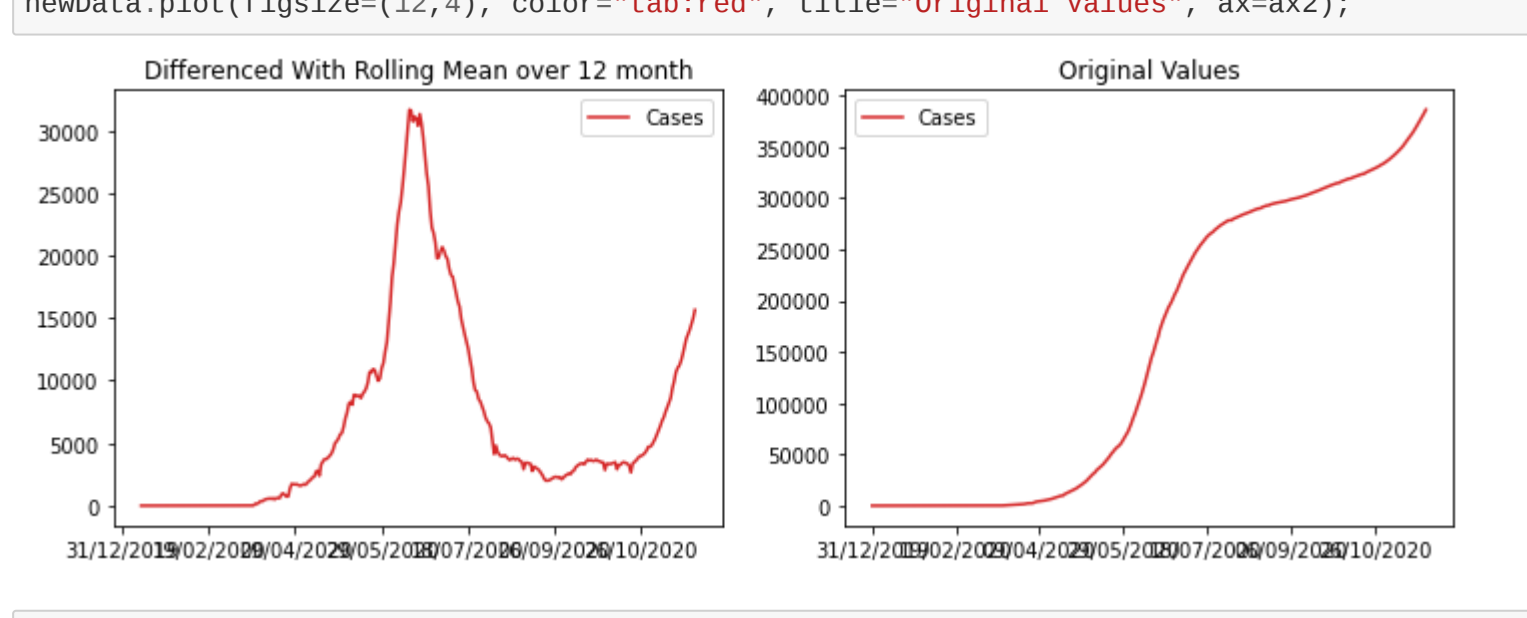
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb5e4e8f668>



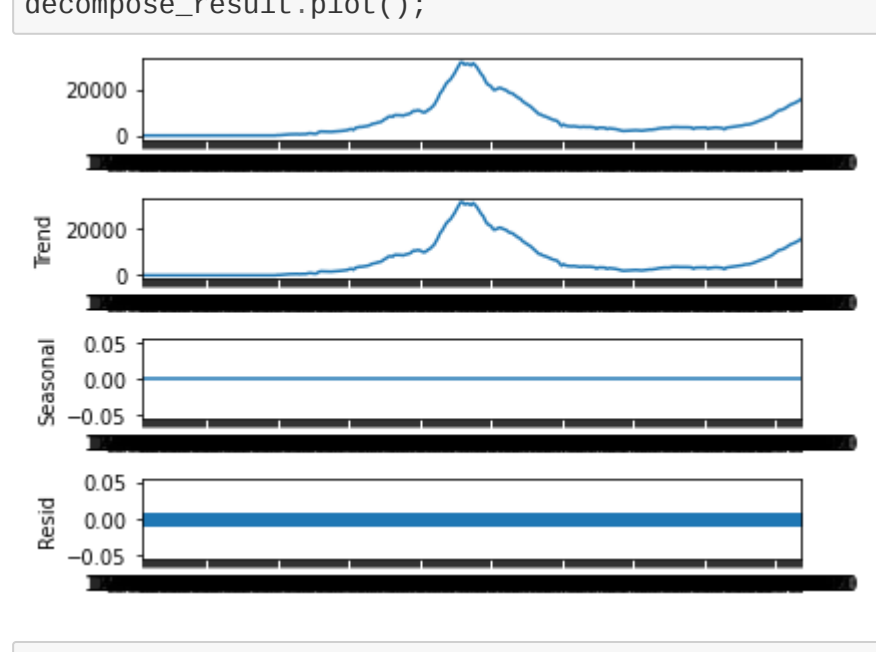
In [ ]:
```
decompose_result = seasonal_decompose(powered_newData,period=1)

decompose_result.plot();
```



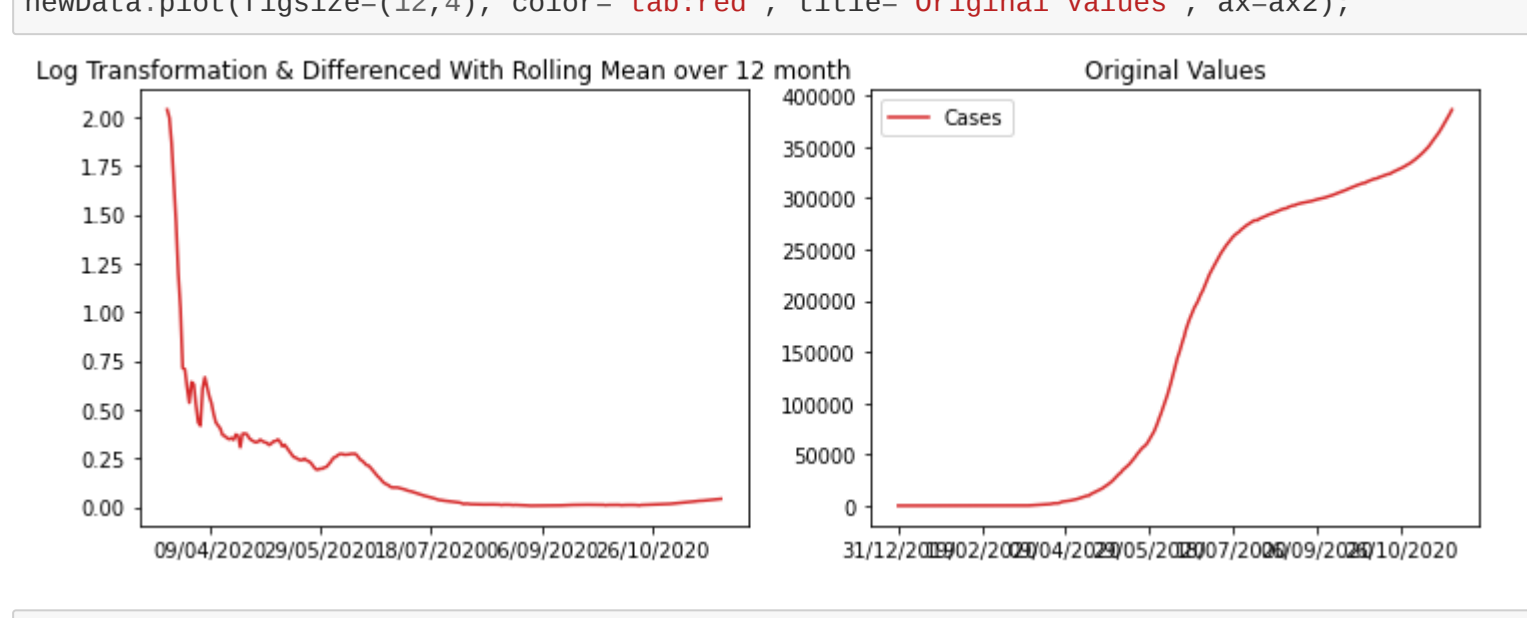
In [ ]:
```
rolling_mean = newData.rolling(window = 12).mean()
Cases_rolled_detrended = newData - rolling_mean

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);
```



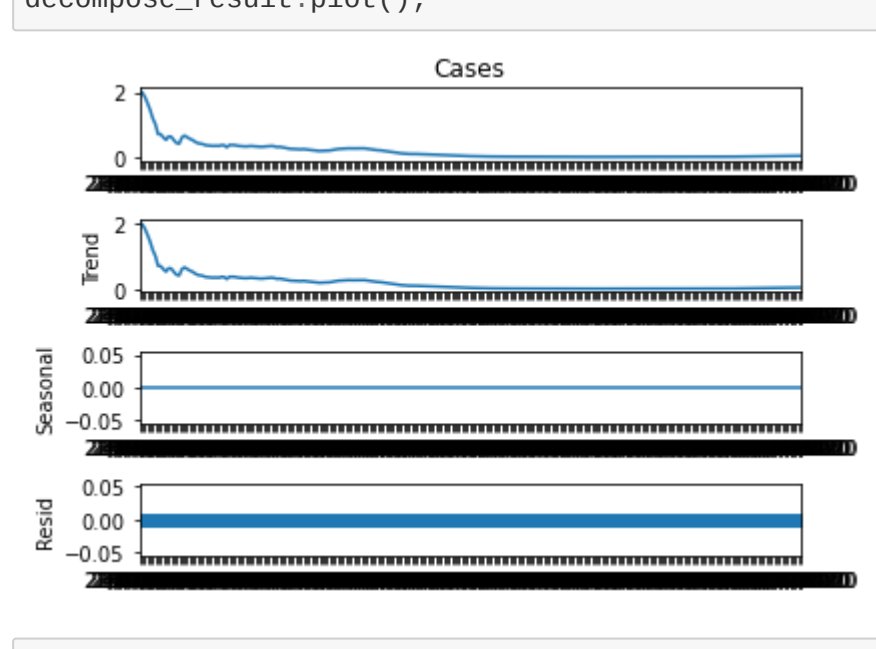
In [ ]:
```
decompose_result = seasonal_decompose(Cases_rolled_detrended.dropna(),period=1)

decompose_result.plot();
```



