In this assignment students have to make ARIMA model over shampoo sales data and

check the MSE between predicted and actual value.

Student can download data in .csv format from the following link:

https://datamarket.com/data/set/22r0/sales-of-shampoo-over-a-three-year-

period#!ds=22r0&display=line

Hint:

Following is the command import packages and data

from pandas import read\_csv

from pandas import datetime

from matplotlib import pyplot

from statsmodels.tsa.arima\_model import ARIMA

from sklearn.metrics import mean\_squared\_error

def parser(x):

return datetime.strptime('190'+x, '%Y-%m')

series = read\_csv('shampoo-sales.csv', header=0, parse\_dates=[0], index\_col=0,

squeeze=True, date\_parser=parser)

NOTE: The solution shared through Github should contain the source code used and

the screenshot of the output.

Code :

*#importing the important libraries*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**from** **statsmodels.tsa.arima\_model** **import** ARIMA

**from** **sklearn.metrics** **import** mean\_squared\_error

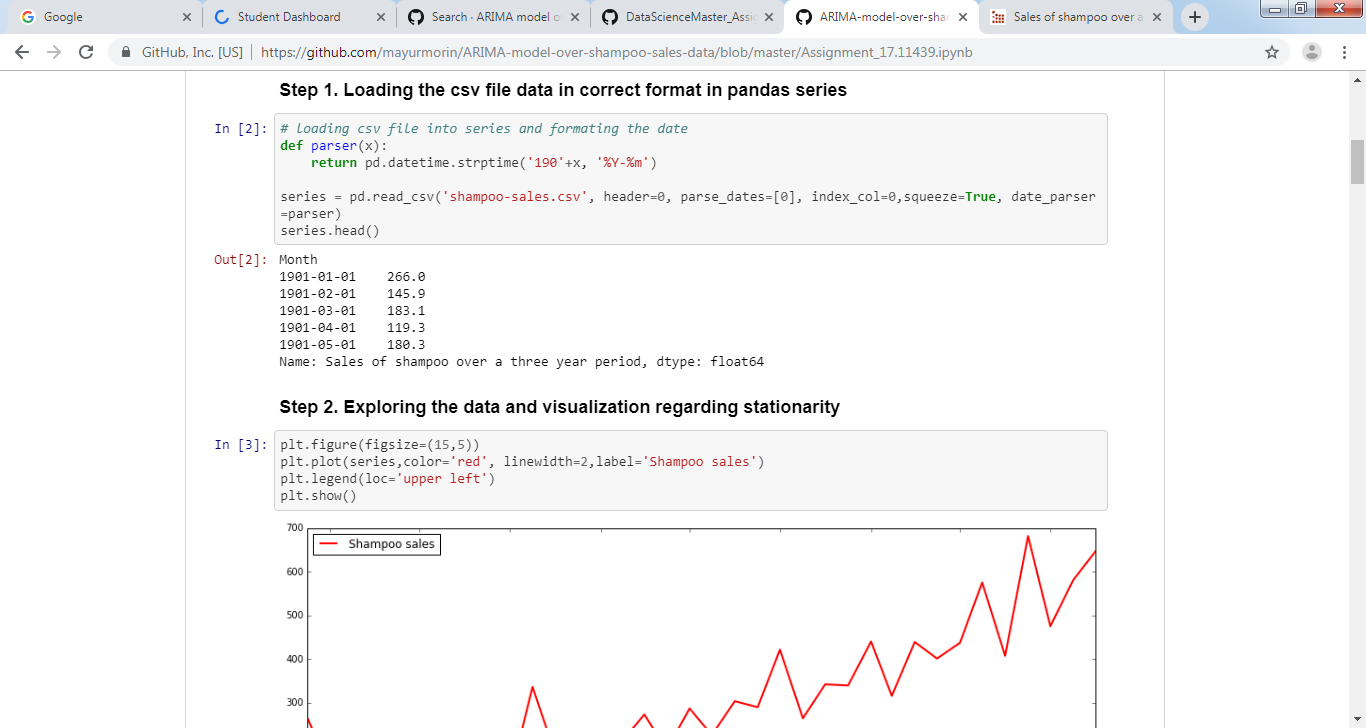
*# loading csv file into series and formating the date*

**def** parser(x):

**return** pd.datetime.strptime('190'+x, '%Y-%m')

series = pd.read\_csv('shampoo-sales.csv', header=0, parse\_dates=[0], index\_col=0,squeeze=**True**, date\_parser=parser)

series.head()

