[PDSF2016] ARIMA-SARIMAX file 02-04-2020

February 4, 2022

[129]:
```
import numpy as np
import pandas as pd
import xlsxwriter as ExcelWriter
import statsmodels.tsa.api as smt
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
import seaborn as sns
import datetime as dt
from pandas.plotting import register_matplotlib_converters # Handle date-time
register_matplotlib_converters()                           #conversions between
                                                           #pandas and␣
 ↪matplotlib
%matplotlib inline
import linearmodels as plm
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.tsa.api as smt
import statsmodels.stats.outliers_influence as smo
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[3]:
```
Panel = pd.read_csv("/Users"
                    "/andrew7"
                    "/Desktop/"
                    "[TONE_DA]/"
                    "Estimation 22/"
                    "Consolidated Data 01 2022"
                    "/Tone_Est_Paneled II.csv",
                   parse_dates = [0])
Panel
```

[3]:

	Date	Month	Day	DateWk	Year	DSP	Streams	Listeners	\
0	2020-03-04	3	4	Wednesday	2020	1	7	7	
1	2020-03-05	3	5	Thursday	2020	1	58	7	
2	2020-03-06	3	6	Friday	2020	1	59	8	
3	2020-03-07	3	7	Saturday	2020	1	57	4	
4	2020-03-08	3	8	Sunday	2020	1	19	5	

```
…          …     …  …         …    …  …       …         …
1998 2021-12-27     12   27     Monday  2021    3     5430       4137
1999 2021-12-28     12   28    Tuesday  2021    3     5917       4513
2000 2021-12-29     12   29  Wednesday  2021    3     6479       4782
2001 2021-12-30     12   30   Thursday  2021    3     5854       4571
2002 2021-12-31     12   31     Friday  2021    3     5476       4283

      CMEngage  CMCrossP  TWFollowers  TWRetweets     IGReach
0       379454    465864          226           0  186.250000
1       404875    487377          226           0  168.500000
2       407748    494572          227           0  164.666667
3       413262    498931          227           0  179.000000
4       411552    497810          227           0  180.666667
…          …         …            …           …           …
1998    131006    160044          381           2  291.500000
1999    128797    157601          381           2    0.000000
2000    117195    143391          381           2    0.000000
2001    116277    142352          381           2    0.000000
2002    114926    140372          382           2  559.000000

[2003 rows x 13 columns]
```

[7]:
```
Panel['lnDiffStreams']   = np.log(Panel['Streams'].diff()).bfill().ffill()
Panel['lnDiffStreams']
```

[7]:
```
0       3.931826
1       3.931826
2       0.000000
3       2.944439
4       2.944439
          …
1998    6.879356
1999    6.188264
2000    6.331502
2001    6.331502
2002    6.331502
Name: lnDiffStreams, Length: 2003, dtype: float64
```

[8]:
```
DayWkdict = {'Sunday': 1,
             'Monday': 2,
             'Tuesday': 3,
             'Wednesday': 4,
             'Thursday': 5,
             'Friday': 6,
             'Saturday': 7}
Panel['DayWk'] = Panel['DateWk'].map(DayWkdict)
DSPdict = {1:'Apple Music',
```

```
            2:'SoundCloud',
            3:'Spotify'}
Panel['DSPstr'] = Panel['DSP'].map(DSPdict)
Panel
```

[8]:

```
            Date  Month  Day     DateWk  Year  DSP  Streams  Listeners  \
0     2020-03-04      3    4  Wednesday  2020    1        7          7
1     2020-03-05      3    5   Thursday  2020    1       58          7
2     2020-03-06      3    6     Friday  2020    1       59          8
3     2020-03-07      3    7   Saturday  2020    1       57          4
4     2020-03-08      3    8     Sunday  2020    1       19          5
…            …    …  …   …          …   …  …      …          …
1998  2021-12-27     12   27     Monday  2021    3     5430       4137
1999  2021-12-28     12   28    Tuesday  2021    3     5917       4513
2000  2021-12-29     12   29  Wednesday  2021    3     6479       4782
2001  2021-12-30     12   30   Thursday  2021    3     5854       4571
2002  2021-12-31     12   31     Friday  2021    3     5476       4283

      CMEngage  CMCrossP  …  TWRetweets     IGReach  lnDiffIGReach  \
0       379454    465864  …           0  186.250000       2.662588
1       404875    487377  …           0  168.500000       2.662588
2       407748    494572  …           0  164.666667       2.662588
3       413262    498931  …           0  179.000000       2.662588
4       411552    497810  …           0  180.666667       0.510826
…          …         …  …  …          …           …              …
1998    131006    160044  …           2  291.500000       5.675040
1999    128797    157601  …           2    0.000000           -inf
2000    117195    143391  …           2    0.000000           -inf
2001    116277    142352  …           2    0.000000           -inf
2002    114926    140372  …           2  559.000000       6.326149

      lnDiffEGR  lnDiffStreams  lnDiffListeners  lnDiffTWR  lnDiffTWF  DayWk  \
0     10.143331       3.931826             -inf       -inf       -inf      4
1     10.143331       3.931826             -inf       -inf       -inf      5
2      7.963112       0.000000         0.000000       -inf        0.0      6
3      8.615046       2.944439         0.000000       -inf       -inf      7
4      8.358197       2.944439         0.000000       -inf       -inf      1
…           …              …                …          …          …    …
1998   9.358588       6.879356         6.728629   0.693147       -inf      2
1999   9.358588       6.188264         5.929589       -inf       -inf      3
2000   9.358588       6.331502         5.594711       -inf       -inf      4
2001   9.358588       6.331502         5.594711       -inf       -inf      5
2002   9.358588       6.331502         5.594711       -inf        0.0      6

