timeserieslab6-rohramehak-251524

April 6, 2024

[2]:
```
import yfinance as yahooFinance
import datetime
import pandas as pd
import statsmodels.stats.diagnostic
```

IBM Stock Data from yahoo finance

[3]:
```
startDate = datetime.datetime(2022, 10, 1)
endDate = datetime.datetime(2022, 12, 15)
data = yahooFinance.Ticker("IBM")
```

[5]:
```
ibm_data = data.history(start=startDate, end=endDate)
ibm_data
```

[5]:
```
                                 Open        High         Low       Close  \
Date
2022-10-03 00:00:00-04:00  112.103794  114.016346  111.581335  113.363281
2022-10-04 00:00:00-04:00  114.566788  117.225707  114.305555  117.085762
2022-10-05 00:00:00-04:00  116.348728  117.981398  115.900914  117.309669
2022-10-06 00:00:00-04:00  116.507327  116.899173  113.605838  114.035004
2022-10-07 00:00:00-04:00  113.353940  113.633829  110.153907  110.853622
2022-10-10 00:00:00-04:00  111.758597  111.917198  109.192973  109.855370
2022-10-11 00:00:00-04:00  109.584815  111.236148  109.099682  109.902023
2022-10-12 00:00:00-04:00  110.088602  110.844293  109.342236  109.687431
2022-10-13 00:00:00-04:00  108.315992  113.960368  107.802871  113.624504
2022-10-14 00:00:00-04:00  113.633838  114.324222  111.805241  111.991837
2022-10-17 00:00:00-04:00  113.633830  114.641415  113.288634  113.372597
2022-10-18 00:00:00-04:00  114.753378  115.630357  113.652492  114.697403
2022-10-19 00:00:00-04:00  114.156290  115.630359  113.811094  114.296234
2022-10-20 00:00:00-04:00  117.785477  120.313789  116.759229  119.698036
2022-10-21 00:00:00-04:00  119.782002  122.077076  119.035636  121.190758
2022-10-24 00:00:00-04:00  122.123705  124.185540  121.144115  123.793701
2022-10-25 00:00:00-04:00  123.149968  124.362811  122.496903  124.017609
2022-10-26 00:00:00-04:00  124.754656  126.751177  123.905664  125.958160
2022-10-27 00:00:00-04:00  126.461956  127.254959  125.435701  125.734253
2022-10-28 00:00:00-04:00  126.471283  129.550035  126.154082  129.223495
2022-10-31 00:00:00-04:00  128.803674  129.466078  127.441569  129.018250
2022-11-01 00:00:00-04:00  128.980916  129.354092  127.534834  128.934265
```