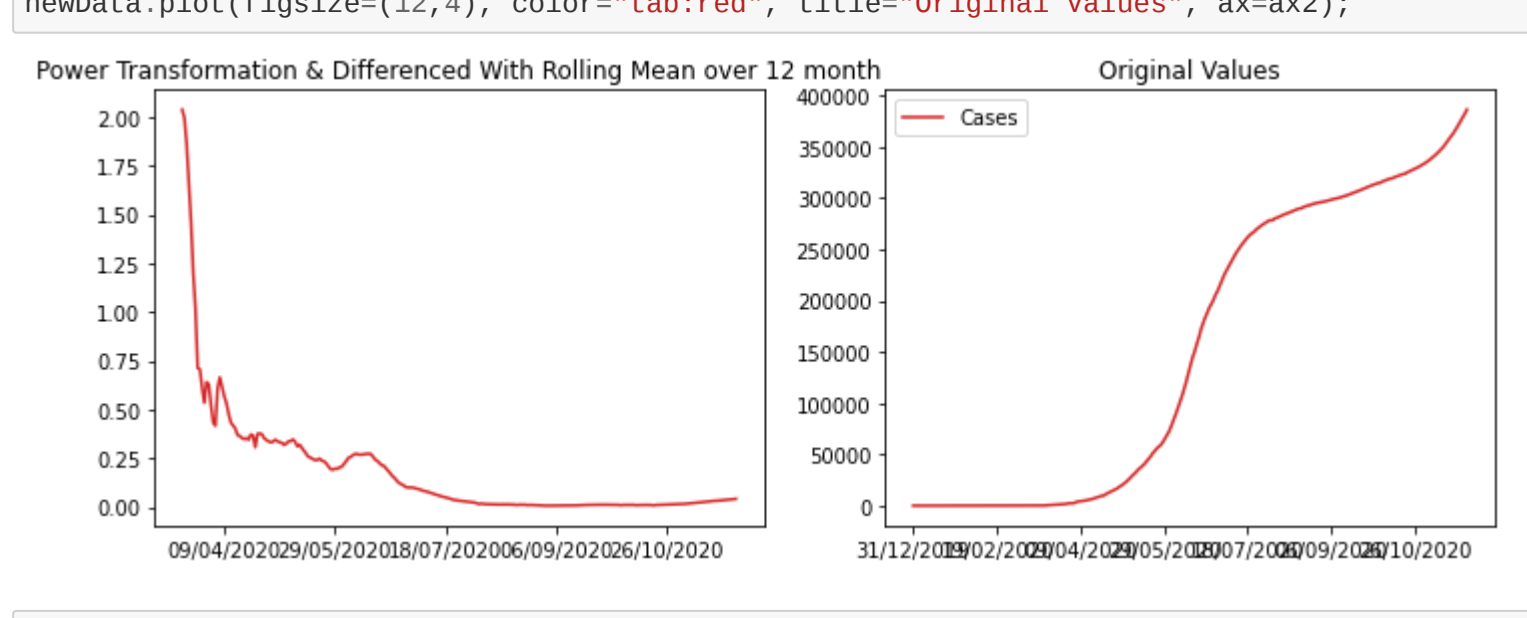
In [ ]:
```
logged_newData = pd.DataFrame(newData["Cases"].apply(lambda x : np.log(x)))

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);
```



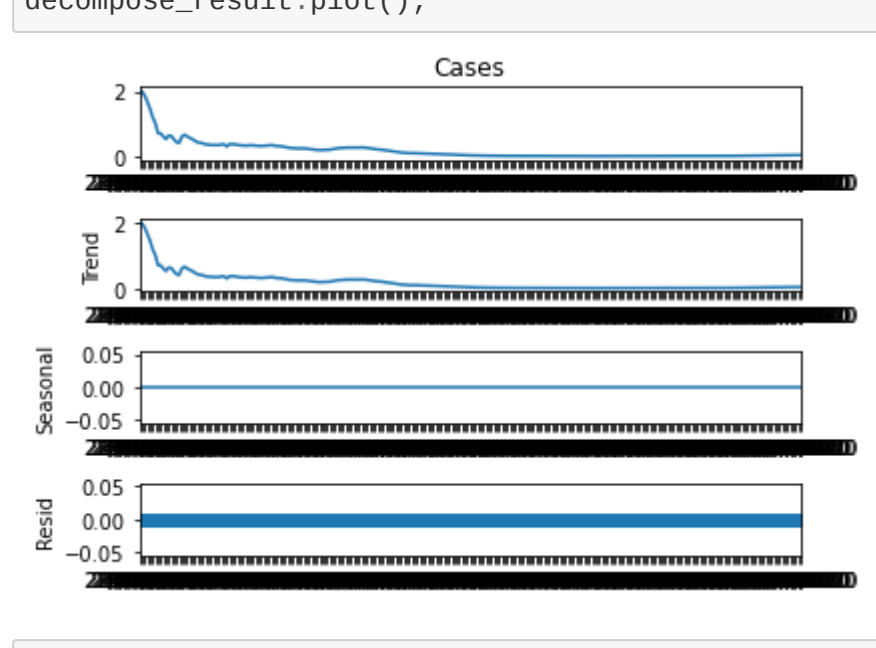
In [ ]:
```
decompose_result = seasonal_decompose(cases_log_rolled_detrended.dropna(),period=1)

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);
```



