Financial Forecast Sandesh Brand 4

Data Analysis:

1) **Dataset taken: trainv.csv**

   The dataset contains 66 instances with 10 attributes including the class.

   For Each Attribute: (all numeric-valued)

1. Generic LookupKey
2. Segment 2Sandesh Brand 1MobileLeopardClosing Base
3. Segment 2Sandesh Brand 1MobileLeopardLeavers
4. Segment 2Sandesh Brand 1MobilePantherClosing Base
5. Segment 2Sandesh Brand 1MobilePantherLeavers
6. Segment 2Sandesh Brand 1MobilePantherGross Adds
7. Segment 2Sandesh Brand 1MobileHyenaClosing Base
8. Segment 2Sandesh Brand 1MobileHyenaLeavers
9. Segment 2Sandesh Brand 1MobileHyenaGross Adds
10. Segment 2Sandesh Brand 1MobilePanther - Leopard - HyenaTotal Revenue

Model used to predict/forecast the future values is SARIMA model

SARIMA Model

Up until now, we have not considered the effect of seasonality in time series. However, this behavior is surely present in many cases, such as gift shop sales, or total number of air passengers.

A seasonal ARIMA model or SARIMA is written as follows:

SARIMA (p,d,q)(P,D,Q)m

You can see that we add P, D, and Q for the seasonal portion of the time series. They are the same terms as the non-seasonal components, by they involve backshifts of the seasonal period.

In the formula above, m is the number of observations per year or the period. If we are analyzing quarterly data, m would equal 4.

Steps involved in Project:

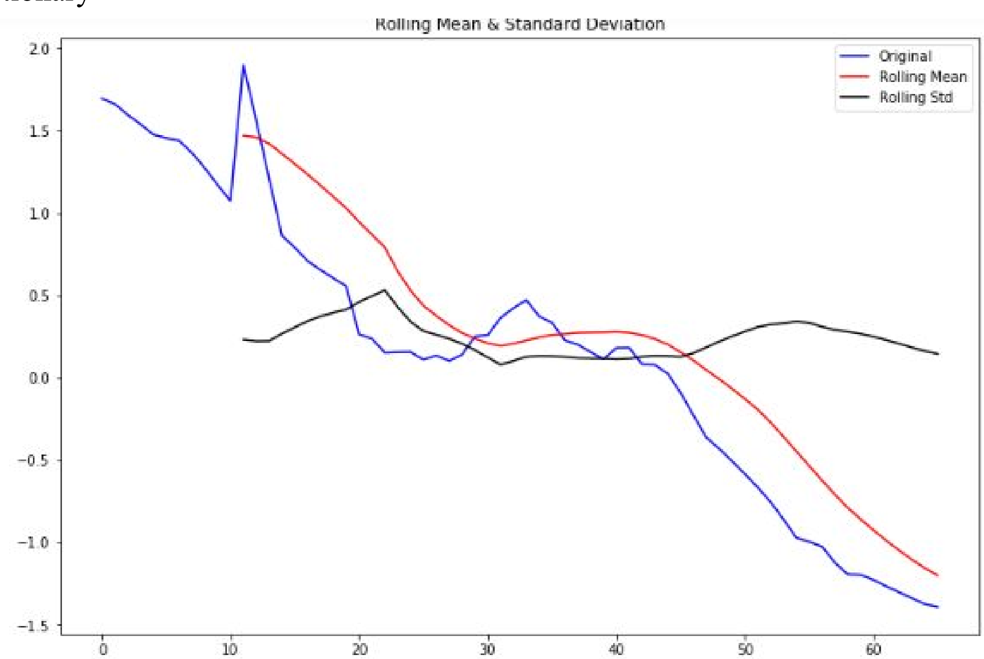
Step 1: Initializing the libraries

```
In [143]: import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
from time import gmtime, strftime
from pylab import rcParams
import statsmodels.api as sm
import itertools
from statsmodels.tsa.statespace.sarimax import SARIMAX
from itertools import product
from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.graphics.tsaplots import plot_acf

from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import adfuller

from tqdm import tqdm_notebook
from itertools import product

%matplotlib inline
```