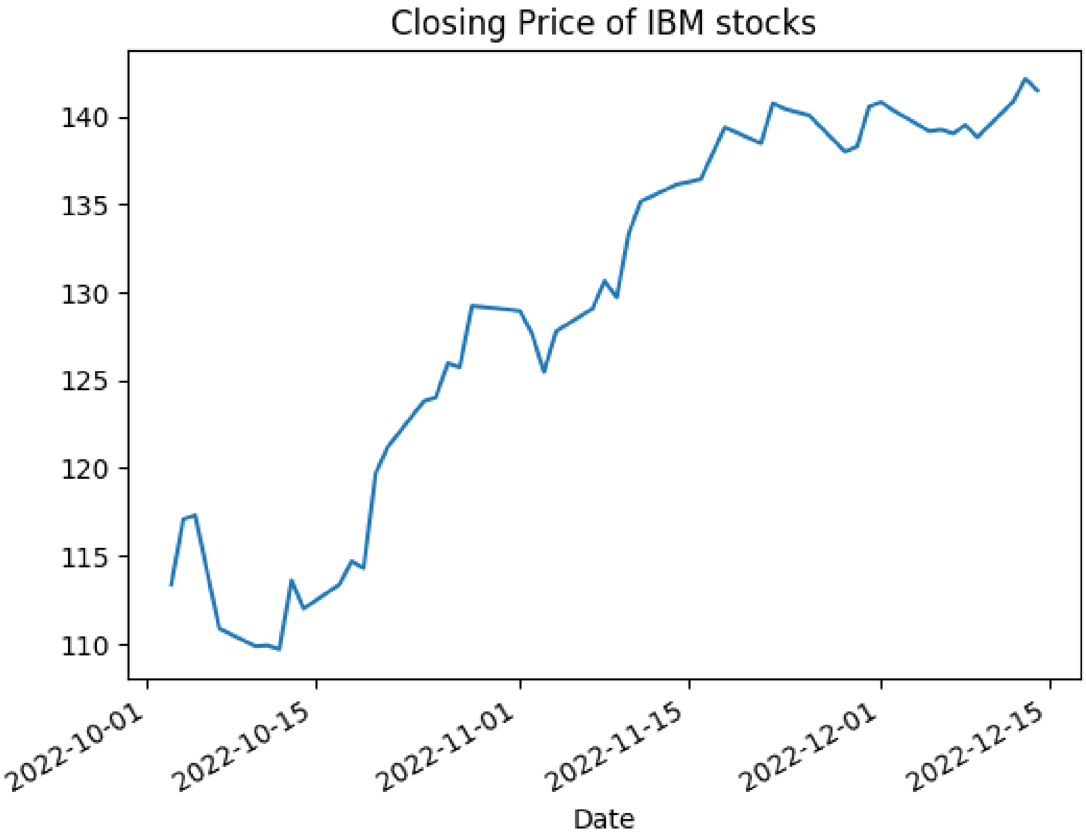
```
2022-11-02 00:00:00-04:00  128.514459  130.772207  127.628156  127.656143
2022-11-03 00:00:00-04:00  127.273619  127.329594  124.987884  125.454361
2022-11-04 00:00:00-04:00  126.555253  128.495800  125.892864  127.777435
2022-11-07 00:00:00-05:00  127.478861  129.400744  127.357572  129.064880
2022-11-08 00:00:00-05:00  129.680643  131.481238  129.419417  130.650909
2022-11-09 00:00:00-05:00  130.235499  131.132370  129.281986  129.706818
2022-11-10 00:00:00-05:00  132.416351  133.464278  130.556516  133.332108
2022-11-11 00:00:00-05:00  133.586997  136.069926  133.077201  135.163605
2022-11-14 00:00:00-05:00  134.653822  137.910888  134.228976  136.136017
2022-11-15 00:00:00-05:00  136.022733  137.986417  134.059050  136.268188
2022-11-16 00:00:00-05:00  136.069921  136.844057  135.956622  136.438110
2022-11-17 00:00:00-05:00  135.390182  138.005267  135.239126  137.920303
2022-11-18 00:00:00-05:00  138.364031  140.016167  137.778707  139.383636
2022-11-21 00:00:00-05:00  139.298683  139.657423  138.260192  138.477325
2022-11-22 00:00:00-05:00  139.345862  140.997998  138.798295  140.761978
2022-11-23 00:00:00-05:00  140.761997  142.045944  140.006732  140.431564
2022-11-25 00:00:00-05:00  139.978416  141.130192  139.817924  140.072815
2022-11-28 00:00:00-05:00  139.704607  139.950076  137.778695  138.005264
2022-11-29 00:00:00-05:00  137.750376  138.939909  137.552114  138.297943
2022-11-30 00:00:00-05:00  138.014723  141.271789  137.523799  140.573166
2022-12-01 00:00:00-05:00  141.592786  141.621107  139.100421  140.818649
2022-12-02 00:00:00-05:00  139.846238  140.818637  139.468598  140.356033
2022-12-05 00:00:00-05:00  139.666864  140.601492  138.590616  139.166504
2022-12-06 00:00:00-05:00  139.062667  139.534706  138.496215  139.251480
2022-12-07 00:00:00-05:00  139.090975  139.817919  138.109126  139.034332
2022-12-08 00:00:00-05:00  139.629093  140.809190  139.128733  139.515808
2022-12-09 00:00:00-05:00  139.157063  140.044499  138.751117  138.826645
2022-12-12 00:00:00-05:00  139.553577  140.865845  138.722784  140.865845
2022-12-13 00:00:00-05:00  141.960968  144.642160  141.564458  142.149796
2022-12-14 00:00:00-05:00  142.055380  143.414854  140.148339  141.479492

                             Volume  Dividends  Stock Splits
Date
2022-10-03 00:00:00-04:00   4261700       0.00           0.0
2022-10-04 00:00:00-04:00   4566100       0.00           0.0
2022-10-05 00:00:00-04:00   3212900       0.00           0.0
2022-10-06 00:00:00-04:00   5074600       0.00           0.0
2022-10-07 00:00:00-04:00   4499700       0.00           0.0
2022-10-10 00:00:00-04:00   5990000       0.00           0.0
2022-10-11 00:00:00-04:00   4043100       0.00           0.0