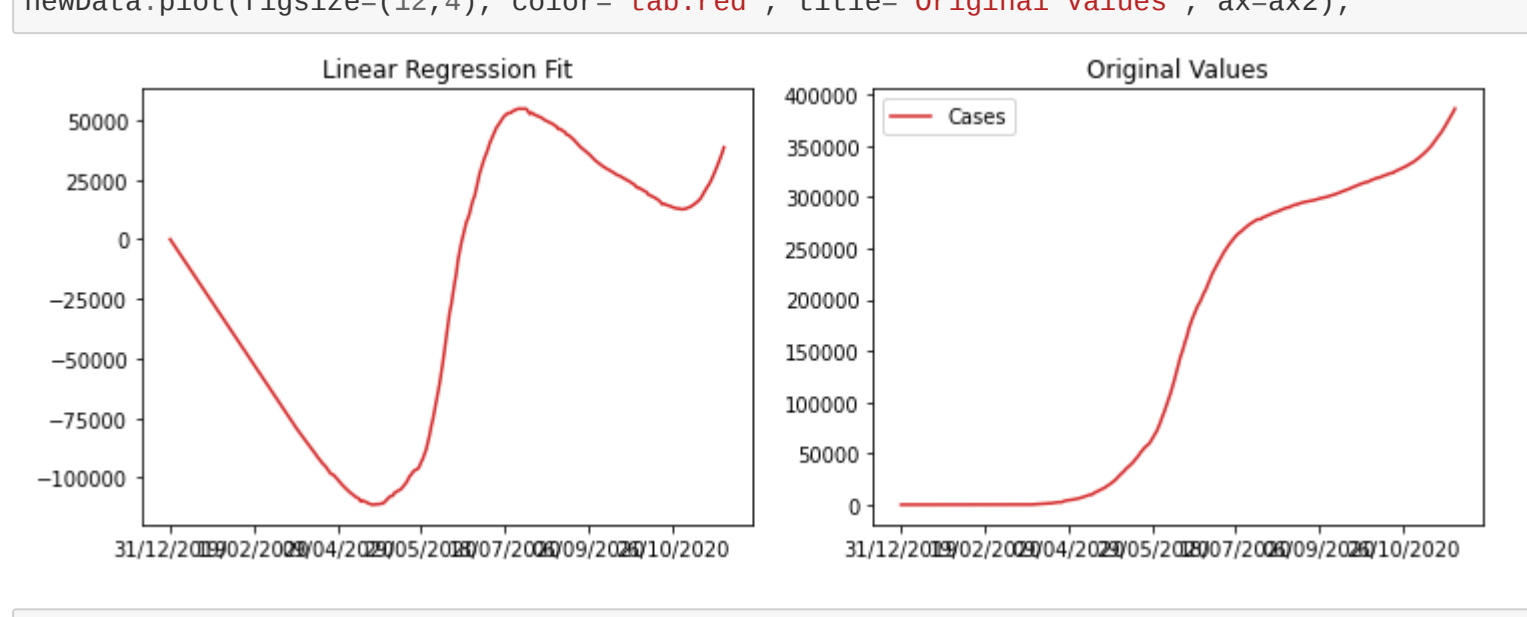
In [ ]:
```
decompose_result = seasonal_decompose(cases_pow_rolled_detrended.dropna(),period=1)

decompose_result.plot();
```



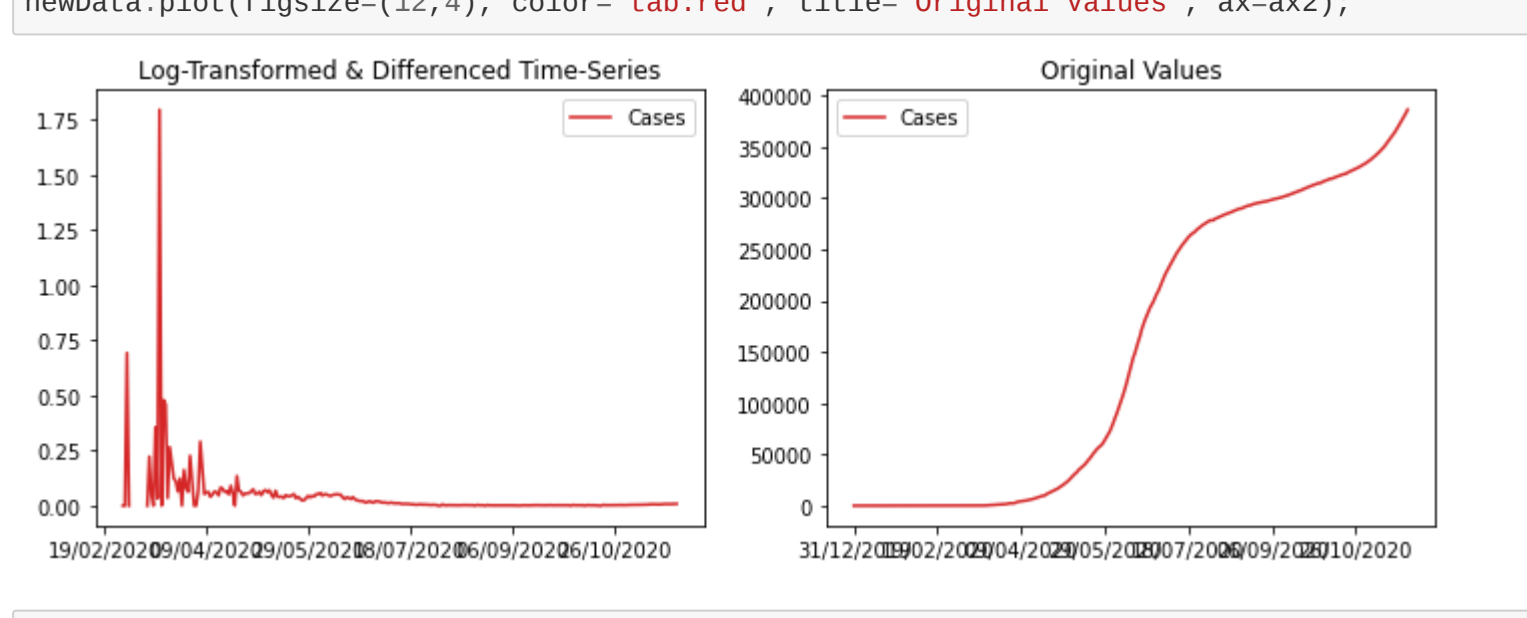
In [ ]:
```
ThisList=[]
for i in range(newData.shape[0]):
  ThisList.append(i)
ThisList
```