Step 2: Reading the CSV file using panda library

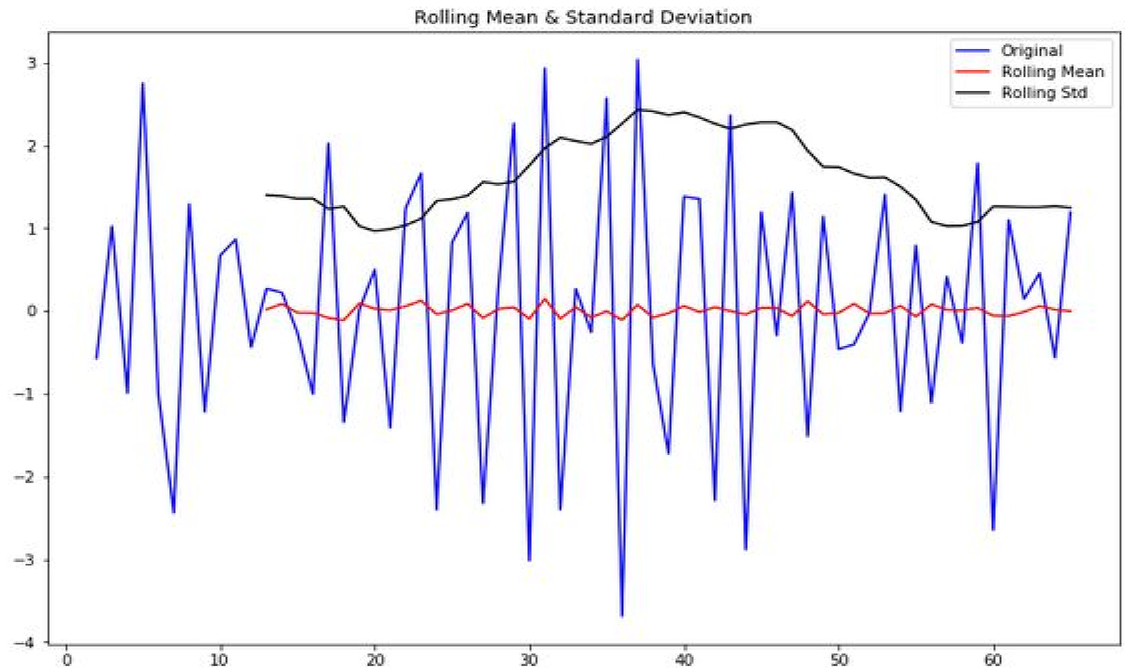
```
In [144]: df = pd.read_csv("trainv.csv")

In [145]: df.columns

Out[145]: Index(['Generic LookupKey',
                 'Segment 2Sandesh Brand 1MobileLeopardClosing Base',
                 'Segment 2Sandesh Brand 1MobileLeopardLeavers',
                 'Segment 2Sandesh Brand 1MobileLeopardGross Adds',
                 'Segment 2Sandesh Brand 1MobilePantherClosing Base',
                 'Segment 2Sandesh Brand 1MobilePantherLeavers',
                 'Segment 2Sandesh Brand 1MobilePantherGross Adds',
                 'Segment 2Sandesh Brand 1MobileHyenaClosing Base',
                 'Segment 2Sandesh Brand 1MobileHyenaLeavers',
                 'Segment 2Sandesh Brand 1MobileHyenaGross Adds',
                 'Segment 2Sandesh Brand 1MobilePanther - Leopard - HyenaTotal Revenue'],
                dtype='object')
```

Step 3: Checking whether the columns which are to be predicted are Stationary or Non-Stationary




```
Results of Dickey-Fuller Test:
p-value = 0.8738. The series is likely non-stationary.
Test Statistic                 -0.587389
p-value                         0.873806
#Lags Used                      0.000000
Number of Observations Used    65.000000
Critical Value (1%)            -3.535217
Critical Value (5%)            -2.907154
Critical Value (10%)           -2.591103
dtype: float64
```

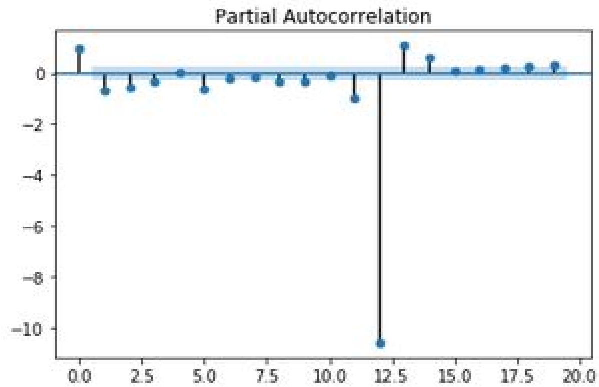
Step 4: Convert the Non-Stationary column to Stationary (Note: Check p-value i.e. p-value<= 0.01)

