Energy Demand Forecasting

Challenge Question for Code Jam @ McGill 2013

Overview

According to the International Energy Agency, we need to half our projected coal use, and triple our projected use of renewables by 2035 in order to avoid dangerous levels of global warming….

Fortunately, you have just been hired to develop cutting edge smart grid technology and help find a solution. Your team is looking to make advances in energy demand forecasting, an area of growing importance for several reasons. On a larger scale, forecasting is important because it helps utilities control the grid more efficiently, and facilitates the integration of new renewable sources. On a smaller scale, it allows engineers and building operators to reduce power demands at critical times, ultimately reducing their environmental impact, and saving money.

You have been given over two years of historic power demand and weather data for a large institution (McGill University), as well as access to a live stream of power demand and weather information. You have been asked to use these as a prototyping ground to:

Devise an application that can provide accurate, real-time forecasts of McGill's electricity consumption up to 12 hours in advance.

Your success will be judged on based on **three primary criteria**:

- The accuracy of the predictions generated,
- The application's ability to connect to a live data stream and generate real-time forecasts,
- The clarity with which it presents the information.

Specifications

Your team will be provided with a large data-set similar to the one below. It will include over two years of energy data (at 15 minute intervals) and weather data (at 1 hour intervals) from McGill. This data set is called **data_set.csv.** Its contents look like:

Date	Radiation	Humidity	Temperature	Wind Speed	Power Demand
2011-06-20T09:00-04:00	271	0.67	18.6	4	20670.2
2011-06-20T09:15-04:00					20908.976
2011-06-20T09:30-04:00					21097.521
2011-06-20T09:45-04:00					21177.393
2011-06-20T10:00-04:00	415	0.62	20.1	2	21302.315
2011-06-20T10:15-04:00					21462.118
2011-06-20T10:30-04:00					21688.764
2011-06-20T10:45-04:00					21803.499

2011-06-20T11:00-04:00	491	0.59	20.4	6	21701.781
2011-06-20T11:15-04:00					21772.078
2011-06-20T11:30-04:00					21795.055
2011-06-20T11:45-04:00					21734.4889999999
2011-06-20T12:00-04:00	528	0.38	22	9	21649.6969999999
… (2+ years worth)					

Using this historic information, you must design a web application **capable of predicting McGill's power demand for a 16 hour window into the future** (in 15 minute intervals). More specifically, the application should have have two URLs: one providing a machine connection, the other linking to a graphical user interface.

1- Machine Interface

Requirements

You must provide a URL that can respond to a POST request containing input csv data, and output predictions. Specifically, your app will be tested using the following cURL command:

curl -F file=@sample_input.csv http://your_site.com/some_extension > sample_output.txt

Given the provided file 'sample_input.csv', your app should return content in the format found in the provided 'sample_output.txt'.

Examining the contents of 'sample_input.csv', we find 97 lines of text in almost the same format as the data set.

- Line 1 = title line
- Line 2-33 = lines of historic data
- Line 34-97 = lines of **dates and weather data only** (with power demand set to 'null') corresponding to "the future".

Example:

Date	Radiation	Humidity	Temperature	Wind Speed	Power Demand
…32 rows (total) of historic					
2013-11-09T06:45-05:00					14333.461
2013-11-09T07:00-05:00	-141	0.64	1	2	14595.815
2013-11-09T07:15-05:00					14614.983
2013-11-09T07:30-05:00					14636.123
2013-11-09T07:45-05:00					14414.236
2013-11-09T08:00-05:00	15	0.64	1	4	null

2013-11-09T08:15-05:00					null
2013-11-09T08:30-05:00					null
2013-11-09T08:45-05:00					null
2013-11-09T09:00-05:00	120	0.6	2	6	null
2013-11-09T09:15-05:00					null
…64 rows (total) of “future”					

Your app should respond with the following content.

Date	Power Demand
2013-11-09T08:00-05:00	14369.872
2013-11-09T08:15-05:00	14425.662
2013-11-09T08:30-05:00	14467.903
2013-11-09T08:45-05:00	14563.201
2013-11-09T09:00-05:00	14780.018
2013-11-09T09:15-05:00	14758.391
2013-11-09T09:30-05:00	14759.164
2013-11-09T09:45-05:00	14866.025
… 64 rows (total) of forecasts	

Where the first date** entry in the output (in bold) should correspond exactly with the date of the first power demand entry to be ‘null’ in the input (also in bold). Therefore all entries in the power demand column are a forecast (corresponding to the ‘null’ entries in the posted file).

Please see ‘sample_input.csv’, and ‘sample_output.txt’ for exact format specifications.

****Note that **dates must be in the ISO 8601 format,** as specified by the sample files.

Assessment

The application’s performance will be judged via a square percent error function. We will feed your application several different input files from a dataset you do not have access to (including data collected from after the competition has ended), and the output forecasts will be assessed.

Each forecast received will be compared against the actual realized values, the difference will then be normalized, squared, summed. Therefore your objective is to **minimize** the below expression.

$$\left(\sum_{i=1}^{i=64}((predictions_i - actualValue_i) / actualValue_i))^2\right) / N$$

You will be judged based on the application’s mean performance over series of several input files.

2- User Interface (Dashboard)

Requirements

Your application should also connect to Pulse Energy's API (http://developer.pulseenergy.com/), query the necessary data, and using the forecasting engine you designed in part 1, display a realtime energy forecast. This should be visible at a URL that you specify.

Exactly how you choose to display the information is up to you. However, the objective is to design a "dashboard" that clearly displays all relevant information, including but not limited to your sixteen hour electricity demand forecast.

