Attribute Information:

1.date: Date in format dd/mm/yyyy

2.time: time in format hh:mm:ss

3.global\_active\_power: household global minute-averaged active power (in kilowatt)

4.global\_reactive\_power: household global minute-averaged reactive power (in kilowatt)

5.voltage: minute-averaged voltage (in volt)

6.global\_intensity: household global minute-averaged current intensity (in ampere)

7.sub\_metering\_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).

8.sub\_metering\_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.

9.sub\_metering\_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

### \* I will apply recurrent nueral network (LSTM) which is best suited for time-seriers and sequential problem. This approach is the best if we have large data.

### \* I will frame the supervised learning problem as predicting the Global\_active\_power at the current time (t) given the Global\_active\_power measurement and other features at the prior time step.

**Above I showed 7 input variables (input series) and the 1 output variable for 'Global\_active\_power' at the current time in hour (depending on resampling).**

# Model architecture

### 1) LSTM with 100 neurons in the first visible layer

### 3) dropout 20%

### 4) 1 neuron in the output layer for predicting Global\_active\_power.

### 5) The input shape will be 1 time step with 7 features.

### 6) I use the Mean Absolute Error (MAE) loss function and the efficient Adam version of stochastic gradient descent.

### 7) The model will be fit for 20 training epochs with a batch size of 70.

# Final remarks

### \* Here I have used the LSTM neural network which is now the state-of-the-art for sequencial problems.

### \* In order to reduce the computation time, and get some results quickly, I took the first year of data (resampled over hour) to train the model and the rest of data to test the model.

### \* I put together a very simple LSTM neural-network to show that one can obtain reasonable predictions. However numbers of rows is too high and as a result the computation is very time-consuming (even for the simple model in the above it took few mins to be run on 2.8 GHz Intel Core i7). The Best is to write the last part of code using Spark (MLlib) running on GPU.

### \* Moreover, the neural-network architecture that I have designed is a toy model. It can be easily improved by adding CNN and dropout layers. The CNN is useful here since there are correlations in data (CNN layer is a good way to probe the local structure of data).