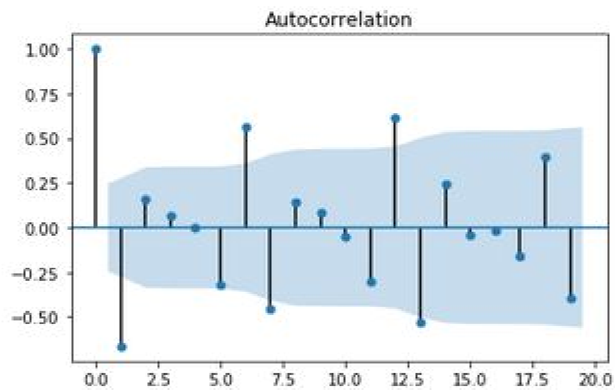



```
Results of Dickey-Fuller Test:
p-value = 0.0000. The series is likely stationary.
Test Statistic                -5.648931e+00
p-value                        9.963694e-07
#Lags Used                     1.000000e+01
Number of Observations Used    5.300000e+01
Critical Value (1%)           -3.560242e+00
Critical Value (5%)           -2.917850e+00
Critical Value (10%)          -2.596796e+00
dtype: float64
```

Step 5: Plot the Auto-Correlation and Correlation

In [206]:
```
plot_pacf(first_diff_3);
plot_acf(first_diff_3);
```





Step 6: Finding the best optimal values of AIC for p,d,q values for SARIMA model

```
In [209]: result_df3 = optimize_SARIMA(parameters_list, 1, 1, 4, first_diff_3)
          result_df3
```

100% 256/256 [19:03<00:00, 4.47s/it]

Out[209]:

	(p,q)x(P,Q)	AIC
0	(1, 2, 2, 0)	156.030542
1	(1, 2, 2, 1)	156.582222
2	(1, 2, 3, 0)	156.631582
3	(0, 3, 2, 0)	157.014988
4	(3, 1, 2, 1)	157.609832
...	...	...
251	(0, 0, 0, 1)	290.789225
252	(0, 0, 1, 2)	291.564348
253	(1, 0, 0, 0)	307.161654
254	(0, 0, 1, 0)	312.932649
255	(0, 0, 0, 0)	333.082697