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]
```

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()

ax1 = plt.subplot(121)
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);
```

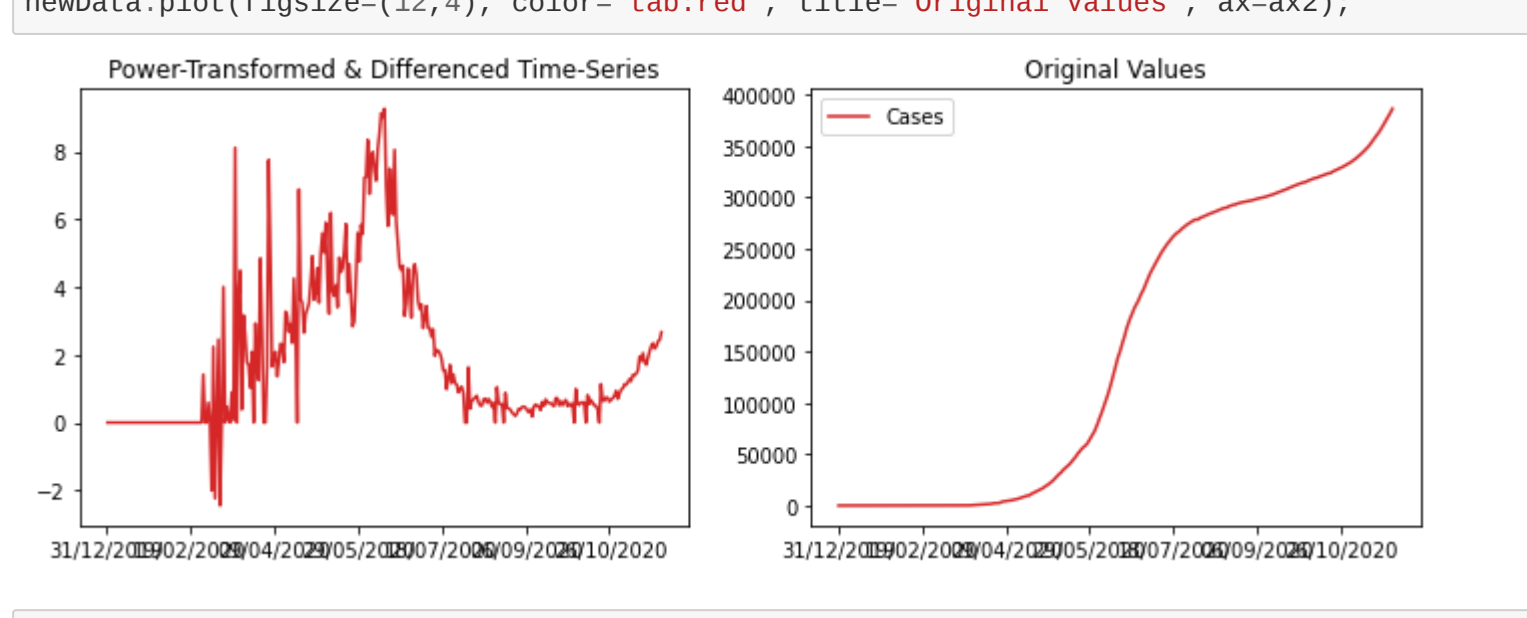


In [ ]:
```
logged_newData_diff=logged_newData_diff.replace([np.inf, -np.inf], np.nan)
logged_newData_diff=logged_newData_diff.dropna()
```

In [ ]:
```
dftest = adfuller(logged_newData_diff["Cases"].values, 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)
```

```
1. ADF :  -5.887239947861143
2. P-Value :  2.9816455550473034e-07
3. Num Of Lags :  15
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 249
5. Critical Values :
	 1% :  -3.4568881317725864
	 5% :  -2.8732185133016057
	 10% :  -2.5729936189738876
```