2022-10-12 00:00:00-04:00   3338800       0.00           0.0
2022-10-13 00:00:00-04:00   5837500       0.00           0.0
2022-10-14 00:00:00-04:00   3762400       0.00           0.0
2022-10-17 00:00:00-04:00   5458600       0.00           0.0
2022-10-18 00:00:00-04:00   5120300       0.00           0.0
2022-10-19 00:00:00-04:00   5906600       0.00           0.0
2022-10-20 00:00:00-04:00  13623100       0.00           0.0
```

```
2022-10-21 00:00:00-04:00   7201300       0.00          0.0
2022-10-24 00:00:00-04:00   5610900       0.00          0.0
2022-10-25 00:00:00-04:00   5957600       0.00          0.0
2022-10-26 00:00:00-04:00   5139000       0.00          0.0
2022-10-27 00:00:00-04:00   3993200       0.00          0.0
2022-10-28 00:00:00-04:00   5965500       0.00          0.0
2022-10-31 00:00:00-04:00   4915300       0.00          0.0
2022-11-01 00:00:00-04:00   3590600       0.00          0.0
2022-11-02 00:00:00-04:00   5360500       0.00          0.0
2022-11-03 00:00:00-04:00   4442400       0.00          0.0
2022-11-04 00:00:00-04:00   4178000       0.00          0.0
2022-11-07 00:00:00-05:00   4043100       0.00          0.0
2022-11-08 00:00:00-05:00   5042800       0.00          0.0
2022-11-09 00:00:00-05:00   4720000       1.65          0.0
2022-11-10 00:00:00-05:00   5389000       0.00          0.0
2022-11-11 00:00:00-05:00   5871300       0.00          0.0
2022-11-14 00:00:00-05:00   5293500       0.00          0.0
2022-11-15 00:00:00-05:00   4727100       0.00          0.0
2022-11-16 00:00:00-05:00   3445200       0.00          0.0
2022-11-17 00:00:00-05:00   3954000       0.00          0.0
2022-11-18 00:00:00-05:00   4661700       0.00          0.0
2022-11-21 00:00:00-05:00   3476200       0.00          0.0
2022-11-22 00:00:00-05:00   7062100       0.00          0.0
2022-11-23 00:00:00-05:00   3658200       0.00          0.0
2022-11-25 00:00:00-05:00   2075200       0.00          0.0
2022-11-28 00:00:00-05:00   3538100       0.00          0.0
2022-11-29 00:00:00-05:00   2754700       0.00          0.0
2022-11-30 00:00:00-05:00   6377600       0.00          0.0
2022-12-01 00:00:00-05:00   4495900       0.00          0.0
2022-12-02 00:00:00-05:00   2900000       0.00          0.0
2022-12-05 00:00:00-05:00   2784600       0.00          0.0
2022-12-06 00:00:00-05:00   2847600       0.00          0.0
2022-12-07 00:00:00-05:00   3971300       0.00          0.0
2022-12-08 00:00:00-05:00   2665700       0.00          0.0
2022-12-09 00:00:00-05:00   3047600       0.00          0.0
2022-12-12 00:00:00-05:00   4032800       0.00          0.0
2022-12-13 00:00:00-05:00   8811500       0.00          0.0
2022-12-14 00:00:00-05:00   4205900       0.00          0.0
```

