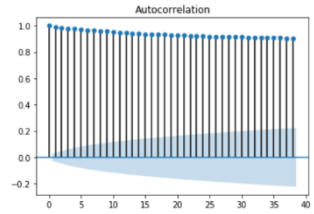
Data Analytics Assignment 6

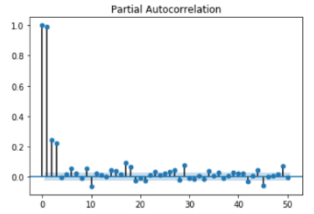
ARMA AND ARIMA

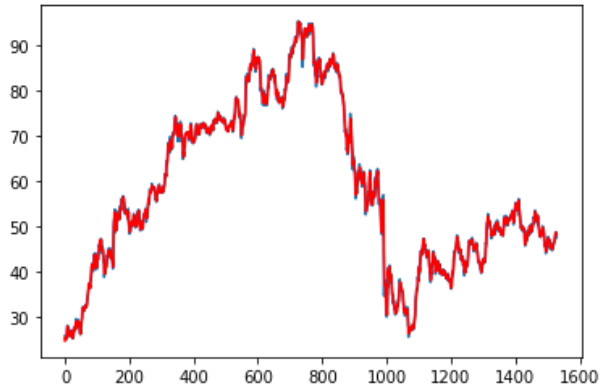
ARMA ACF AND PACF PLOT:

```
In [21]: from statsmodels.graphics.tsaplots import plot_acf
         plot_acf(exchangeRateSeries)
         plt.plot()
Out[21]: []
```




```
In [22]: from statsmodels.graphics.tsaplots import plot_pacf
         plot_pacf(exchangeRateSeries,lags=50)
         plt.plot()
Out[22]: []
```





ARMA CODE SNIPPET:

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np

def forecast_accuracy(forecast, actual):
    diff=[]
    for i in range(0,min(len(forecast),len(actual))):
        diff.append(forecast[i] - actual[i])
    mape = np.mean(np.abs(diff)/np.abs(actual))  # MAPE
    me = np.mean(diff)             # ME
    mae = np.mean(np.abs(diff))    # MAE
    rmse = np.mean([(diff[i])**2 for i in range(0,len(diff))])**.5  # RMSE
    print('MAPE :',mape,'ME :',me, 'MAE :',mae,'RMSE :',rmse)
#get data
def GetData(fileName):
    return read_csv(fileName, header=0, parse_dates=[0], index_col=0).values

#Function that calls ARIMA model to fit and forecast the data
def StartARIMAForecasting(Actual, P, D, Q):
        model = ARIMA(Actual, order=(P, D, Q))
        model_fit = model.fit(disp=0)
        prediction = model_fit.forecast()[0]
        return prediction

#Get exchange rates
ActualData = GetData('C:/Users/Navneet/Desktop/exchange.csv')
#Size of exchange rates
NumberOfElements = len(ActualData)

#Use 70% of data as training, rest 30% to Test model
TrainingSize = int(NumberOfElements * 0.7)
TrainingData = ActualData[0:TrainingSize]
TestData = ActualData[TrainingSize:NumberOfElements]

#new arrays to store actual and predictions
Actual = [x for x in TrainingData]
Predictions = list()


#in a for loop, predict values using ARIMA model
for timepoint in range(len(TestData)):
        ActualValue =  TestData[timepoint]
        #forcast value
        Prediction = StartARIMAForecasting(Actual, 3,1,0)
        print('Actual=%f, Predicted=%f' % (ActualValue, Prediction))
        #add it in the list
        Predictions.append(Prediction)
        Actual.append(ActualValue)

#Print MSE to see how good the model is
Error = mean_squared_error(TestData, Predictions)
print('Test Mean Squared Error (smaller the better fit): %.3f' % Error)
# plot
pyplot.plot(TestData)
pyplot.plot(Predictions, color='red')
pyplot.show()

forecast_accuracy(Predictions,Actual)
```

ARIMA CODE SNIPPET:

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

#get data
def GetData(fileName):
    return read_csv(fileName, header=0, parse_dates=[0], index_col=0).values

#Function that calls ARIMA model to fit and forecast the data
def StartARIMAForecasting(Actual, P, D, Q):
        model = ARIMA(Actual, order=(P, 0, Q))
        model_fit = model.fit(disp=0)
        prediction = model_fit.forecast()[0]
        return prediction

#Get exchange rates
ActualData = GetData('C:/Users/Navneet/Desktop/data.csv')
#Size of exchange rates
NumberOfElements = len(ActualData)

#Use 70% of data as training, rest 30% to Test model
TrainingSize = int(NumberOfElements * 0.7)
TrainingData = ActualData[0:TrainingSize]
TestData = ActualData[TrainingSize:NumberOfElements]

#new arrays to store actual and predictions
Actual = [x for x in TrainingData]
Predictions = list()


#in a for loop, predict values using ARIMA model
for timepoint in range(len(TestData)):
        ActualValue =  TestData[timepoint]
        #forcast value
        Prediction = StartARIMAForecasting(Actual, 3,1,0)
        print('Actual=%f, Predicted=%f' % (ActualValue, Prediction))
        #add it in the list
        Predictions.append(Prediction)
        Actual.append(ActualValue)

#Print MSE to see how good the model is
Error = mean_squared_error(TestData, Predictions)
#print('Test Mean Squared Error (smaller the better fit): %.3f' % Error)
# plot
pyplot.plot(TestData)
pyplot.plot(Predictions, color='red')
pyplot.show()
```

ARIMA OUTPUT: