Practice: Building a Long Short-Term Memory Recurrent Neural Network Using the Python API

The National Centers for Environmental Information (NCEI) provides public access to environmental data archives. The data set spokane contains hourly weather information for the city of Spokane, Washington from 2008 to 2017. There are approximately 90,000 observations. The goal is to use a recurrent neural network to forecast the average hourly temperature in Spokane, Washington.

Name	Model Role	Measurement Level	Description
LST_DATE	Date	Date	Date of each observation (YYYYMMDD)
LST_TIME	Input	Nominal	Hour of the observation (0-2300 by 100)
T_HR_AVG	Target	Interval	Average temperature in degrees C during the hour
P_CALC	Input	Interval	Total amount of precipitation in mm during the hour

1. From the Jupyter Lab Home directory, select the plus symbol to open a new page and then select Python under Notebook to create a new notebook.
2. Load the os, sys, SWAT, pandas, and matplotlib packages. Use the %MATPLOTLIB inline option to print graphics in the notebook.
3. Connect to CAS and create a connection object called **conn**.
4. Load the **spokane.csv** file on the server and name it **spokane**. Find the dimension of the table and print the first few observations.
5. Using a DATA step, subset the first year of data (2008). Use the runCode action from the dataStep action set.
6. Download the first year of data to the client and plot the time series.
7. Use a DATA step to create five, hour, lag variables for **t_hr_avg**, **lst_time**, and **p_calc**. Be sure to use the single="Yes" option to restrict execution to a single thread.
8. Use a DATA step to remove missing values and values less than -30 degrees Celsius.
9. Use a DATA step to partition the data into training (before 2015), validation (2016 and 2017), and test (2017) data sets.
10. Use the deepLearn action set to build a long short-term memory recurrent neural network with two hidden layers and 15 neurons in each layer.
11. Create variables for the target (**t_hr_avg**) and inputs (**lst_time**, **p_calc**, and the **lag** variables) to pass into subsequent CAS actions.
12. Use the dlTrain action to train the LSTM RNN model. Save the trained weights and train for 50 iterations. Be sure to use a timeStep of five units.
13. Use the trained weights to score the test data set with the dlScore action.
14. Download the scoring information to the client and order the data frame by the **lst_date** and **lst_time** variables.
15. Print the average absolute error for the forecast period and plot the first 1000 hours of the original data and the model forecast to compare the fit.
16. End the CAS session.