[6]:
```
series_closing = ibm_data["Close"]
```

[8]:
```
series_closing.plot(title="Closing Price of IBM stocks")
```

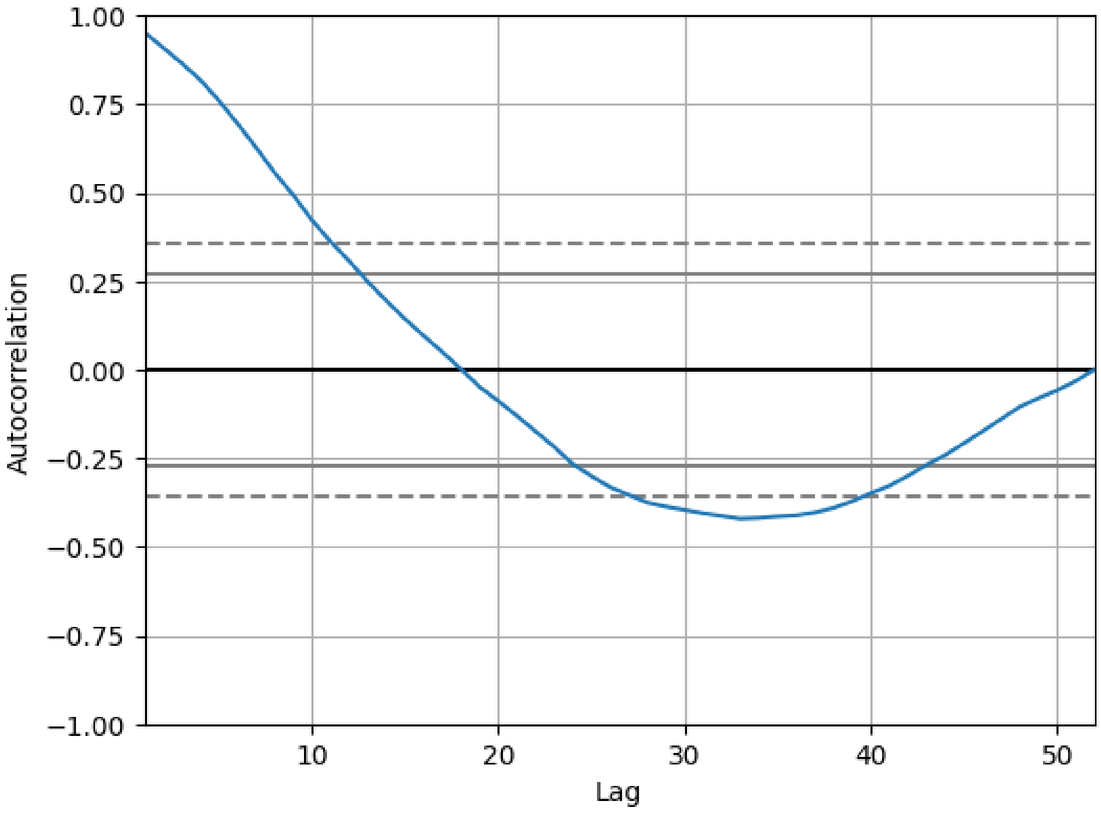
[8]:
```
<Axes: title={'center': 'Closing Price of IBM stocks'}, xlabel='Date'>
```



Analysis of Co-relation in the IBM Closing price time series

```
[9]: pd.plotting.autocorrelation_plot(series_closing)
```

```
[9]: <Axes: xlabel='Lag', ylabel='Autocorrelation'>
```



LjungBox Test on the Closing Series

```
[10]: dt= statsmodels.stats.diagnostic.acorr_ljungbox(series_closing, lags=None,␣
        ↪boxpierce=False, model_df=0, period=None, return_df=True, auto_lag=False)
      dt
```

```
[10]:        lb_stat     lb_pvalue
      1    49.951899  1.575615e-12
      2    96.426697  1.151354e-21
      3   139.373919  5.157653e-30
      4   178.569772  1.512453e-37
      5   213.079518  4.509226e-44
      6   242.559458  1.594377e-49
      7   267.111828  6.282456e-54
      8   286.822185  2.618588e-57
      9   302.677881  7.056830e-60
      10  314.584594  1.278075e-61
```