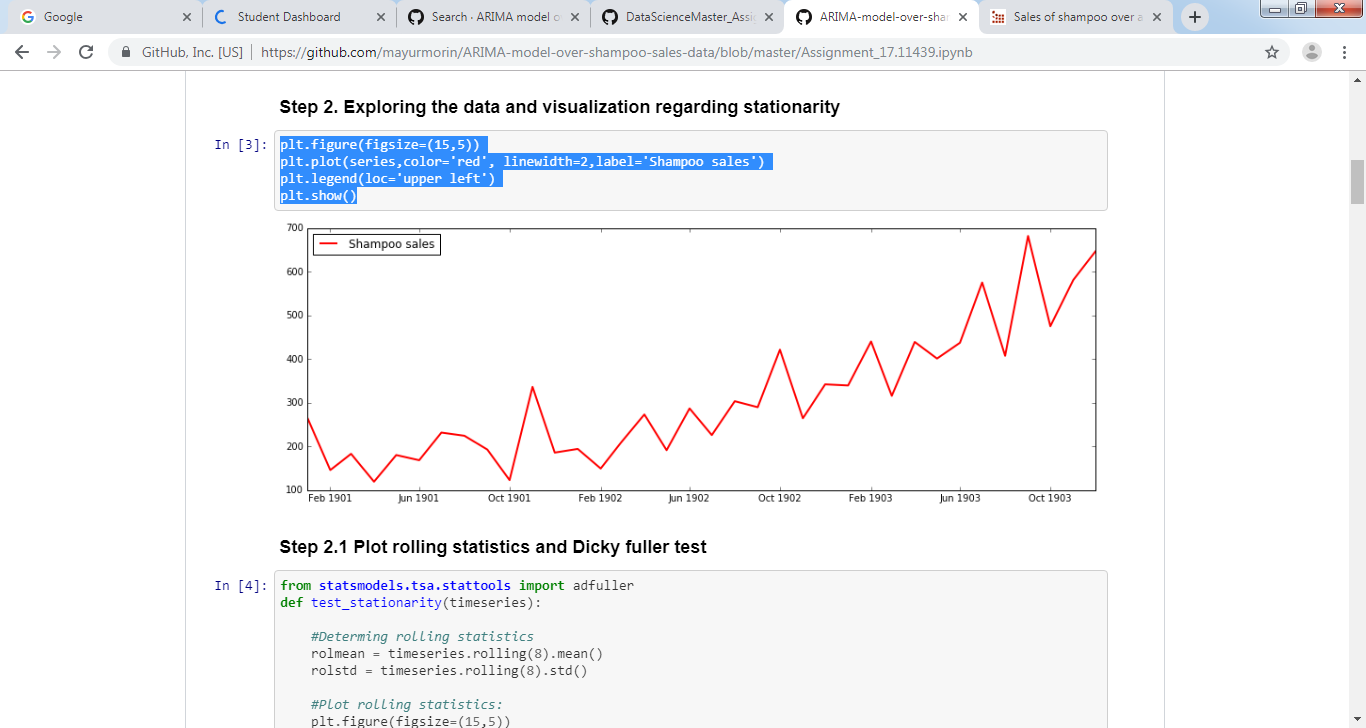


plt.figure(figsize=(15,5))

plt.plot(series,color='red', linewidth=2,label='Shampoo sales')

plt.legend(loc='upper left')

plt.show()



**from** **statsmodels.tsa.stattools** **import** adfuller

**def** test\_stationarity(timeseries):

*#Determing rolling statistics*

rolmean = timeseries.rolling(8).mean()

rolstd = timeseries.rolling(8).std()

*#Plot rolling statistics:*

plt.figure(figsize=(15,5))

orig = plt.plot(timeseries, color='blue',label='Original')

mean = plt.plot(rolmean, color='red', label='Rolling Mean')

std = plt.plot(rolstd, color='black', label = 'Rolling Std')

plt.legend(loc='best')

plt.title('Rolling Mean & Standard Deviation')

plt.show(block=**False**)

*#Perform Dickey-Fuller test:*

print('Results of Dickey-Fuller Test:')

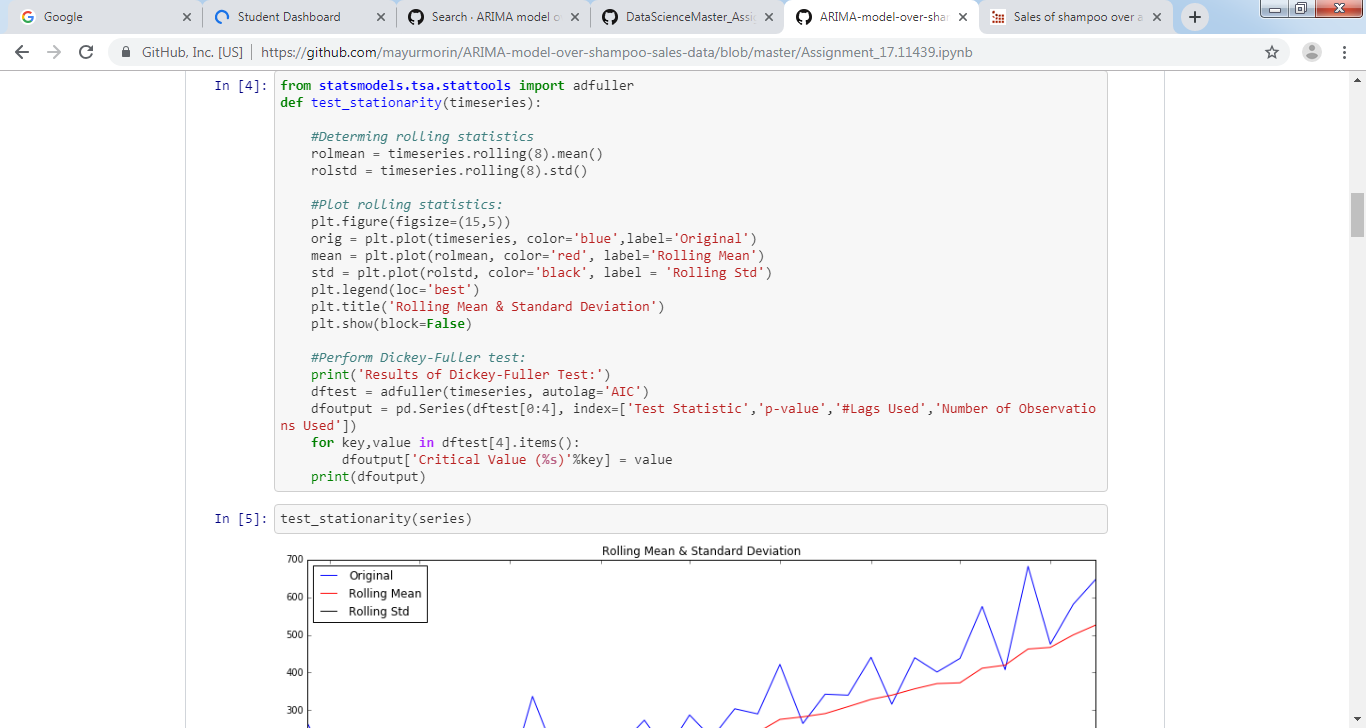
dftest = adfuller(timeseries, autolag='AIC')