          DSPstr
0     Apple Music
1     Apple Music
```

```
2     Apple Music
3     Apple Music
4     Apple Music
…           …
1998      Spotify
1999      Spotify
2000      Spotify
2001      Spotify
2002      Spotify

[2003 rows x 21 columns]
```

[9]:
```
Data_dict = {'Date': Panel['Date'],
             'Platform': Panel['DSPstr'],
             'lnDiffStreams': Panel['lnDiffStreams']}
Data = pd.DataFrame(Data_dict)
Data['lnDiffStreams'].astype(float)
Data.replace([-np.inf, np.inf], np.nan, inplace=True)
Data['lnDiffStreams'] = Data['lnDiffStreams'].bfill().ffill()
```

[10]:
```
Data
```

[10]:
```
           Date     Platform  lnDiffStreams
0    2020-03-04  Apple Music       3.931826
1    2020-03-05  Apple Music       3.931826
2    2020-03-06  Apple Music       0.000000
3    2020-03-07  Apple Music       2.944439
4    2020-03-08  Apple Music       2.944439
…         …           …              …
1998 2021-12-27      Spotify       6.879356
1999 2021-12-28      Spotify       6.188264
2000 2021-12-29      Spotify       6.331502
2001 2021-12-30      Spotify       6.331502
2002 2021-12-31      Spotify       6.331502

[2003 rows x 3 columns]
```

[11]:
```
Data['Date'] = pd.to_datetime(Data['Date'], format='%Y-%m-d')
Data.dtypes
```

[11]:
```
Date             datetime64[ns]
Platform                 object
lnDiffStreams           float64
dtype: object
```

[12]:
```
type(Data.index)
```

[12]:
```
pandas.core.indexes.range.RangeIndex
```

[13]:
```
Data.index = pd.DatetimeIndex(Data['Date'])
Data = Data.drop(['Date'], axis=1)
Data
```

[13]:

Date	Platform	lnDiffStreams
2020-03-04	Apple Music	3.931826
2020-03-05	Apple Music	3.931826
2020-03-06	Apple Music	0.000000
2020-03-07	Apple Music	2.944439
2020-03-08	Apple Music	2.944439
…	…	…
2021-12-27	Spotify	6.879356
2021-12-28	Spotify	6.188264
2021-12-29	Spotify	6.331502
2021-12-30	Spotify	6.331502
2021-12-31	Spotify	6.331502

[2003 rows x 2 columns]

[14]:
```
AM = Data.query("Platform == 'Apple Music'")
SC = Data.query("Platform == 'SoundCloud'")
SP = Data.query("Platform == 'Spotify'")
#AM.head(),SC.head(),SP.head()
AMk = AM.shape[0]
SCk = SC.shape[0]
SPk = SP.shape[0]


n_sampleAM = int(AMk)
n_sampleSC = int(SCk)
n_sampleSP = int(SPk)

print(n_sampleAM)
print(n_sampleSC)
print(n_sampleSP)
```

```
668
668
667
```

[16]:
```
n_trainAM = int(0.90*n_sampleAM)+1
n_testAM = int(n_sampleAM - n_trainAM)+1
n_forecastAM = n_sampleAM - n_trainAM
print(f'Apple Music_Train: {n_trainAM}')
print(f'Apple Music_Test: {n_testAM}')
```

```
print(f'Apple Music_Forecast: {n_forecastAM}')
```

```
Apple Music_Train: 602
Apple Music_Test: 67
Apple Music_Forecast: 66
```

[17]:
```
n_trainSC = int(0.90*n_sampleSC)+1
n_testSC = int(n_sampleSC - n_trainSC)+1
n_forecastSC = n_sampleSC - n_trainSC
print(f'SoundCloud_Train: {n_trainSC}')
print(f'SoundCloud_Test: {n_testSC}')
print(f'SoundCloud_Forecast: {n_forecastSC}')
```

```
SoundCloud_Train: 602
SoundCloud_Test: 67
SoundCloud_Forecast: 66
```

[21]:
```
n_trainSP = int(0.90*n_sampleSP)+2
n_testSP = int(n_sampleSP - n_trainSP)+2
n_forecastSP = (n_sampleSP - n_trainSP)+1
print(f'Spotify_Train: {n_trainSP}')
print(f'Spotify_Test: {n_testSP}')
print(f'Spotify_Forecast: {n_forecastSP}')
```

```
Spotify_Train: 602
Spotify_Test: 67
Spotify_Forecast: 66
```

[23]:
```
AM_train = AM[['Platform','lnDiffStreams']].iloc[:602]
SC_train = SC[['Platform','lnDiffStreams']].iloc[:602]
SP_train = SP[['Platform','lnDiffStreams']].iloc[:602]
```

[24]:
```
AM_train
```

[24]:

Date	Platform	lnDiffStreams
2020-03-04	Apple Music	3.931826
2020-03-05	Apple Music	3.931826
2020-03-06	Apple Music	0.000000
2020-03-07	Apple Music	2.944439
2020-03-08	Apple Music	2.944439
…	…	…
2021-10-22	Apple Music	2.708050
2021-10-23	Apple Music	2.708050
2021-10-24	Apple Music	2.708050
2021-10-25	Apple Music	2.708050
2021-10-26	Apple Music	0.693147

[602 rows x 2 columns]

[25]:
```
SC_train
```

[25]:
```
              Platform  lnDiffStreams
Date
2020-03-04  SoundCloud       3.526361
2020-03-05  SoundCloud       3.526361
2020-03-06  SoundCloud       3.433987
2020-03-07  SoundCloud       3.401197
2020-03-08  SoundCloud       3.401197
…                 …             …
2021-10-22  SoundCloud       6.111467
2021-10-23  SoundCloud       6.111467
2021-10-24  SoundCloud       6.111467
2021-10-25  SoundCloud       6.111467
2021-10-26  SoundCloud       4.043051