Assessment

Points will be awarded for:

- Demonstrating a successful connection to the API, with the forecast updating every hour.
- The clarity and relevance of the information is presented. (Hint - what other information might someone reading an electricity forecast want to know? e.g. statistical information about prediction accuracy)
- Creativity and style.

Every team will be given an API key, as well as further instructions on how to use the Pulse API.

BONUS - Confidence intervals

Since forecasts are probabilistic in nature, the forecasted value is really the mean of a distribution. So, for teams finishing the challenge early and wishing to set themselves apart, we propose the following:

Construct a second machine interface, (with a third and separate URL), that handles the exact same input file format. However, this time the output content should have *three* columns (but otherwise but the same format). The latter two columns should indicate the lower and upper limits of the 75% confidence interval of the forecast, i.e. **75% of the time the actual value should fall within this interval.**

Date	Lower 75% Conf	Upper 75% Conf
2013-11-09T08:00-05:00	13369.872	15369.872
2013-11-09T08:15-05:00	13425.662	15425.662
2013-11-09T08:30-05:00	13467.903	15467.903
2013-11-09T08:45-05:00	13563.201	15563.201
2013-11-09T09:00-05:00	13780.018	15780.018
2013-11-09T09:15-05:00	13758.391	15758.391
2013-11-09T09:30-05:00	13759.164	15759.164
2013-11-09T09:45-05:00	13866.025	15866.025
… 64 rows (total) of forecasts		

Its accuracy will be assessed with the same test files as was the original machine interface, and scoring will be determined as follows.

```
sum = 0
for each row of forecast
        if actual value is within forecast confidence interval
                sum += 0.25
        else
                sum -= 0.75
sum = sum / number of rows tested
```

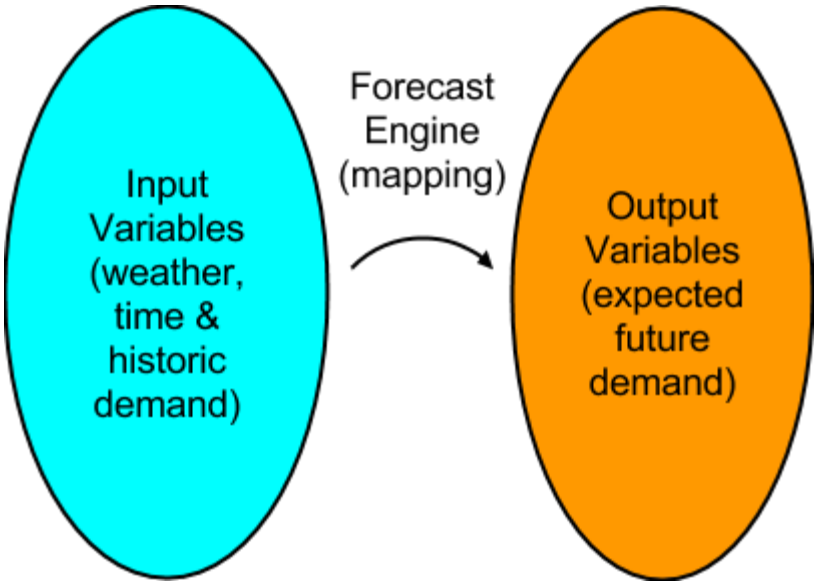
The sum which is closest to 0 is best (i.e. you want to minimize the *absolute value* of the sum)**.** Should two teams perform similarly in both other aspects, this bonus will be used to determine the winner.

Strategies & Basic Theory

The core objective here is to design an accurate forecasting engine. Recalling the input file format:

Date	Radiation	Humidity	Temperature	Wind Speed	Power Demand
…32 rows (total) of historic					
2013-11-09T06:45-05:00					14333.461
2013-11-09T07:00-05:00	-141	0.64	1	2	14595.815
2013-11-09T07:15-05:00					14614.983
2013-11-09T07:30-05:00					14636.123
2013-11-09T07:45-05:00					14414.236
2013-11-09T08:00-05:00	15	0.64	1	4	**null**
2013-11-09T08:15-05:00					**null**
2013-11-09T08:30-05:00					**null**
2013-11-09T08:45-05:00					**null**
2013-11-09T09:00-05:00	120	0.6	2	6	**null**
2013-11-09T09:15-05:00					**null**
…64 rows (total) of future					

This forecast engine is essentially a mapping: taking as input some combination of the variables highlighted in blue, and returning the expected values of the variables in **bold and highlighted in orange**.



Rather than trying to develop and test a deterministic model or an explicit relationship between these variables, it is advised that you take a **statistical learning approach**. Many statistical learning models exist, as do algorithms that will "tune" the parameters of these models when given a set of data to train on.

A couple more hints:

- The team you are working with is looking to apply *existing* **supervised machine learning** techniques to approach the task. (Hint: Several well-known approaches have extensive libraries with documentation online, available in a variety of languages.)
- They plan to **divide the historic data into two sets**, a **training** set (from which the application can learn), and a **test** set, on which to test the accuracy of the forecasts. They intend to ensure that each set is representative of a wide range of conditions.
- They plan to **interpolate the weather data** before training their application.
- They expect that **not only weather but also time** of day, day of the week, and week of the year will play an important role in the forecasting.
- However, they do not expect all the variables to play an equally important role, and think that **including all possible input variables may even reduce performance**.

To be submitted

- URL of Machine interface
- URL of User interface (Dashboard)
- URL of Bonus (if applicable)
- Openshift account credentials

GOOD LUCK!