256 rows × 2 columns

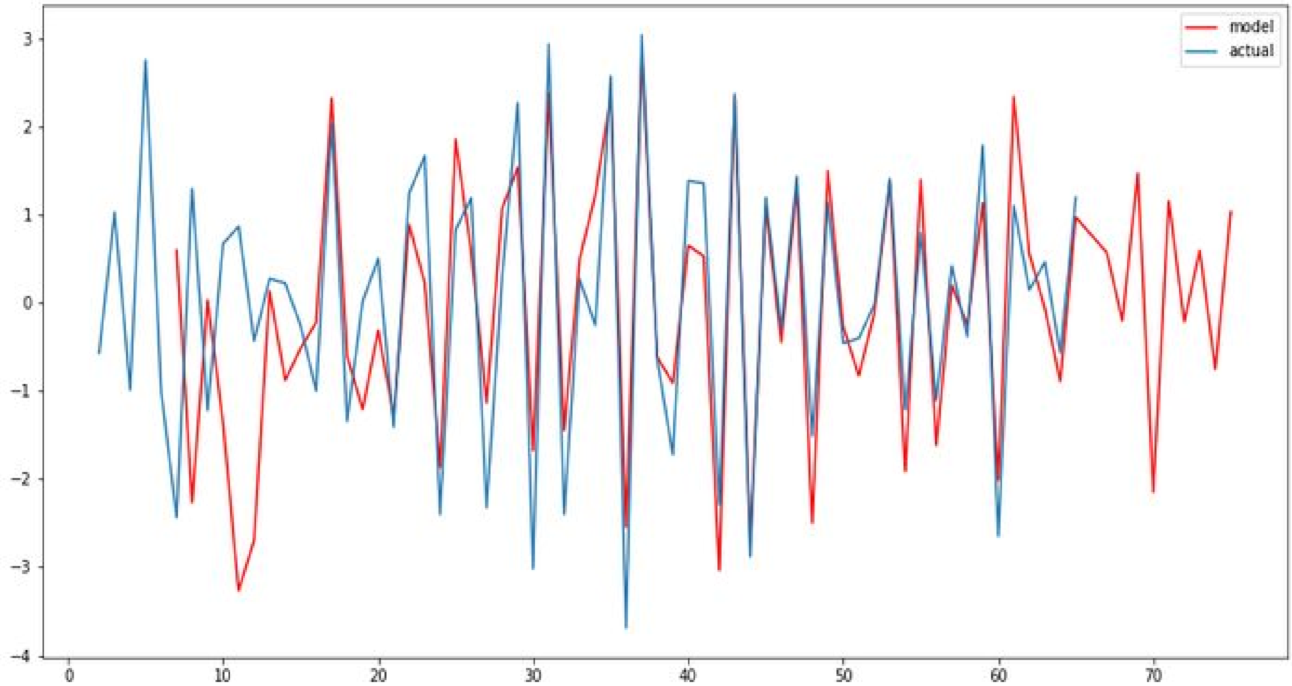
Step 7: Pass the optimal p,d,q values to SARIMA model for fitting

```
best_model3 = SARIMAX(first_diff_3, order=(1, 1, 2), seasonal_order=(2, 1, 0, 4)).fit(dis=-1)
print(best_model3.summary())
```

```
                                          SARIMAX Results
=====================================================================================================
Dep. Variable:     Segment 2Sandesh Brand 1MobileLeopardGross Adds   No. Observations:            64
Model:                           SARIMAX(1, 1, 2)x(2, 1, [], 4)   Log Likelihood           -72.015
Date:                                          Fri, 31 Jul 2020   AIC                      156.031
Time:                                                  14:44:57   BIC                      168.496
Sample:                                                       0   HQIC                     160.896
                                                           - 64
Covariance Type:                                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.3220      0.134     -2.408      0.016      -0.584      -0.060
ma.L1         -1.9854      3.962     -0.501      0.616      -9.752       5.781
ma.L2          0.9881      3.898      0.253      0.800      -6.652       8.628
ar.S.L4       -1.0475      0.080    -13.141      0.000      -1.204      -0.891
ar.S.L8       -0.7839      0.084     -9.293      0.000      -0.949      -0.619
sigma2         0.4303      1.777      0.242      0.809      -3.052       3.913
===================================================================================
Ljung-Box (Q):                       38.81   Jarque-Bera (JB):                 0.95
Prob(Q):                              0.52   Prob(JB):                         0.62
Heteroskedasticity (H):               0.51   Skew:                            -0.17
Prob(H) (two-sided):                  0.14   Kurtosis:                         2.48
===================================================================================
```

Step 8: Plot the SARIMA model

Here, the predicted values are represented by red lines and the actual values are indicated by blue lines.



Step 9: You can see the predicted/forecasted values in the code as shown in the screenshot for duration between October 2019 - September 2020

```
output=output.join(pd.DataFrame({"Segment 2Sandesh Brand 1MobileLeopardGross Adds": x3.tolist()}))
```

```
output
```

	Time Period	Segment 2Sandesh Brand 1MobileLeopardClosing Base	Segment 2Sandesh Brand 1MobileLeopardLeavers	Segment 2Sandesh Brand 1MobileLeopardGross Adds
0	2019-10-01	-0.071075	0.615386	0.625146
1	2019-11-01	-0.057562	-0.018498	-0.275281
2	2019-12-01	-0.083158	0.480947	1.784225
3	2020-01-01	0.015990	-1.026377	-2.476061
4	2020-02-01	-0.071253	0.869960	1.184587
5	2020-03-01	-0.057741	-0.312906	-0.118131
6	2020-04-01	-0.083337	0.696186	0.952406
7	2020-05-01	0.015812	-0.714301	-1.121990
8	2020-06-01	-0.071432	0.183915	1.136452
9	2020-07-01	-0.057919	0.472690	-0.845416
10	2020-08-01	-0.083515	0.087065	1.728466
11	2020-09-01	0.015633	-0.764249	-2.210319