[602 rows x 2 columns]
```

[26]:
```
SP_train
```

[26]:
```
            Platform  lnDiffStreams
Date
2020-03-04  Spotify       3.401197
2020-03-05  Spotify       3.401197
2020-03-06  Spotify       3.258097
2020-03-07  Spotify       3.258097
2020-03-08  Spotify       3.258097
…               …             …
2021-10-23  Spotify       4.488636
2021-10-24  Spotify       6.182085
2021-10-25  Spotify       6.182085
2021-10-26  Spotify       6.182085
2021-10-27  Spotify       6.182085

[602 rows x 2 columns]
```

[27]:
```
AM_test = AM[['Platform','lnDiffStreams']].iloc[601:]
SC_test = SC[['Platform','lnDiffStreams']].iloc[601:]
SP_test = SP[['Platform','lnDiffStreams']].iloc[600:]
```

[28]:
```
AM_test
```

[28]:
```
               Platform  lnDiffStreams
Date
2021-10-26  Apple Music       0.693147
2021-10-27  Apple Music       0.000000
2021-10-28  Apple Music       2.995732
```

```
2021-10-29  Apple Music       2.639057
2021-10-30  Apple Music       1.098612
…                  …            …
2021-12-27  Apple Music       4.564348
2021-12-28  Apple Music       4.499810
2021-12-29  Apple Music       4.499810
2021-12-30  Apple Music       3.526361
2021-12-31  Apple Music       3.526361

[67 rows x 2 columns]
```

[29]:
```
SC_test
```

[29]:
```
              Platform  lnDiffStreams
Date
2021-10-26  SoundCloud       4.043051
2021-10-27  SoundCloud       4.043051
2021-10-28  SoundCloud       4.043051
2021-10-29  SoundCloud       5.043425
2021-10-30  SoundCloud       5.043425
…                  …            …
2021-12-27  SoundCloud       3.784190
2021-12-28  SoundCloud       4.219508
2021-12-29  SoundCloud       3.367296
2021-12-30  SoundCloud       4.595120
2021-12-31  SoundCloud       4.595120

[67 rows x 2 columns]
```

[30]:
```
SP_test
```

[30]:
```
           Platform  lnDiffStreams
Date
2021-10-26  Spotify       6.182085
2021-10-27  Spotify       6.182085
2021-10-28  Spotify       6.182085
2021-10-29  Spotify       6.182085
2021-10-30  Spotify       6.182085
…              …            …
2021-12-27  Spotify       6.879356
2021-12-28  Spotify       6.188264
2021-12-29  Spotify       6.331502
2021-12-30  Spotify       6.331502
2021-12-31  Spotify       6.331502

[67 rows x 2 columns]
```

[31]:
```
n_sample   = int(n_sampleAM + n_sampleSC + n_sampleSP)
n_train    = int(n_trainAM + n_trainSC +n_trainSP)
n_forecast = int(n_forecastAM + n_forecastSC +n_forecastSP)
n_sample, n_train, n_forecast
```

[31]: (2003, 1806, 198)

[32]:
```
AM_test = AM[['Platform','lnDiffStreams']].iloc[350:]
SC_test = SC[['Platform','lnDiffStreams']].iloc[350:]
SP_test = SP[['Platform','lnDiffStreams']].iloc[350:]
```

[33]:
```
#ts_df
ts_train = pd.concat([AM_train, SC_train, SP_train], ignore_index=False)


ts_test = pd.concat([AM_test, SC_test, SP_test], ignore_index=False)
```

[34]:
```
print(f'Train: \n{ts_train.head()}\n {ts_train.tail()}\n')
print(f'Train Dimensions: \n{ts_train.shape}')
print(f'Test: \n{ts_test.head()}\n {ts_test.tail()}\n')
print(f'Test Dimensions: \n{ts_test.shape}\n')
```

```
Train:
              Platform  lnDiffStreams
Date
2020-03-04  Apple Music       3.931826
2020-03-05  Apple Music       3.931826
2020-03-06  Apple Music       0.000000
2020-03-07  Apple Music       2.944439
2020-03-08  Apple Music       2.944439
            Platform  lnDiffStreams
Date
2021-10-23  Spotify       4.488636
2021-10-24  Spotify       6.182085
2021-10-25  Spotify       6.182085
2021-10-26  Spotify       6.182085
2021-10-27  Spotify       6.182085

Train Dimensions:
(1806, 2)
Test:
              Platform  lnDiffStreams
Date
2021-02-17  Apple Music       3.218876
2021-02-18  Apple Music       1.609438
2021-02-19  Apple Music       1.609438
2021-02-20  Apple Music       4.158883
2021-02-21  Apple Music       4.158883
```

```
            Platform  lnDiffStreams
Date
2021-12-27  Spotify       6.879356
2021-12-28  Spotify       6.188264
2021-12-29  Spotify       6.331502
2021-12-30  Spotify       6.331502
2021-12-31  Spotify       6.331502

Test Dimensions:
(953, 2)
```

[35]:
```
# Source:
    # https://github.com/SimiY...
    # .../pydata-sf-2016-arima-tutorial/blob/master/
 ↪Section_3_ARIMA_Modeling_tutorial.ipynb
#Choose the number of lags to display the sample ACF and PACF;
n_lag=25
#graph_title='Series 1'

# Writing a function to plot this specific sub-section of data;
#Make sure the tsplot() function is defined; #**Function: See dji_citi_sent␣
 ↪file;
def tsplot(y1, y2, lags=None, title='', figsize=(14,8)):
    '''Examine the patterns of ACF and PACF, along with the time series plots␣
 ↪and histogram.