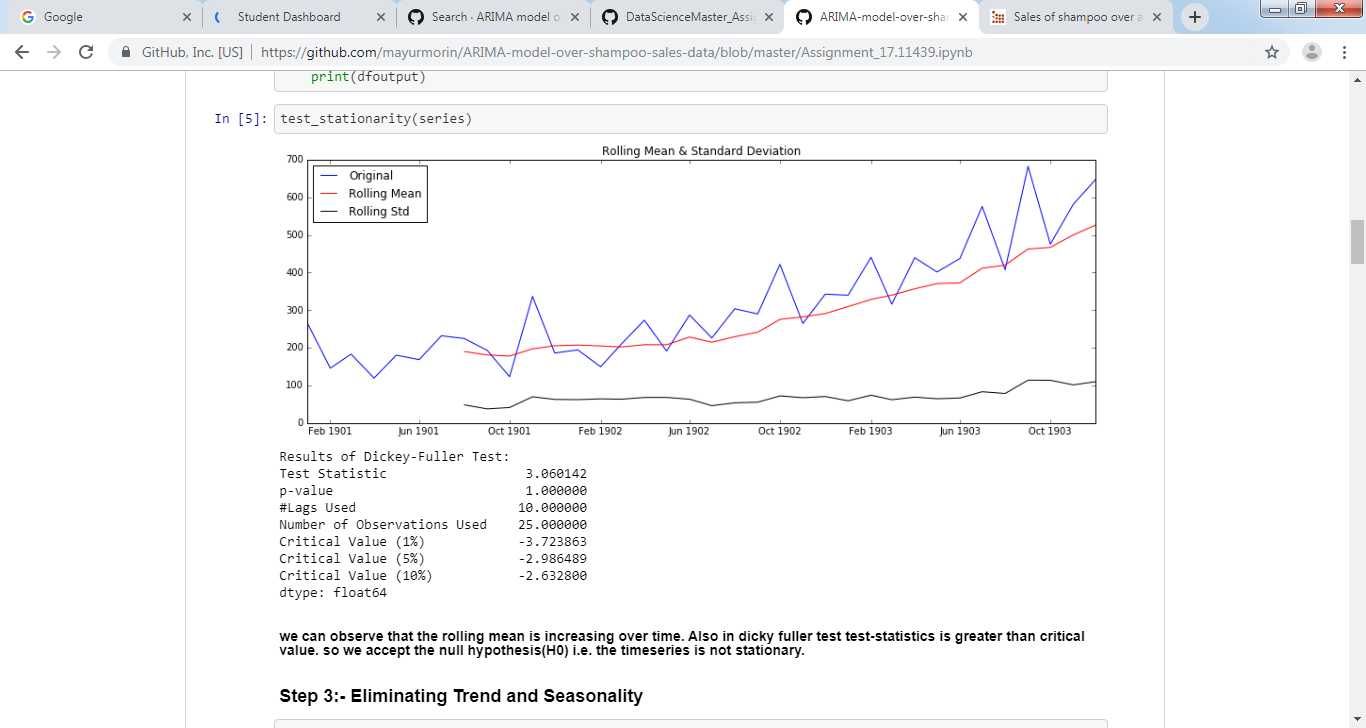
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

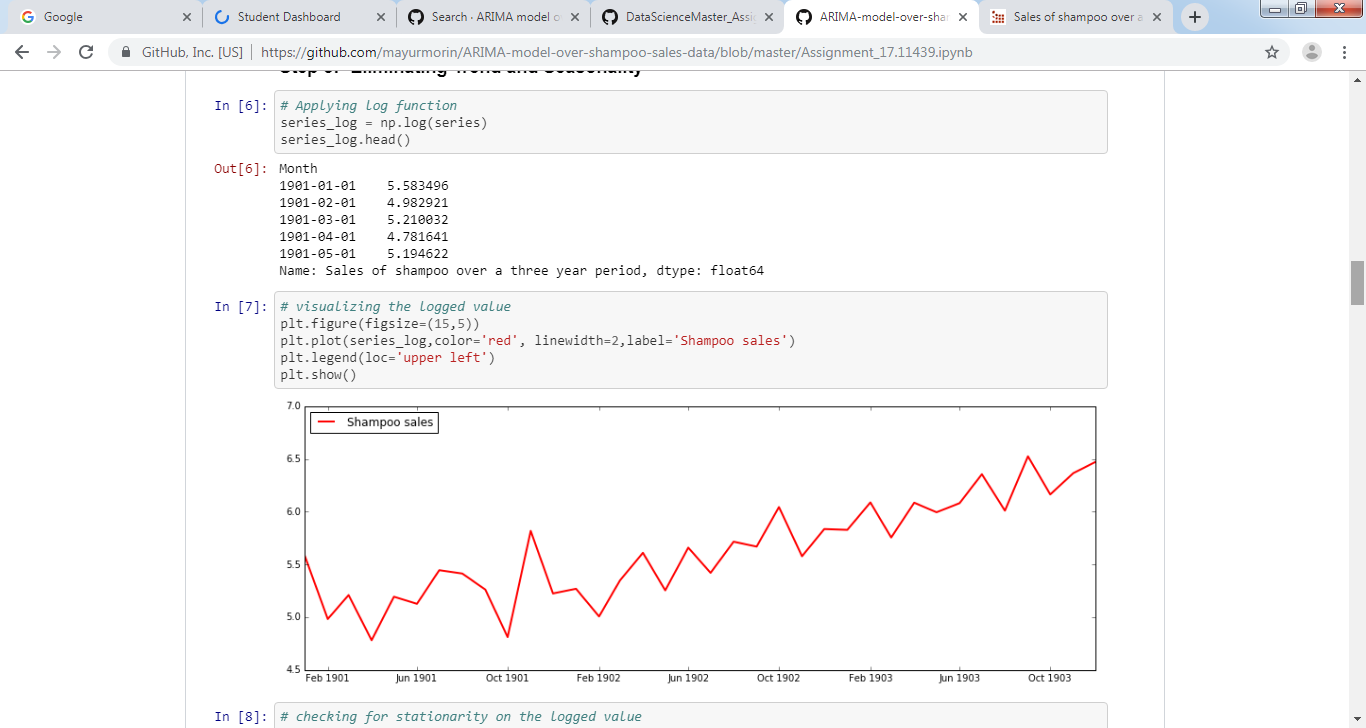
**for** key,value **in** dftest[4].items():

dfoutput['Critical Value (**%s**)'%key] = value

print(dfoutput)