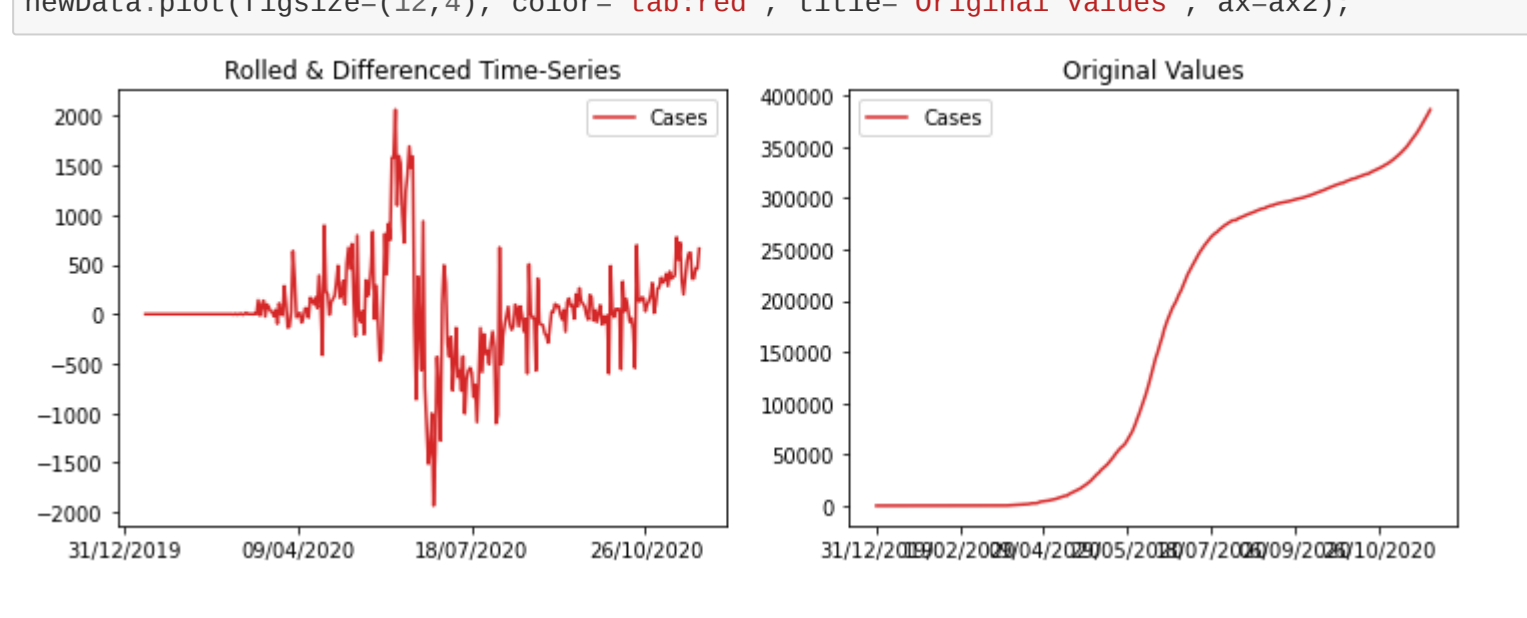
In [ ]:
```
powered_newData_diff = powered_newData - powered_newData.shift()

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);
```



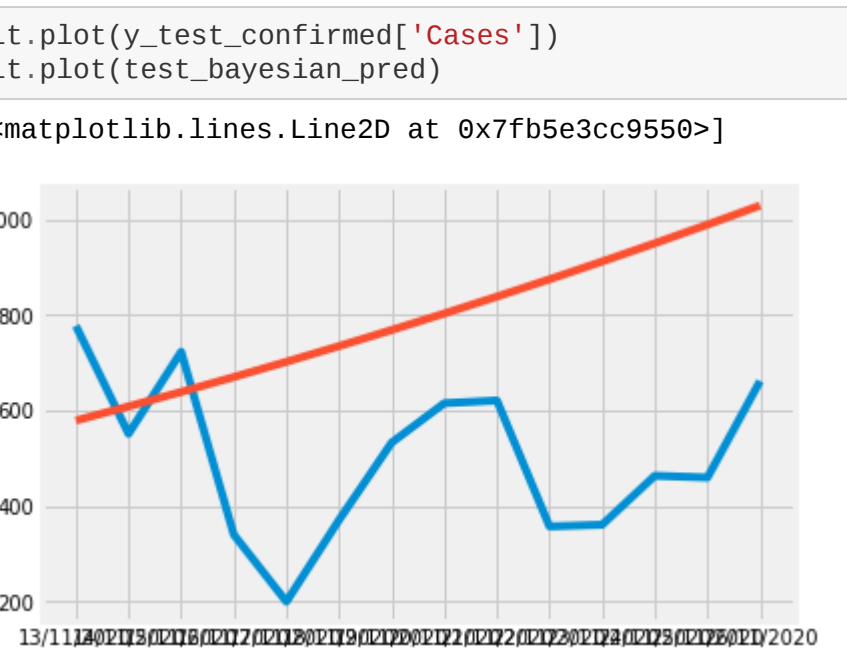
In [ ]:
```
dftest = adfuller(powered_newData_diff.dropna().values, autolag = 'AIC')
print("1. ADF : ",dftest[0])
print("2. P-Value : ", dftest[1])
print("3. Num Of Lags : ", dftest[2])
print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation :", dftest[3])
print("5. Critical Values :")
for key, val in dftest[4].items():
    print("\t",key, ": ", val)
```

```
1. ADF :  -1.694355228641399
2. P-Value :  0.43402315460836394
3. Num Of Lags :  10
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 320
5. Critical Values :
	 1% :  -3.450951662384033
	 5% :  -2.8706147570800784
	 10% :  -2.571604931640625
```