    Original source: https://tomaugspurger.github.io/modern-7-timeseries.html
    '''
    # Plotting layout
    fig = plt.figure(figsize=figsize)
    layout = (2,2)
    ts_ax   = plt.subplot2grid(layout, (0,0))
    hist_ax = plt.subplot2grid(layout, (0,1))
    acf_ax  = plt.subplot2grid(layout, (1,0))
    pacf_ax = plt.subplot2grid(layout, (1,1))

    #Plotting
    y1.plot(ax=ts_ax)
    ts_ax.set_title(title)
    y2.plot(ax=hist_ax, kind='hist', bins=25)
    # Function for rotating tick labels on x-axis only
    for tick in ts_ax.get_xticklabels():
        tick.set_rotation(45)
    ts_ax.legend(loc='upper left')
    hist_ax.set_title('Histogram')
    smt.graphics.plot_acf(y2, lags=lags, ax=acf_ax)
```

```
    smt.graphics.plot_pacf(y2, lags=lags, ax=pacf_ax)
    [ax.set_xlim(0) for ax in [acf_ax, pacf_ax]]
    sns.despine()
    plt.tight_layout()

    return ts_ax, acf_ax, pacf_ax
```

[36]:
```
ts_trainAM = ts_train.query("Platform == 'Apple Music'")
ts_trainSC = ts_train.query("Platform == 'SoundCloud'")
ts_trainSP = ts_train.query("Platform == 'Spotify'")

ts_traindict = {'lnDiffS_AM': ts_trainAM['lnDiffStreams'],
                'lnDiffS_SC': ts_trainSC['lnDiffStreams'],
                'lnDiffS_SP': ts_trainSP['lnDiffStreams']}
ts_trainDSP = pd.DataFrame(ts_traindict)
ts_trainDSP
```

[36]:

Date	lnDiffS_AM	lnDiffS_SC	lnDiffS_SP
2020-03-04	3.931826	3.526361	3.401197
2020-03-05	3.931826	3.526361	3.401197
2020-03-06	0.000000	3.433987	3.258097
2020-03-07	2.944439	3.401197	3.258097
2020-03-08	2.944439	3.401197	3.258097
…	…	…	…
2021-10-23	2.708050	6.111467	4.488636
2021-10-24	2.708050	6.111467	6.182085
2021-10-25	2.708050	6.111467	6.182085
2021-10-26	0.693147	4.043051	6.182085
2021-10-27	NaN	NaN	6.182085

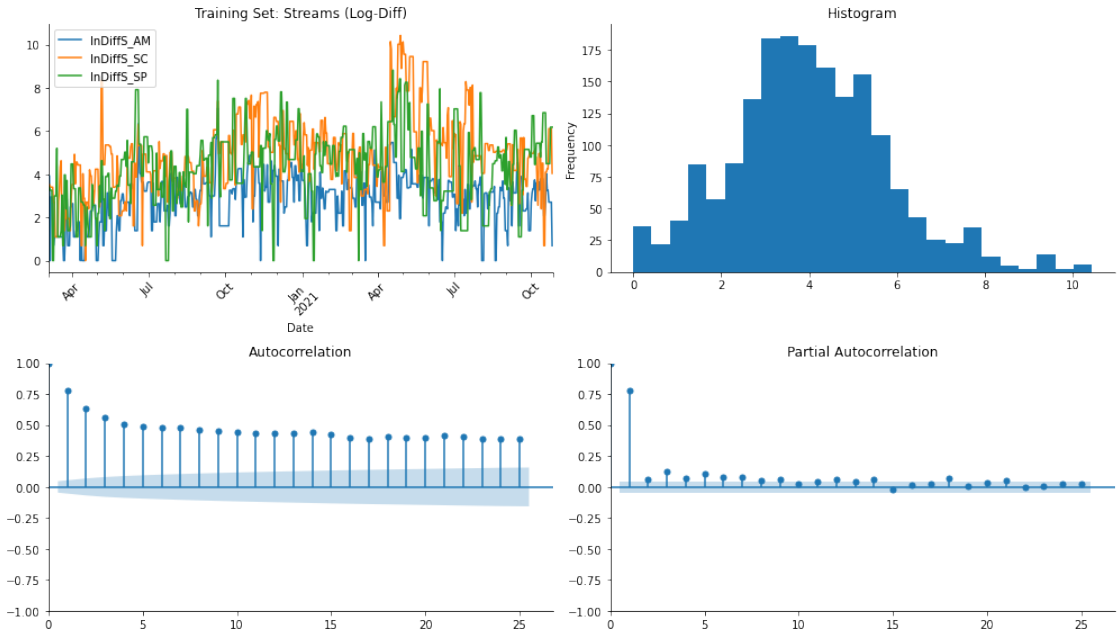
[603 rows x 3 columns]

[37]:
```
#Run command;
tsplot(ts_trainDSP[['lnDiffS_AM','lnDiffS_SC','lnDiffS_SP']],
       ts_train['lnDiffStreams'],
       title='Training Set: Streams (Log-Diff)' ,
       lags=n_lag)
```