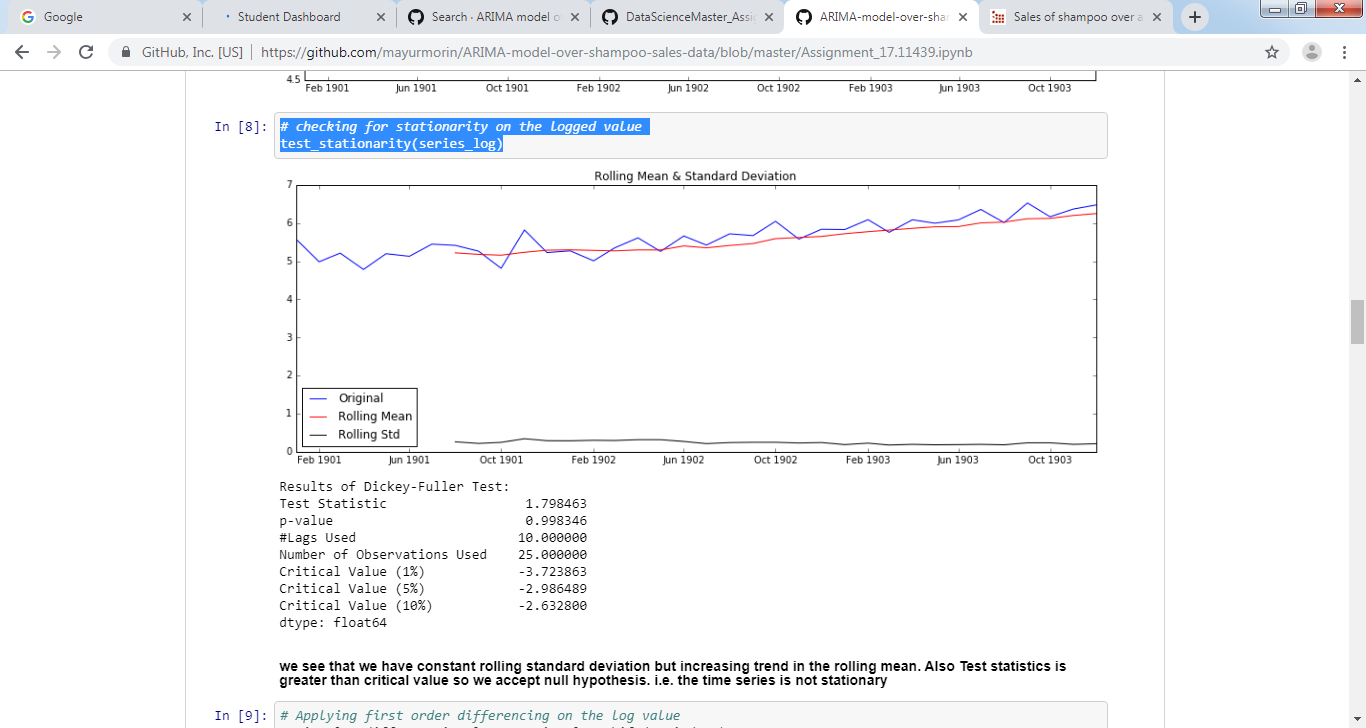






*# checking for stationarity on the logged value*

test\_stationarity(series\_log)



*# Applying first order differencing on the log value*

series\_log\_diff = series\_log - series\_log.shift(periods=1)

series\_log\_diff.head()

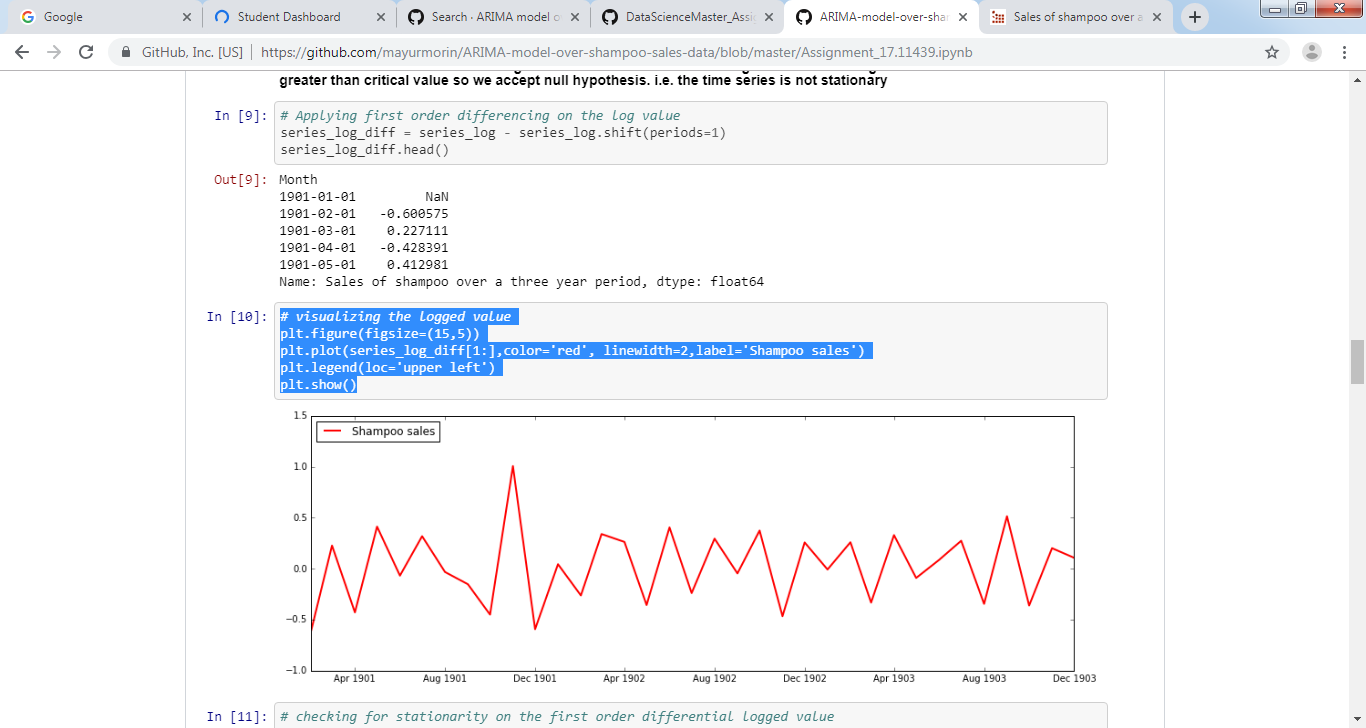
*# visualizing the logged value*

plt.figure(figsize=(15,5))