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

Adfuller Test on the Closing Series

[12]:
```
ad_fullter_test_data = adfuller(series_closing)
ad_fullter_test_data
```

[12]:
```
(-0.8810826058142052,
 0.7941383383157626,
 0,
 51,
 {'1%': -3.5656240522121956,
  '5%': -2.920142229157715,
  '10%': -2.598014675124952},
 147.36694968104985)
```

p_value > .05 so it is non stationary

Differencing the time series

[13]:
```
ibm_data['diff'] = ibm_data["Close"].diff()
```

Performing Adfuller Test on the difference

[14]:
```
ad_fullter_test_data = adfuller(ibm_data['diff'].dropna())
ad_fullter_test_data
```

[14]:
```
(-3.4889881968435374,
 0.008274445362623582,
 9,
 41,
 {'1%': -3.60098336718852,
  '5%': -2.9351348158036012,
  '10%': -2.6059629803688282},
 147.22173012040238)
```

p_val is less that 0.05. series is stationary

ELECTRICITY TIME SERIES

[15]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[16]:
```
data = pd.read_csv('/content/drive/MyDrive/TSA_BDA_2024/Lab4/
 ↪Retail_sales_of_electricity_United_States_monthly.csv', skiprows=4)
```

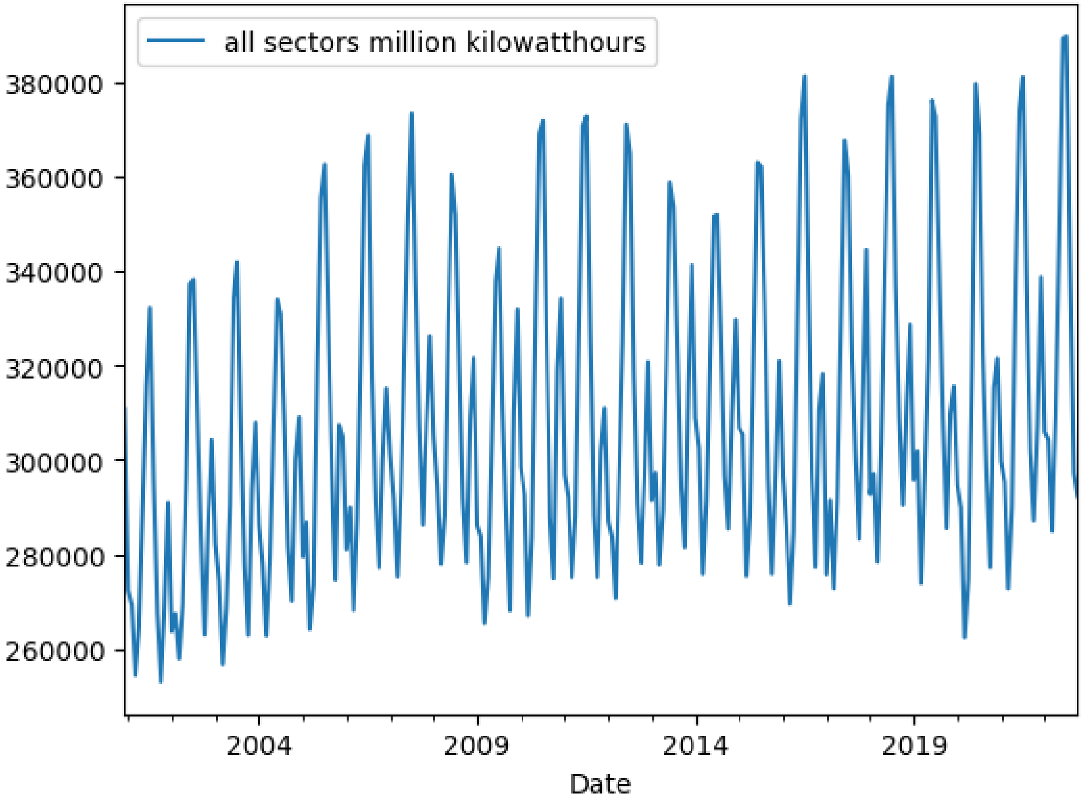
[17]:
```
data.dropna(inplace=True)
data['DatePart'] = pd.to_datetime(data['Month'], format='%b %Y',␣
 ↪errors='coerce')
data.rename(columns = {"DatePart" : "Date"}, inplace=True)
data.set_index("Date", inplace=True)
data.drop(columns=["Month"], inplace=True)
```