```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/graphics/tsaplots.py:348:
FutureWarning: The default method 'yw' can produce PACF values outside of the
[-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker
('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```

[37]:
```
(<AxesSubplot:title={'center':'Training Set: Streams (Log-Diff)'},
xlabel='Date'>,
 <AxesSubplot:title={'center':'Autocorrelation'}>,
```

```
<AxesSubplot:title={'center':'Partial Autocorrelation'}>)
```



[38]:
```
ts_testAM = ts_test.query("Platform == 'Apple Music'")
ts_testSC = ts_test.query("Platform == 'SoundCloud'")
ts_testSP = ts_test.query("Platform == 'Spotify'")

ts_testdict = {'lnDiffS_AM': ts_testAM['lnDiffStreams'],
               'lnDiffS_SC': ts_testSC['lnDiffStreams'],
               'lnDiffS_SP': ts_testSP['lnDiffStreams']}
ts_testDSP = pd.DataFrame(ts_testdict)
ts_testDSP
```

[38]:

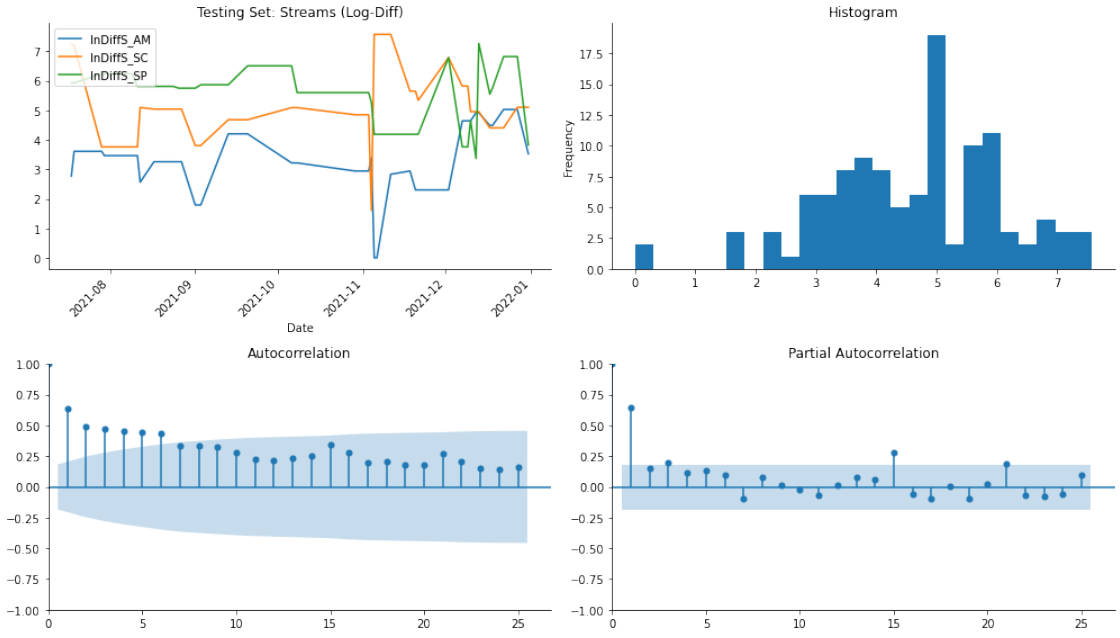
Date	lnDiffS_AM	lnDiffS_SC	lnDiffS_SP
2021-02-17	3.218876	3.555348	5.710427
2021-02-18	1.609438	5.880533	5.710427
2021-02-19	1.609438	5.880533	6.226537
2021-02-20	4.158883	5.880533	6.226537
2021-02-21	4.158883	5.880533	6.226537
…	…	…	…
2021-12-27	4.564348	3.784190	6.879356
2021-12-28	4.499810	4.219508	6.188264
2021-12-29	4.499810	3.367296	6.331502
2021-12-30	3.526361	4.595120	6.331502
2021-12-31	3.526361	4.595120	6.331502

```
[318 rows x 3 columns]
```

[76]:
```
tsplot(ts_testDSP[['lnDiffS_AM','lnDiffS_SC','lnDiffS_SP']],
       ts_test['lnDiffStreams'],
       title='Testing Set: Streams (Log-Diff)' ,
       lags=n_lag)
```

```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/graphics/tsaplots.py:348:
FutureWarning: The default method 'yw' can produce PACF values outside of the
[-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker
('ywm'). You can use this method now by setting method='ywm'.
  warnings.warn(
```

[76]:
```
(<AxesSubplot:title={'center':'Testing Set: Streams (Log-Diff)'},
 xlabel='Date'>,
  <AxesSubplot:title={'center':'Autocorrelation'}>,
  <AxesSubplot:title={'center':'Partial Autocorrelation'}>)
```



[39]:
```
# Vars.

print(f'Length, Train: \n{len(ts_train)}\n')
print(f'Length, Test: \n{len(ts_test)}\n')
```

```
Length, Train:
1806

Length, Test:
```

```
953
```