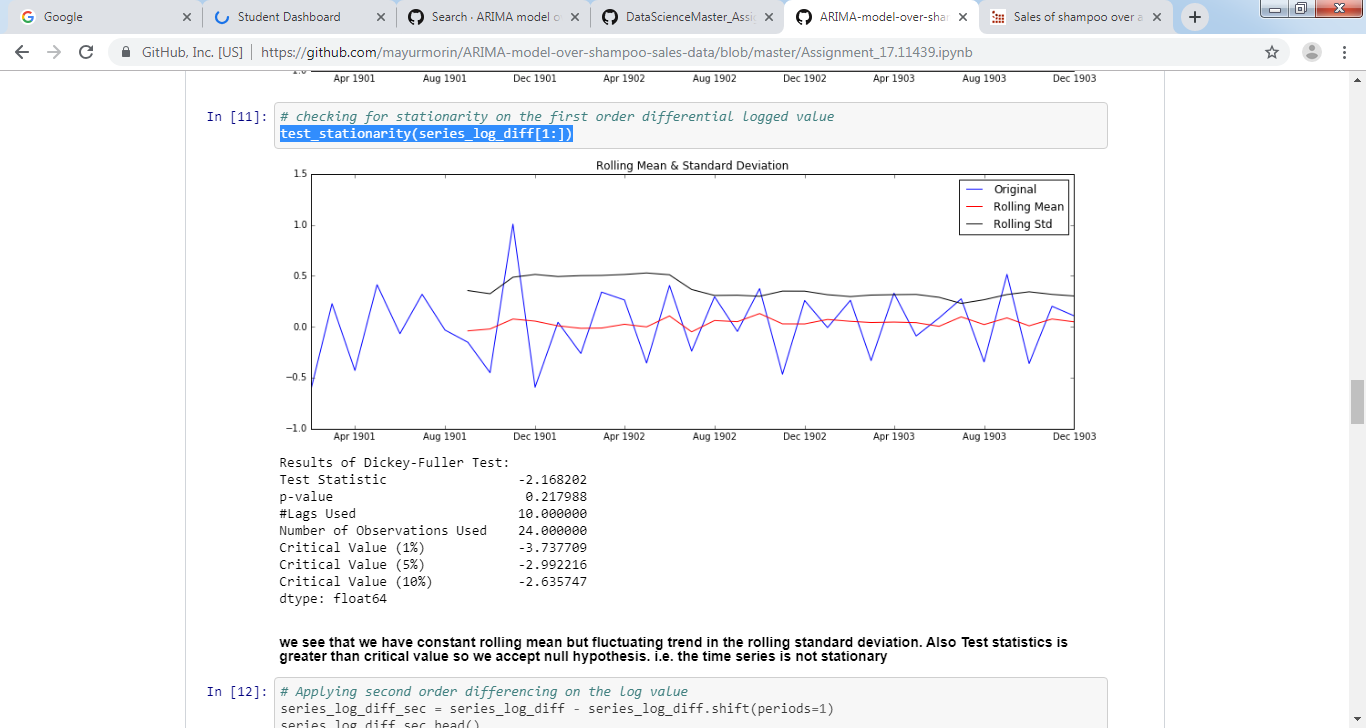
plt.plot(series\_log\_diff[1:],color='red', linewidth=2,label='Shampoo sales')

plt.legend(loc='upper left')

plt.show()



test\_stationarity(series\_log\_diff[1:])



*# Applying second order differencing on the log value*

series\_log\_diff\_sec = series\_log\_diff - series\_log\_diff.shift(periods=1)

series\_log\_diff\_sec.head()

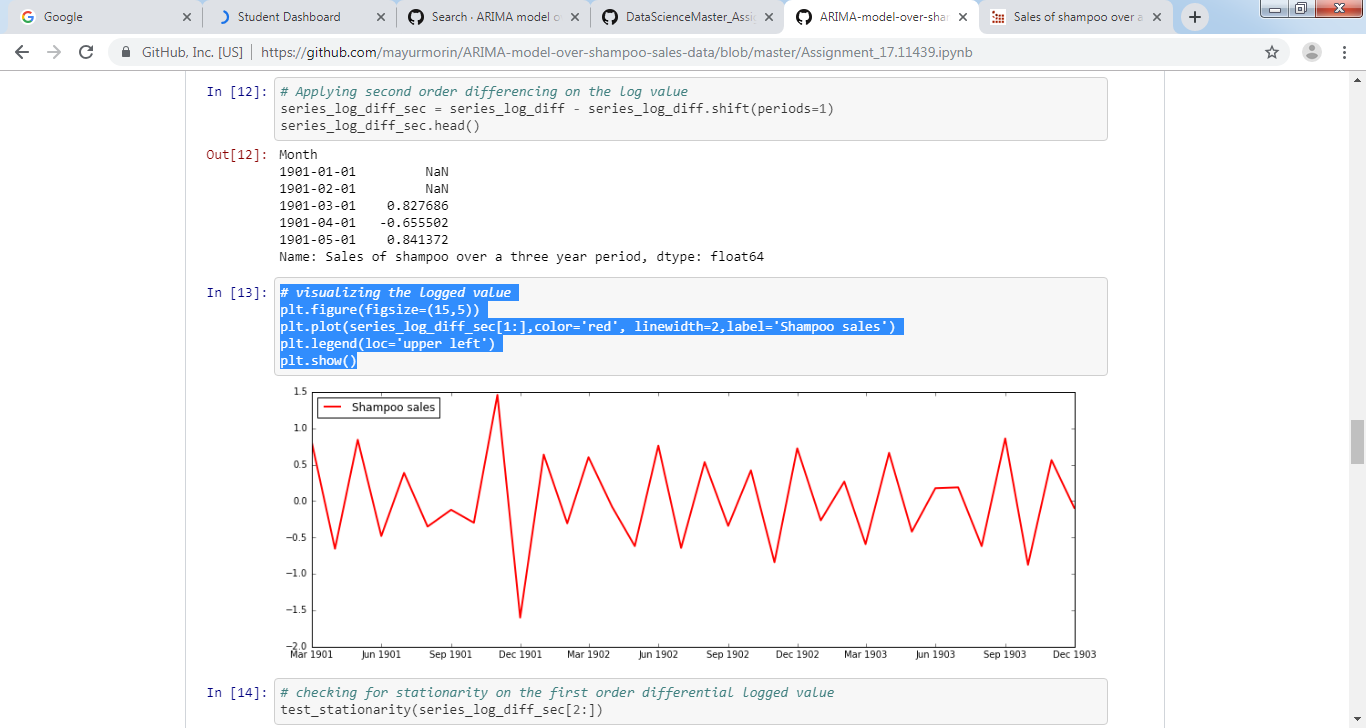
*# visualizing the logged value*

plt.figure(figsize=(15,5))

plt.plot(series\_log\_diff\_sec[1:],color='red', linewidth=2,label='Shampoo sales')