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 [ ]:
```python
plt.plot(y_test_confirmed['Cases'])
plt.plot(test_bayesian_pred)
```
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7fb5e3cc9550>]
```



Appling Other Regression

In [ ]:
```python

```

In [ ]:
```python

```

In [ ]:
```python
import sklearn.metrics as metrics
def regression_results(y_true, y_pred):
# Regression metrics
    explained_variance=metrics.explained_variance_score(y_true, y_pred)
    mean_absolute_error=metrics.mean_absolute_error(y_true, y_pred)
    mse=metrics.mean_squared_error(y_true, y_pred)
    mean_squared_log_error=metrics.mean_squared_log_error(y_true, y_pred)
    median_absolute_error=metrics.median_absolute_error(y_true, y_pred)
    r2=metrics.r2_score(y_true, y_pred)
    print('explained_variance: ', round(explained_variance,4))
    print('mean_squared_log_error: ', round(mean_squared_log_error,4))
    print('r2: ', round(r2,4))
    print('MAE: ', round(mean_absolute_error,4))
    print('MSE: ', round(mse,4))
    print('RMSE: ', round(np.sqrt(mse),4))
```

In [ ]:
```python
data_cases = newData[['Cases']]
data_cases.loc[:,'Yesterday'] = data_cases.loc[:,'Cases'].shift()
data_cases.loc[:,'Yesterday_Diff'] = data_cases.loc[:,'Yesterday'].diff()
data_cases = data_cases.dropna()
```

In [ ]:
```python
data_cases
```
Out[ ]:

	Cases	Yesterday	Yesterday_Diff
14/01/2020	0.000000	0.000000	0.000000
15/01/2020	0.000000	0.000000	0.000000
16/01/2020	0.000000	0.000000	0.000000
17/01/2020	0.000000	0.000000	0.000000
18/01/2020	0.000000	0.000000	0.000000
...	...	...	...
22/11/2020	356.916667	620.583333	5.583333
23/11/2020	360.583333	356.916667	-263.666667
24/11/2020	463.083333	360.583333	3.666667
25/11/2020	459.333333	463.083333	102.500000
26/11/2020	661.250000	459.333333	-3.750000

318 rows × 3 columns

In [ ]:
```python
X_train = data_cases[:'30/09/2020'].drop(['Cases'], axis = 1)
y_train = data_cases.loc[:'30/09/2020', 'Cases']
X_test = data_cases['01/09/2020':].drop(['Cases'], axis = 1)
y_test = data_cases.loc['01/09/2020':, 'Cases']
```

In [ ]:
```python
X_train
```
Out[ ]:

	Yesterday	Yesterday_Diff
14/01/2020	0.000000	0.000000
15/01/2020	0.000000	0.000000
16/01/2020	0.000000	0.000000
17/01/2020	0.000000	0.000000
18/01/2020	0.000000	0.000000
...	...	...
26/09/2020	170.166667	-23.666667
27/09/2020	-64.083333	-234.250000
28/09/2020	39.750000	103.833333
29/09/2020	-80.000000	-119.750000
30/09/2020	18.166667	98.166667

261 rows × 2 columns

In [ ]:
```python
y_train
```
Out[ ]:
```
14/01/2020     0.000000
15/01/2020     0.000000
16/01/2020     0.000000
17/01/2020     0.000000
18/01/2020     0.000000
                 ...
26/09/2020   -64.083333
27/09/2020    39.750000
28/09/2020   -80.000000
29/09/2020    18.166667
30/09/2020    90.583333
Name: Cases, Length: 261, dtype: float64
```

In [ ]:
```python
X_test
```
Out[ ]:

	Yesterday	Yesterday_Diff
01/09/2020	-288.416667	-68.750000
02/09/2020	-132.583333	155.833333
03/09/2020	24.166667	156.750000
04/09/2020	20.666667	-3.500000
05/09/2020	102.416667	81.750000
...	...	...
22/11/2020	620.583333	5.583333
23/11/2020	356.916667	-263.666667
24/11/2020	360.583333	3.666667
25/11/2020	463.083333	102.500000
26/11/2020	459.333333	-3.750000

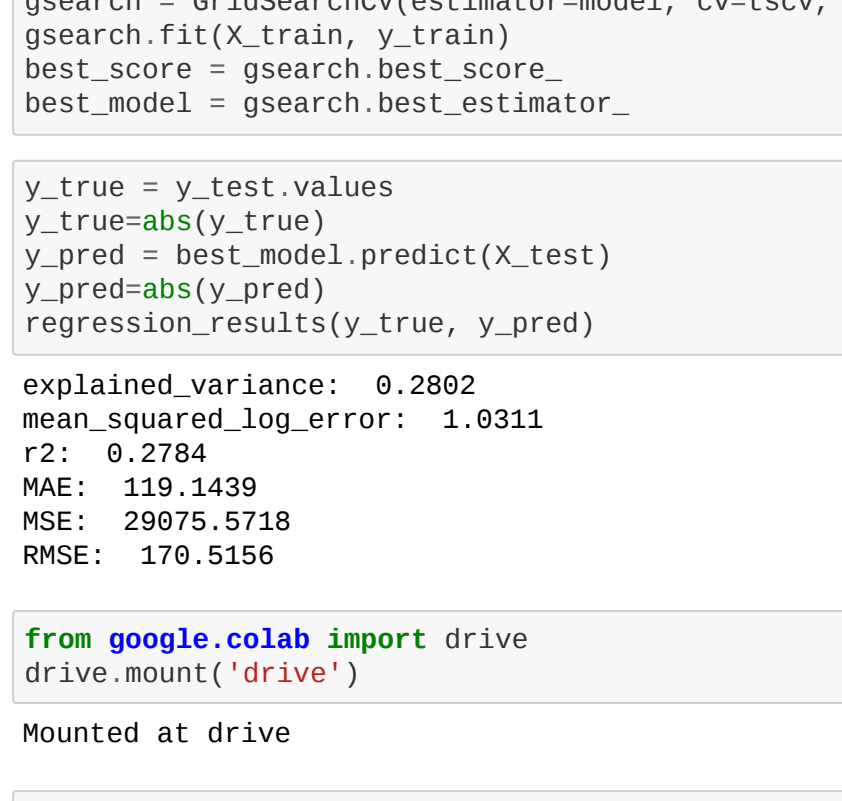
87 rows × 2 columns

In [ ]:
```python
y_test
```
Out[ ]:
```
01/09/2020   -132.583333
02/09/2020     24.166667
03/09/2020     20.666667
04/09/2020    102.416667
05/09/2020     74.666667
                 ...
22/11/2020    356.916667
23/11/2020    360.583333
24/11/2020    463.083333
25/11/2020    459.333333
26/11/2020    661.250000
Name: Cases, Length: 87, dtype: float64
```

In [ ]:
```python
X_train = abs(X_train)
y_train = abs(y_train)
X_test = abs(X_test)
y_test=abs(y_test)
```

In [ ]:
```python
from sklearn.model_selection import TimeSeriesSplit
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
```

In [ ]:
```python
models = []
models.append(('LR', LinearRegression()))
models.append(('NN', MLPRegressor(solver = 'lbfgs')))  #neural network
models.append(('KNN', KNeighborsRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 10))) # Ensemble method - collection of ma
ny decision trees
models.append(('SVR', SVR(gamma='auto'))) # kernel = linear
# Evaluate each model in turn
results = []
names = []
for name, model in models:
    # TimeSeries Cross validation
 tscv = TimeSeriesSplit(n_splits=10)

 cv_results = cross_val_score(model, X_train, y_train, cv=tscv, scoring='r2')
 results.append(cv_results)
 names.append(name)
 print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

# Compare Algorithms
plt.boxplot(results, labels=names)
plt.title('Algorithm Comparison')
plt.show()
```
```
LR: -0.340848 (0.409119)
NN: -1.122387 (1.259097)
KNN: -0.818463 (1.142538)
RF: -0.962257 (1.031090)
SVR: -1.282304 (1.321668)
```



In [ ]:
```python
from sklearn.metrics import mean_squared_error
```

In [ ]:
```python
from sklearn.metrics import make_scorer
def rmse(actual, predict):
  predict = np.array(predict)
  actual = np.array(actual)
  distance = predict - actual
  square_distance = distance ** 2
  mean_square_distance = square_distance.mean()
  score = np.sqrt(mean_square_distance)
  return score
rmse_score = make_scorer(rmse, greater_is_better = False)
```

In [ ]:
```python
from sklearn.model_selection import GridSearchCV
model = RandomForestRegressor()
param_search = {
    'n_estimators': [20, 50, 100],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [i for i in range(5,15)]
}
tscv = TimeSeriesSplit(n_splits=10)
gsearch = GridSearchCV(estimator=model, cv=tscv, param_grid=param_search, scoring = rmse_score)
gsearch.fit(X_train, y_train)
best_score = gsearch.best_score_
best_model = gsearch.best_estimator_
```

In [ ]:
```python
y_true = y_test.values
y_true=abs(y_true)
y_pred = best_model.predict(X_test)
y_pred=abs(y_pred)
regression_results(y_true, y_pred)
```
```
explained_variance:  0.2802
mean_squared_log_error:  1.0311
r2:  0.2784
MAE:  119.1439
MSE:  29075.5718
RMSE:  170.5156
```

In [ ]:
```python
from google.colab import drive
drive.mount('drive')
```
```
Mounted at drive
```

In [ ]:
```python
import pickle
```

In [ ]:
```python
pickle.dump(best_model,open('model.pkl','wb'))
```

In [ ]:
```python
!cp model.pkl "drive/My Drive/"
```

In [ ]:
```python
model=pickle.load(open('model.pkl','rb'))
```

In [ ]:
```python
newCheck={'Yesterday':3200,'Yesterday_diff':65}
newCheck=pd.DataFrame(newCheck,index=[0])
newCheck
```
Out[ ]:

	Yesterday	Yesterday_diff
0	3200	65

In [ ]:
```python
best_model.predict(newCheck)
```
Out[ ]:
```
array([817.5607619])
```

In [ ]:
```python

```