[40]:
```
ts_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1806 entries, 2020-03-04 to 2021-10-27
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Platform       1806 non-null   object
 1   lnDiffStreams  1806 non-null   float64
dtypes: float64(1), object(1)
memory usage: 42.3+ KB
```

[41]:
```
ts_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 953 entries, 2021-02-17 to 2021-12-31
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Platform       953 non-null    object
 1   lnDiffStreams  953 non-null    float64
dtypes: float64(1), object(1)
memory usage: 22.3+ KB
```

[42]:
```
ts_train.dtypes
```

[42]:
```
Platform          object
lnDiffStreams    float64
dtype: object
```

[43]:
```
ts_test.dtypes
```

[43]:
```
Platform          object
lnDiffStreams    float64
dtype: object
```

```
# ARIMA Model Attempt:
# Estimating 'lnDiffStreams' on its own;
```

[44]:
```
type(ts_train['lnDiffStreams'].index)
```

[44]:
```
pandas.core.indexes.datetimes.DatetimeIndex
```

[50]:
```
with pd.option_context('display.max_rows', None):
    print(ts_train.index)
```

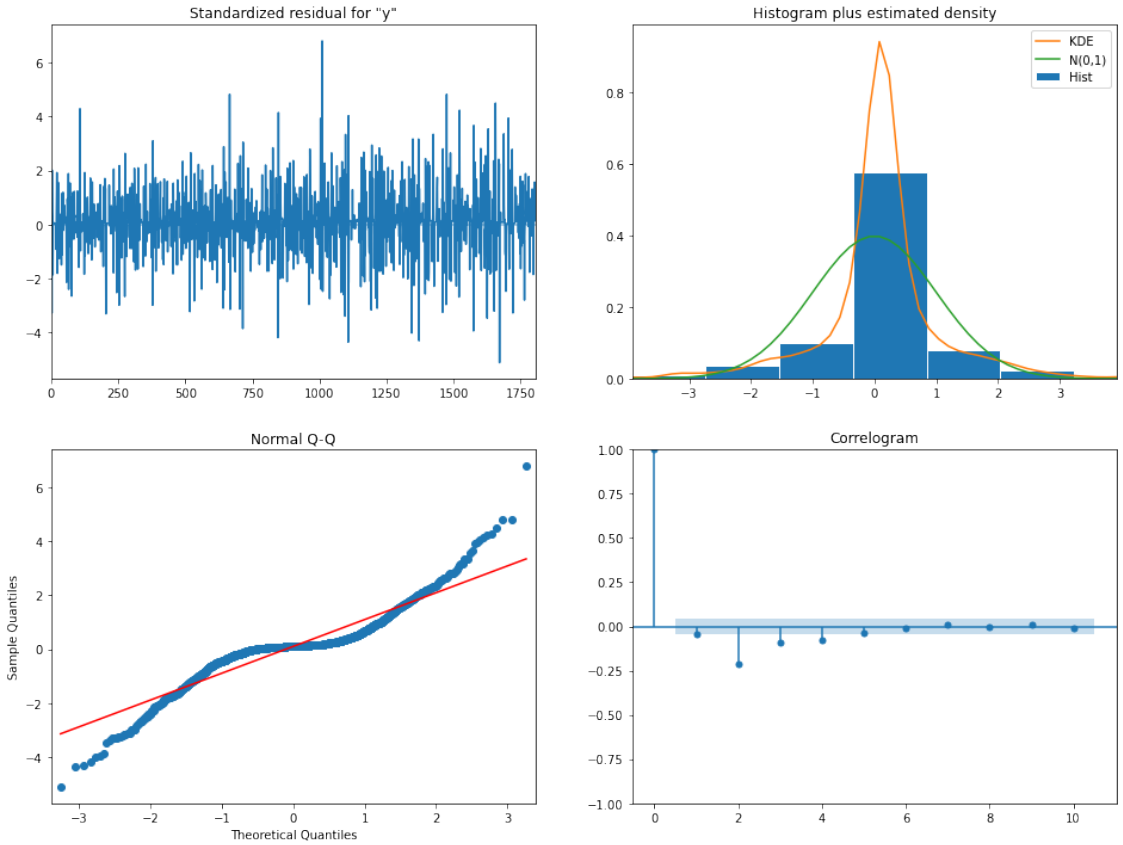
```
DatetimeIndex(['2020-03-04', '2020-03-05', '2020-03-06', '2020-03-07',
               '2020-03-08', '2020-03-09', '2020-03-10', '2020-03-11',
               '2020-03-12', '2020-03-13',
               …
               '2021-10-18', '2021-10-19', '2021-10-20', '2021-10-21',
               '2021-10-22', '2021-10-23', '2021-10-24', '2021-10-25',
               '2021-10-26', '2021-10-27'],
              dtype='datetime64[ns]', name='Date', length=1806, freq=None)
```

[45]:
```
#Model Estimation

# Fit the model
arima200 = sm.tsa.SARIMAX(ts_train['lnDiffStreams'].to_numpy(), order=(2,0,0))
model_results = arima200.fit(disp=0)
model_results.summary()
```