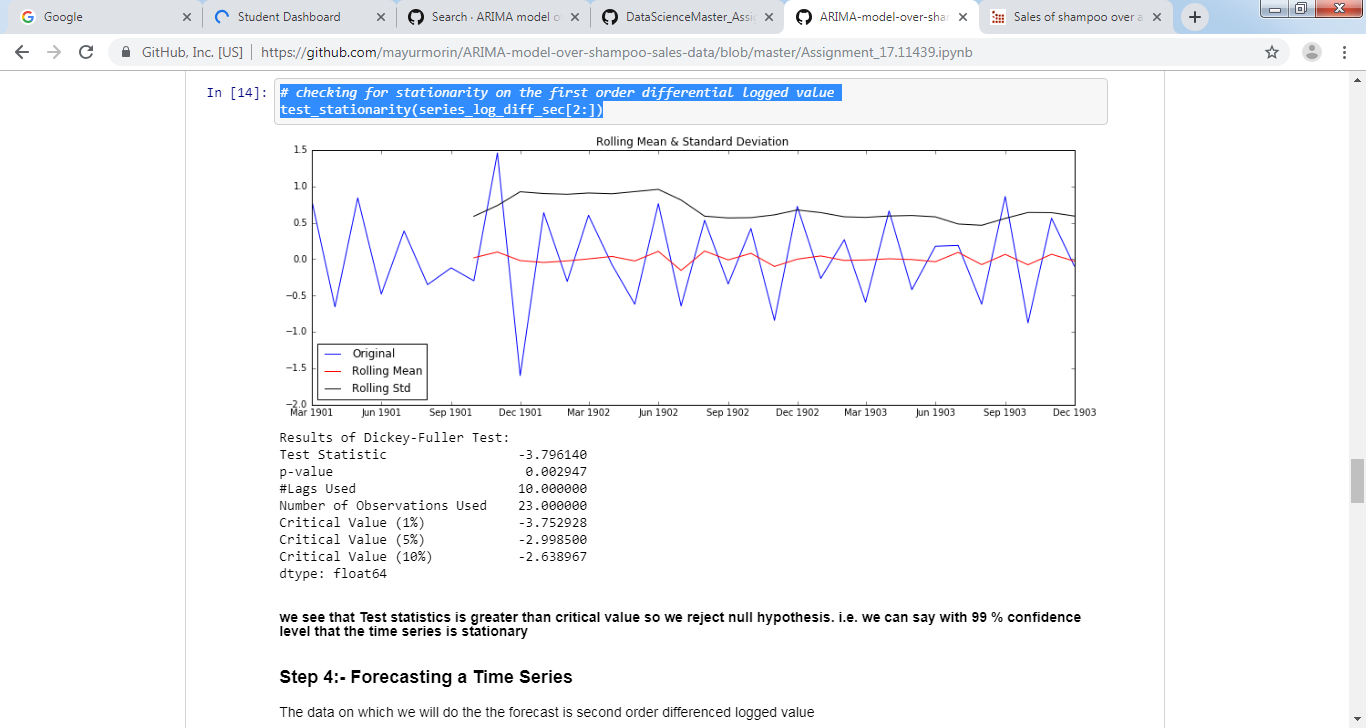
plt.legend(loc='upper left')

plt.show()



*# checking for stationarity on the first order differential logged value*

test\_stationarity(series\_log\_diff\_sec[2:])



series\_log\_diff\_sec.values[2:]

*#ACF and PACF plots:*

**import** **statsmodels.api** **as** **sm**

*# show plots in the notebook*

%matplotlib inline

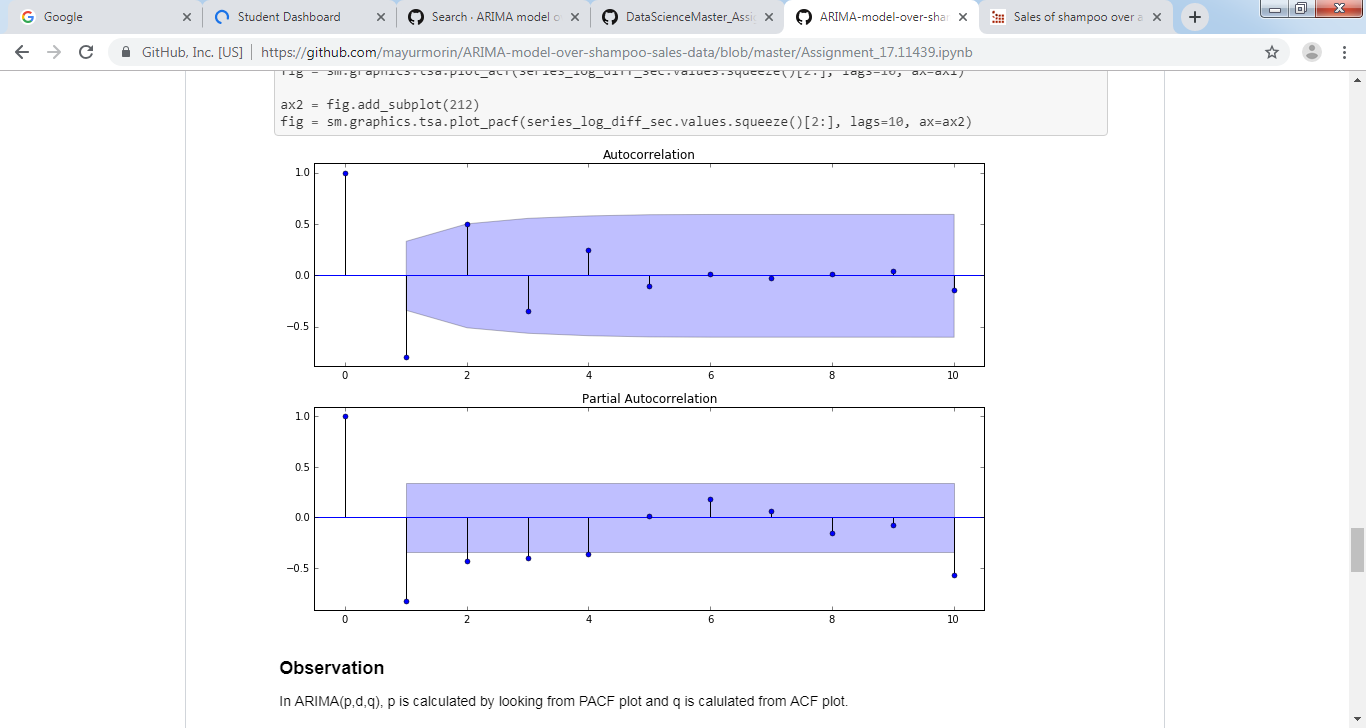
fig = plt.figure(figsize=(12,8))

ax1 = fig.add\_subplot(211)

fig = sm.graphics.tsa.plot\_acf(series\_log\_diff\_sec.values.squeeze()[2:], lags=10, ax=ax1)

ax2 = fig.add\_subplot(212)

fig = sm.graphics.tsa.plot\_pacf(series\_log\_diff\_sec.values.squeeze()[2:], lags=10, ax=ax2)



**import** **warnings**

warnings.filterwarnings('ignore')

*# for model ARIMA(4,2,0) i.e. AR Model*

model = ARIMA(series\_log, order=(4, 2, 0))

results\_ARIMA = model.fit(disp=-1)

plt.figure(figsize=(15,5))

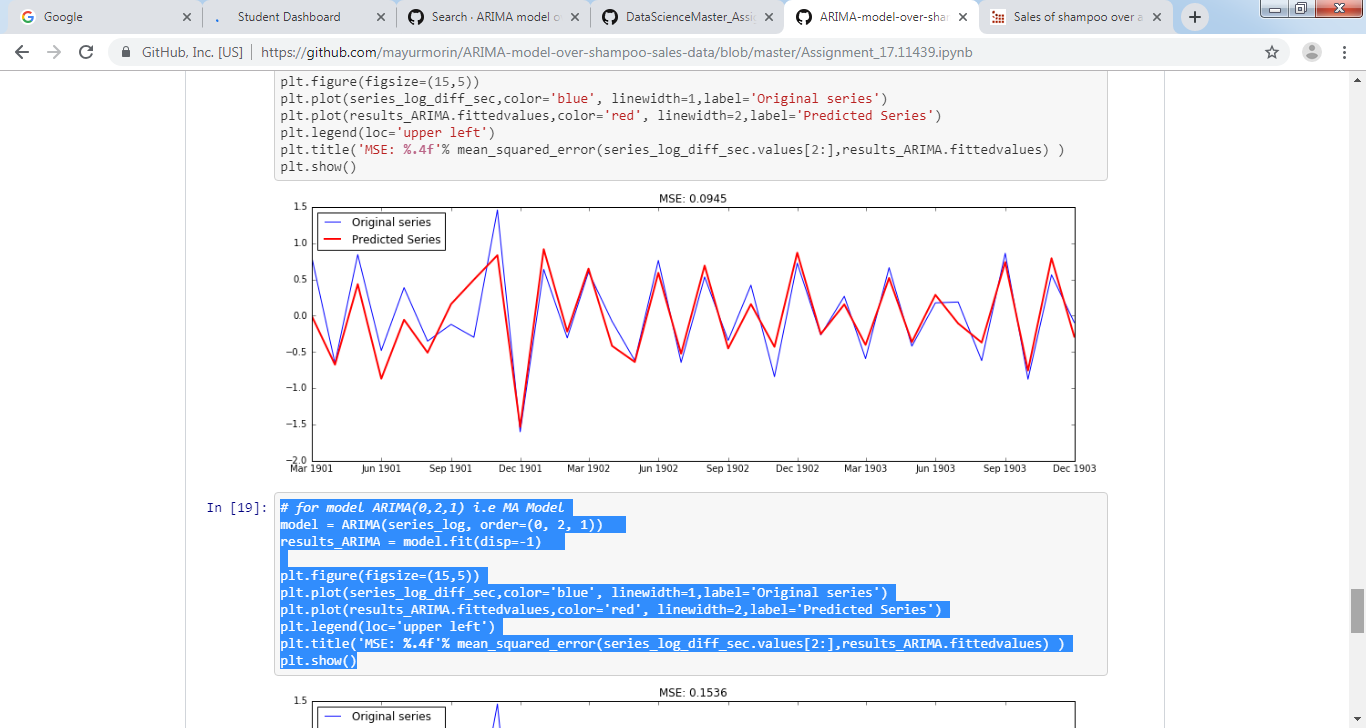
plt.plot(series\_log\_diff\_sec,color='blue', linewidth=1,label='Original series')

plt.plot(results\_ARIMA.fittedvalues,color='red', linewidth=2,label='Predicted Series')

plt.legend(loc='upper left')

plt.title('MSE: **%.4f**'% mean\_squared\_error(series\_log\_diff\_sec.values[2:],results\_ARIMA.fittedvalues) )

plt.show()



*# for model ARIMA(0,2,1) i.e MA Model*

model = ARIMA(series\_log, order=(0, 2, 1))

results\_ARIMA = model.fit(disp=-1)

plt.figure(figsize=(15,5))

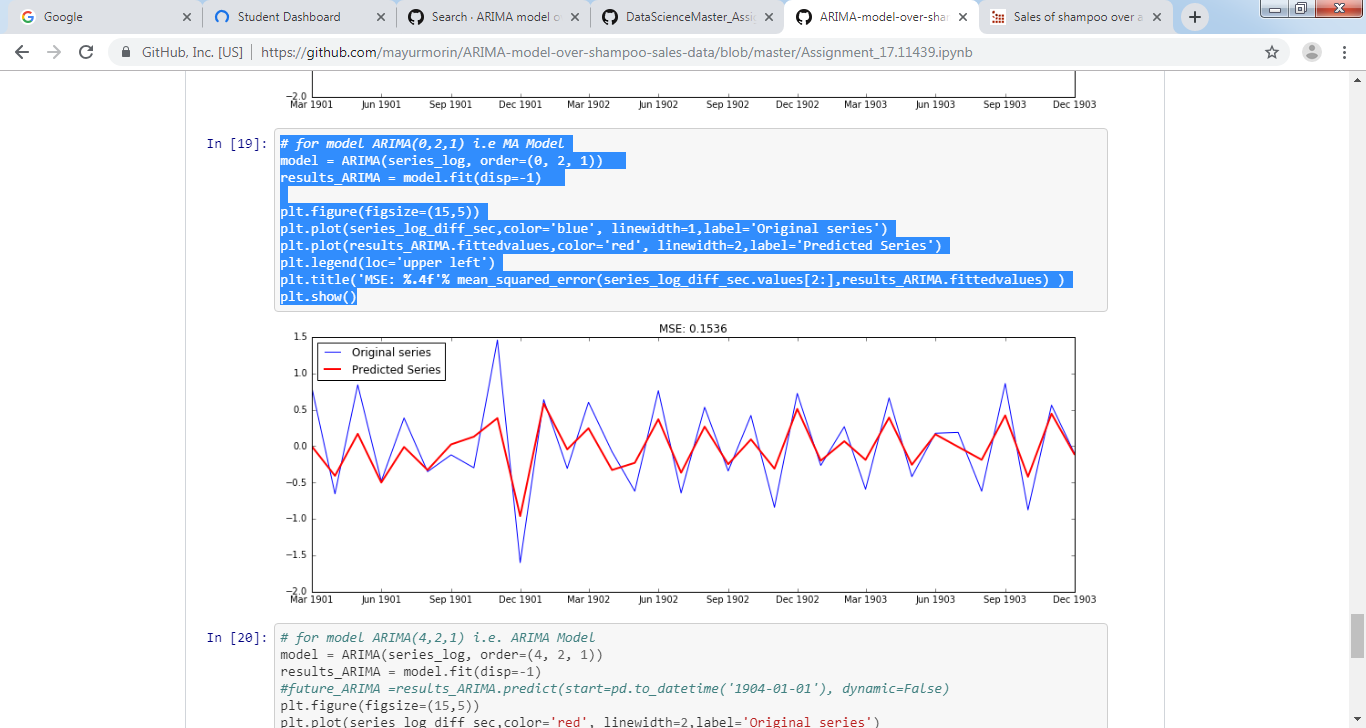
plt.plot(series\_log\_diff\_sec,color='blue', linewidth=1,label='Original series')

plt.plot(results\_ARIMA.fittedvalues,color='red', linewidth=2,label='Predicted Series')

plt.legend(loc='upper left')

plt.title('MSE: **%.4f**'% mean\_squared\_error(series\_log\_diff\_sec.values[2:],results\_ARIMA.fittedvalues) )

plt.show()



*# for model ARIMA(4,2,1) i.e. ARIMA Model*

model = ARIMA(series\_log, order=(4, 2, 1))

results\_ARIMA = model.fit(disp=-1)

*#future\_ARIMA =results\_ARIMA.predict(start=pd.to\_datetime('1904-01-01'), dynamic=False)*

plt.figure(figsize=(15,5))

plt.plot(series\_log\_diff\_sec,color='red', linewidth=2,label='Original series')

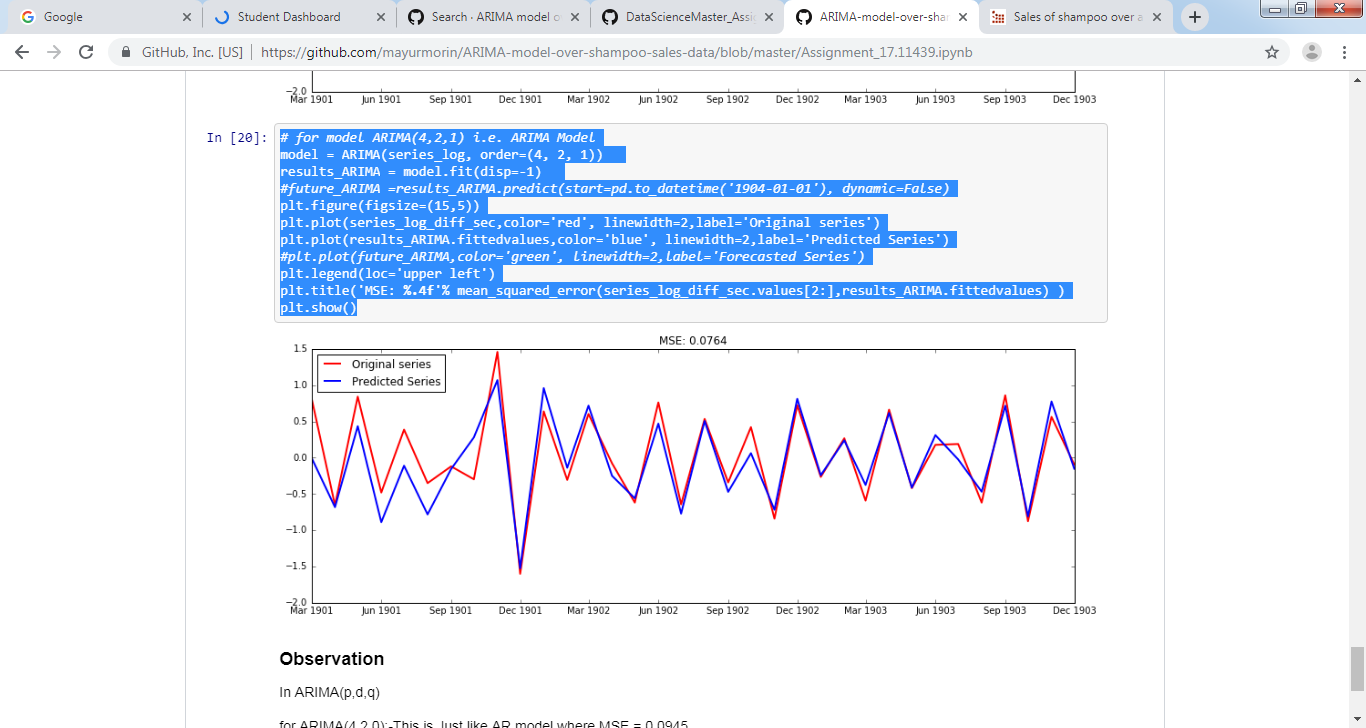
plt.plot(results\_ARIMA.fittedvalues,color='blue', linewidth=2,label='Predicted Series')

*#plt.plot(future\_ARIMA,color='green', linewidth=2,label='Forecasted Series')*

plt.legend(loc='upper left')

plt.title('MSE: **%.4f**'% mean\_squared\_error(series\_log\_diff\_sec.values[2:],results\_ARIMA.fittedvalues) )

plt.show()



In ARIMA(p,d,q)

for ARIMA(4,2,0):-This is Just like AR model where MSE = 0.0945

for ARIMA(0,2,1):-This is Just like MA model where MSE = 0.1536

for ARIMA(4,2,1):-This is Just like ARIMA model where MSE = 0.0764

**Hence, ARIMA(4,2,1) is best model which forecasts the time series very well.**