Autoregression

You are given the **electricity power units (MegaWatts) consumed on a daily basis** by Himachal Pradesh as a csv file (`datasetA6_HP.csv`). To study the power consumption in the light of COVID19 (lockdown) data is recorded in the form of a time series for a period of 17 months beginning from 2nd Jan 2019 till 23rd May 2020. Rows are indexed with dates, first column represents the date and second column represent power consumed in Himachal Pradesh. Rows and columns put together, each data point in second column reflects the power consumed in Mega Units (MU) by the Himachal Pradesh (column) at the given date (row).

1. Autocorrelation line plot with lagged values:
   a. Create a line plot with x-axis as index of the day and y-axis as power consumed in mega units (MU).
   b. Generate another time sequence with one day lag to the given time sequence. Find the Pearson correlation (autocorrelation) coefficient between the generated one day lag time sequence and the given time sequence.
   c. Generate a scatter plot between given time sequence and one day lagged generated sequence in 1.b. What do you infer regarding correlation? Does it match with the computed correlation coefficient in 1.b?
   d. Generate multiple time sequences with different lag values (1 day, 2 days, 3 days upto 7 days). Compute the Pearson correlation (autocorrelation) coefficient between each of the generated time sequences and the given time sequence. Create a line plot between obtained correlation coefficients (on y-axis) and lagged values (on x-axis).
   e. Use python inbuilt function 'plot_acf' to generate the line plot which you manually coded in 1.d. Observe the trend in line plot with increase in lagged values.

2. Consider the last 250 days as test data. (No shuffling is done to preserve the time sequence information.) Given a dataframe with data points for $(t\text{-}1)^{th}$ timestamp and $(t)^{th}$ timestamp, problem statement is to predict $(t)^{th}$ datapoint given $(t\text{-}1)^{th}$ datapoint. The persistence algorithm outputs the same value as input while the expected value is corresponding $(t)^{th}$ datapoint from the original dataframe. This is known as the persistence model and is the simplest autoregression model. Compute the RMSE between predicted power consumed for test data and original values for test data.

3. A general autoregression model estimates the unknown data values as a linear combination of given lagged data values. For example, data value at $(t+1)$ instant, denoted by $x(t+1)$ can be estimated from its previous instance values, such as $x(t+1) = w0 + w1*x(t) + w2*x(t\text{-}1) + \ldots + wp*x(t\text{-}p+1)$. The coefficients $w0$, $w1$, … wp can be estimated while training the autoregression model on training dataset.
   a. Split the data into two parts for training and testing. Choose the first 250 days as training data and last remaining days as test data. Generate an autoregression (AR) model using `AutoReg()`. (See the code snippet below for this.) This function generates an AR model

with the specified training data and lagged values (given as its input). Use 5 lagged values as its input (*p*=5). Train/Fit the model onto the training dataset. Use the trained AR model to predict the values for the test dataset. Compute RMSE computed for test data

and compare it with RMSE obtained in the question 2. Generate a plot between the original test data time sequence and predicted test data time sequence.

b. Generate five AR models using `AutoReg()` function with lagged values as last 1, 5, 10, 15 and 25 days. Compute the RMSE between predicted and original data values. Please infer the changes in RMSE with changes in lagged values.

c. Compute the heuristic value for optimal number of lags up to the condition on autocorrelation such that *abs*(*AutoCorrelation*) > *2/sqrt*(*T*), where *T* is the number of observations in training data. Use it as input in `AutoReg()` function to predict the power consumed in test days and compute the RMSE value.

d. Compare the optimal number of lags (*p*) in parts 3.b and 3.c. Compare the RMSE values in parts 3.b and 3.c.

Functions/Code Snippets:

```
import statsmodels.api as sm from
statsmodels.tsa.ar_model import AutoReg

# load dataset
series = read_csv(‘name.csv’)

# split dataset
X = series.values train, test = X[1:len(X)-
250], X[len(X)-250:]

# train autoregression
model = AutoReg(train, lags=5) model_fit
= model.fit()
print('Coefficients: %s' % model_fit.params)

# make predictions predictions = model_fit.predict(start=len(train),
end=len(train)+len(test)-1, dynamic=False) for i in range(len(predictions)):
 print('predicted=%f, expected=%f' % (predictions[i], test[i]))
rmse   =   sqrt(mean_squared_error(test,   predictions))
print('Test RMSE: %.3f' % rmse)
```

Instructions:

- **Your python program(s) should be well commented. Comment section at the beginning of the program(s) should include your name, registration number and mobile number.**
- **The python program(s) should be in the file extension .py**
- **Report should be strictly in PDF form. Write the report in word or latex form and then convert to PDF form. Template for the report (in word and latex) is uploaded.**
- **First page of your report must include your name, registration number and mobile number. Use the template of the report given in the assignment.**
- **Upload your program(s) and report in a single zip file. Give the name as <roll_*number*>_Assignment6.zip. Example: *b20001*_Assignment6.zip**