[45]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                              SARIMAX Results
==========================================================================
Dep. Variable:                        y   No. Observations:                 1806
Model:                SARIMAX(2, 0, 0)   Log Likelihood               -2835.521
Date:                Fri, 04 Feb 2022   AIC                             5677.043
Time:                        17:34:26   BIC                             5693.539
Sample:                             0   HQIC                            5683.131
                                - 1806
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1          0.8158      0.021     38.373      0.000       0.774       0.858
ar.L2          0.1537      0.021      7.451      0.000       0.113       0.194
sigma2         1.3508      0.025     54.219      0.000       1.302       1.400
==========================================================================
===
Ljung-Box (L1) (Q):                   3.16   Jarque-Bera (JB):
2180.60
Prob(Q):                              0.08   Prob(JB):
0.00
Heteroskedasticity (H):               1.77   Skew:
0.04
Prob(H) (two-sided):                  0.00   Kurtosis:
8.38
==========================================================================
===

Warnings:
```

```
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

[46]:
```
# Residual Diagnostics
# The plot_diagnostics function associated with the estimated result object␣
 ↪produce a few plots that allow us
# to examine the distribution and correlation of the estimated residuals

model_results.plot_diagnostics(figsize=(16, 12));
```



[47]:
```
# Re-run the above statistical tests, and more. To be used when selecting␣
 ↪viable models.

het_method='breakvar'
norm_method='jarquebera'
sercor_method='ljungbox'

(het_stat, het_p) = model_results.test_heteroskedasticity(het_method)[0]
norm_stat, norm_p, skew, kurtosis = model_results.test_normality(norm_method)[0]
```

```
sercor_stat, sercor_p = model_results.
 ↪test_serial_correlation(method=sercor_method)[0]
sercor_stat = sercor_stat[-1] # last number for the largest lag
sercor_p = sercor_p[-1] # last number for the largest lag

# Run Durbin-Watson test on the standardized residuals.
# The statistic is approximately equal to 2*(1-r), where r is the sample␣
 ↪autocorrelation of the residuals.
# Thus, for r == 0, indicating no serial correlation, the test statistic equals␣
 ↪2.
# This statistic will always be between 0 and 4. The closer to 0 the statistic,
# the more evidence for positive serial correlation. The closer to 4,
# the more evidence for negative serial correlation.
# Essentially, below 1 or above 3 is bad.
dw = sm.stats.stattools.durbin_watson(model_results.filter_results.
 ↪standardized_forecasts_error[0, model_results.loglikelihood_burn:])

# check whether roots are outside the unit circle (we want them to be);
# will be True when AR is not used (i.e., AR order = 0)
arroots_outside_unit_circle = np.all(np.abs(model_results.arroots) > 1)
# will be True when MA is not used (i.e., MA order = 0)
maroots_outside_unit_circle = np.all(np.abs(model_results.maroots) > 1)

print('Test heteroskedasticity of residuals ({}): stat={:.3f}, p={:.3f}'.
 ↪format(het_method, het_stat, het_p));
print('\nTest normality of residuals ({}): stat={:.3f}, p={:.3f}'.
 ↪format(norm_method, norm_stat, norm_p));
print('\nTest serial correlation of residuals ({}): stat={:.3f}, p={:.3f}'.
 ↪format(sercor_method, sercor_stat, sercor_p));
print('\nDurbin-Watson test on residuals: d={:.2f}\n\t(NB: 2 means no serial␣
 ↪correlation, 0=pos, 4=neg)'.format(dw))
print('\nTest for all AR roots outside unit circle (>1): {}'.
 ↪format(arroots_outside_unit_circle))
print('\nTest for all MA roots outside unit circle (>1): {}'.
 ↪format(maroots_outside_unit_circle))
```

```
Test heteroskedasticity of residuals (breakvar): stat=1.765, p=0.000

Test normality of residuals (jarquebera): stat=2180.602, p=0.000

Test serial correlation of residuals (ljungbox): stat=115.703, p=0.000

Durbin-Watson test on residuals: d=2.06
        (NB: 2 means no serial correlation, 0=pos, 4=neg)

Test for all AR roots outside unit circle (>1): True
```

```
Test for all MA roots outside unit circle (>1): True
```

[140]:
```
1806 + 953
```

[140]:
```
2759
```

[151]:
```
ts_train
```

[151]:
```
               Platform  lnDiffStreams
Date
2020-03-04  Apple Music       3.931826
2020-03-05  Apple Music       3.931826
2020-03-06  Apple Music       0.000000
2020-03-07  Apple Music       2.944439
2020-03-08  Apple Music       2.944439
…                  …            …
2021-10-23      Spotify       4.488636
2021-10-24      Spotify       6.182085
2021-10-25      Spotify       6.182085
2021-10-26      Spotify       6.182085
2021-10-27      Spotify       6.182085

[1806 rows x 2 columns]
```

[158]:
```
pred_ci
```

[158]:
```
array([[ 3.71574428,  8.27171265],
       [ 2.90013133,  8.77998603],
       [ 2.20344737,  9.16803274],
       …,
       [-8.66988735,  8.66988735],
       [-8.66988735,  8.66988735],
       [-8.66988735,  8.66988735]])
```

[202]:
```
pred_ci[0:904]
```

[202]:
```
array([[-8.66988735,  8.66988735],
       [ 1.48490342,  6.09564883],
       [ 1.53404615,  6.09001452],
       …,
       [ 2.2704272 ,  6.82639556],
       [ 1.78336463,  6.339333  ],
       [ 3.91534093,  8.47130929]])
```