[18]:

```
data.plot()
```

[18]:

```
<Axes: xlabel='Date'>
```



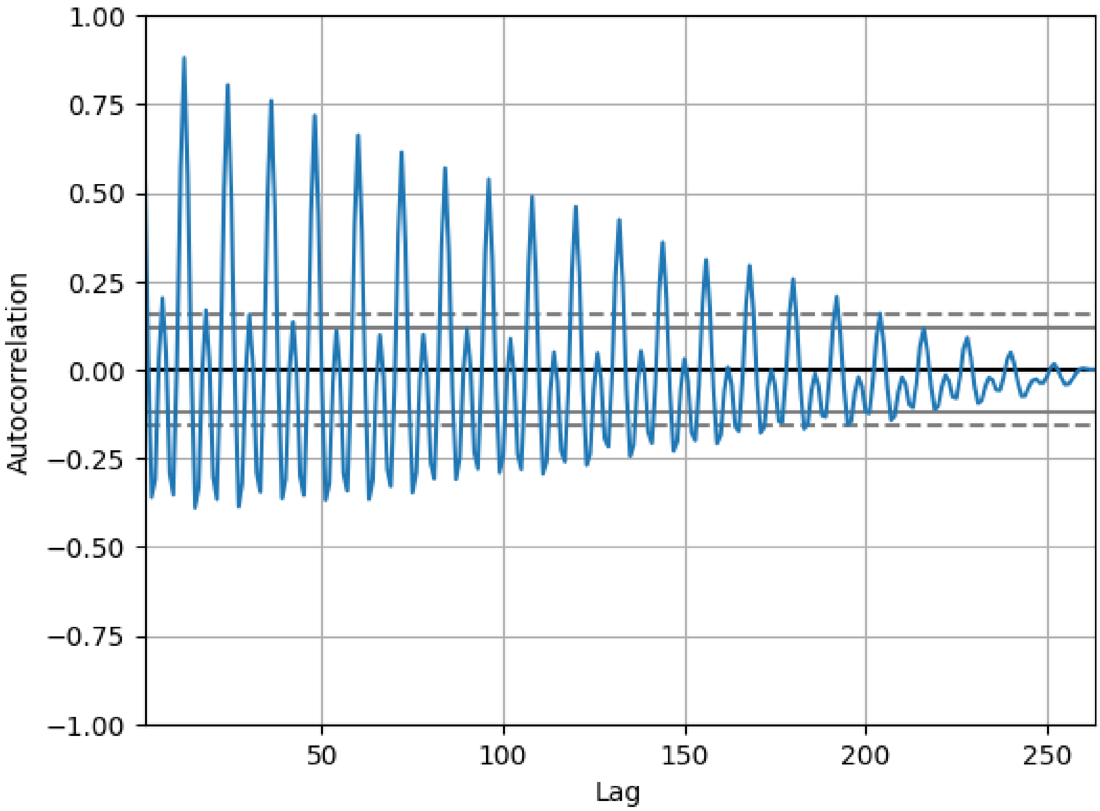
Autocorelation plot of the “All sectors” time series

[19]:

```
pd.plotting.autocorrelation_plot(data['all sectors million kilowatthours'])
```

[19]:

```
<Axes: xlabel='Lag', ylabel='Autocorrelation'>
```



Differencing the time series with a lag of 12

```
[20]: data['diff'] = data["all sectors million kilowatthours"].diff(12)
```

Adfuller test on the differences time series

```
[21]: ad_fullter_test_data = adfuller(data['diff'].dropna())
      ad_fullter_test_data
```

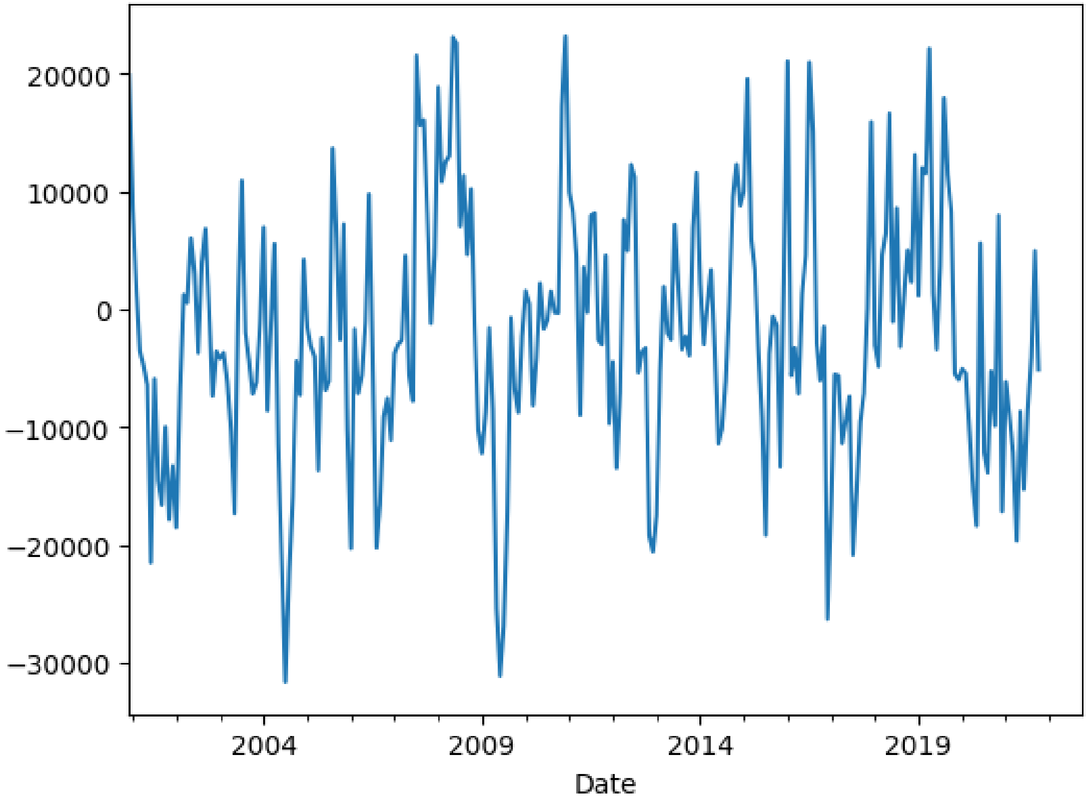
```
[21]: (-4.702498710240669,
       8.324793841732234e-05,
       12,
       238,
       {'1%': -3.458128284586202,
        '5%': -2.873761835239286,
        '10%': -2.5732834559706235},
       4881.89911722033)
```

p_val < 0.05 . series is stationary

Plot of the differenced time series :

```
[22]: data['diff'].plot()
```

```
[22]: <Axes: xlabel='Date'>
```



LjungBox test on the differenced Time Series - all sectors data

```
[23]: dt= statsmodels.stats.diagnostic.acorr_ljungbox(data['diff'].dropna(),␣
       ↪lags=None, boxpierce=False, model_df=0,
                                                    period=None, return_df=True,␣
       ↪auto_lag=False)
      dt
```

```
[23]:       lb_stat     lb_pvalue
      1    90.091768  2.273658e-21
      2   123.728696  1.357235e-27
      3   141.016254  2.282085e-30
      4   148.552069  4.158961e-31
      5   152.256917  4.416147e-31
      6   153.609170  1.334273e-30
      7   153.969932  5.948939e-30
      8   154.033811  2.822368e-29
      9   154.510602  1.024277e-28
      10  155.041854  3.413888e-28
```