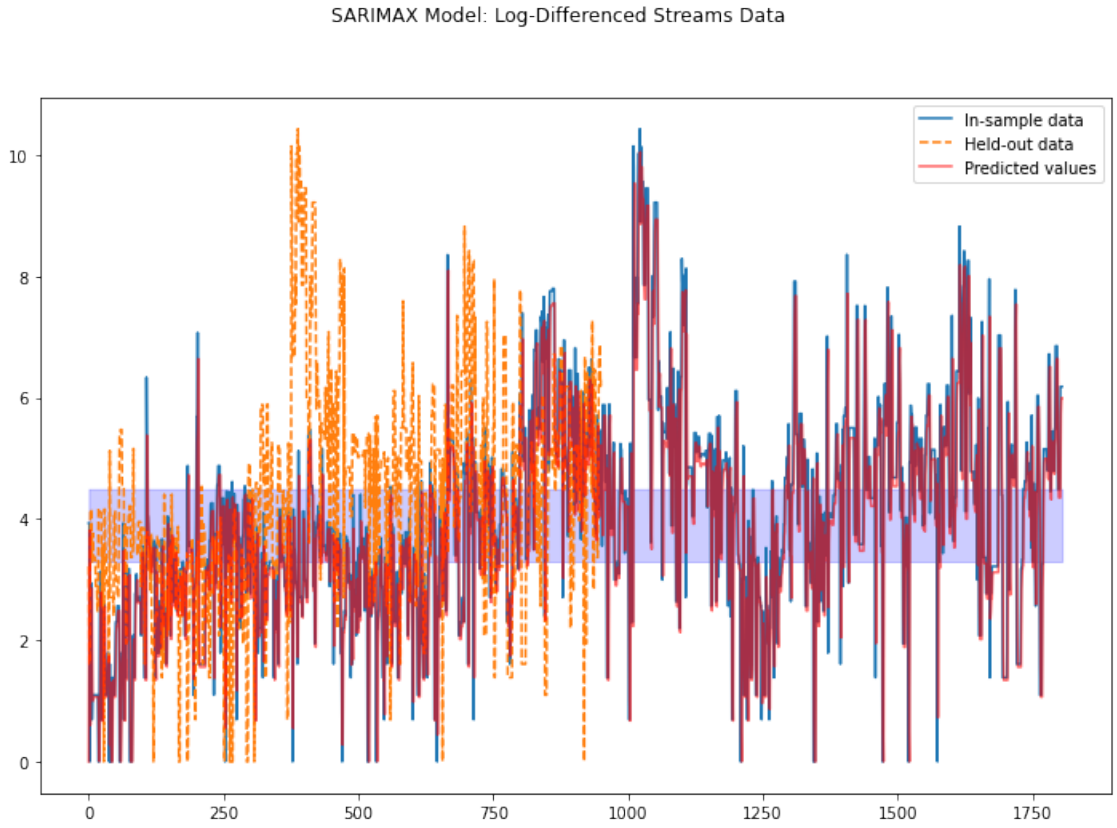
[203]:
```
pred_ci[904:1807]
```

[203]:
```
array([[2.25925793, 6.8152263 ],
       [1.86838   , 6.42434837],
```

```
       [1.86838   , 6.42434837],
       …,
       [3.71574428, 8.27171265],
       [3.71574428, 8.27171265],
       [3.71574428, 8.27171265]])
```

[233]:
```
fig, ax1 = plt.subplots(nrows=1, ncols=1, figsize=(12, 8))

ax1.plot(ts_train['lnDiffStreams'].to_numpy(), label='In-sample data',␣
↪linestyle='-')
# subtract 1 only to connect it to previous point in the graph
ax1.plot(ts_test['lnDiffStreams'].to_numpy(), label='Held-out data',␣
↪linestyle='--')

# yes DatetimeIndex
pred_begin = 0
pred_end = 1805
pred = model_results.get_prediction(start=pred_begin,end=pred_end, freq='d')
pred_mean = pred.predicted_mean
pred_ci = pred.conf_int(alpha=0.05)



ax1.plot(pred_mean, 'r', alpha=.6, label='Predicted values')
ax1.fill_between(range(len(pred_mean)), pred_ci[0:904].mean(), pred_ci[904:
↪1807].mean(), color='b', alpha=.2)

ax1.legend(loc='best')
fig.suptitle('SARIMAX Model: Log-Differenced Streams Data')
```

[233]:
```
Text(0.5, 0.98, 'SARIMAX Model: Log-Differenced Streams Data')
```



[239]:
```
def get_rmse(y, y_hat):
    '''Root Mean Square Error
    https://en.wikipedia.org/wiki/Root-mean-square_deviation
    '''
    mse = np.mean((y - y_hat)**2)
    return np.sqrt(mse)

def get_mape(y, y_hat):
    '''Mean Absolute Percent Error
    https://en.wikipedia.org/wiki/Mean_absolute_percentage_error
    '''
    perc_err = (100*(y - y_hat))/y
    return np.mean(abs(perc_err))

def get_mase(y, y_hat):
    '''Mean Absolute Scaled Error
    https://en.wikipedia.org/wiki/Mean_absolute_scaled_error
    '''
    abs_err = abs(y - y_hat)
    dsum=sum(abs(y[1:] - y_hat[1:]))
    t = len(y)
```

```
    denom = (1/(t - 1))* dsum
    return np.mean(abs_err/denom)

rmse = get_rmse(ts_train['lnDiffStreams'], pred_mean)
print("RMSE: ", rmse)

mape = get_mape(ts_train['lnDiffStreams'], pred_mean)
print("MAPE: ", mape)

mase = get_mase(ts_train['lnDiffStreams'], pred_mean)
print("MASE: ", mase)
```

```
RMSE:  1.1656885116331923
MAPE:  inf
MASE:  1.0025678286868618
```

[ ]: