PREDICTING WIND ENERGY PRODUCTION WITH SCIKIT-LEARN

• INTRODUCTION

Nowadays, electricity networks of advanced countries rely more and more in non-operable renewable energy sources, mainly wind and solar. However, in order to integrate energy sources in the electricity network, it is required that the amount of energy to be generated to be forecasted 24 hours in advance, so that energy plants connected to the electricity network can be planned and prepared to meet supply and demand during the next day (For more details, check "Electricity Market" at Wikipedia).

This is not an issue for traditional energy sources (gas, oil, hydropower, …) because they can be generated at will (by burning more gas, for example). But solar and wind energies are not under the control of the energy operator (i.e. they are non-operable), because they depend on the weather. Therefore, they must be forecasted with high accuracy. This can be achieved to some extent by accurate weather forecasts. The *Global Forecast System* (GFS, USA) and the *European Centre for Medium-Range Weather Forecasts* (ECMWF) are two of the most important Numerical Weather Prediction models (NWP) for this purpose.

Yet, although NWP's are very good at predicting accurately variables like "100-meter U wind component", related to wind speed, the relation between those variables and the electricity actually produced is not straightforward. Machine Learning models can be used for this task.

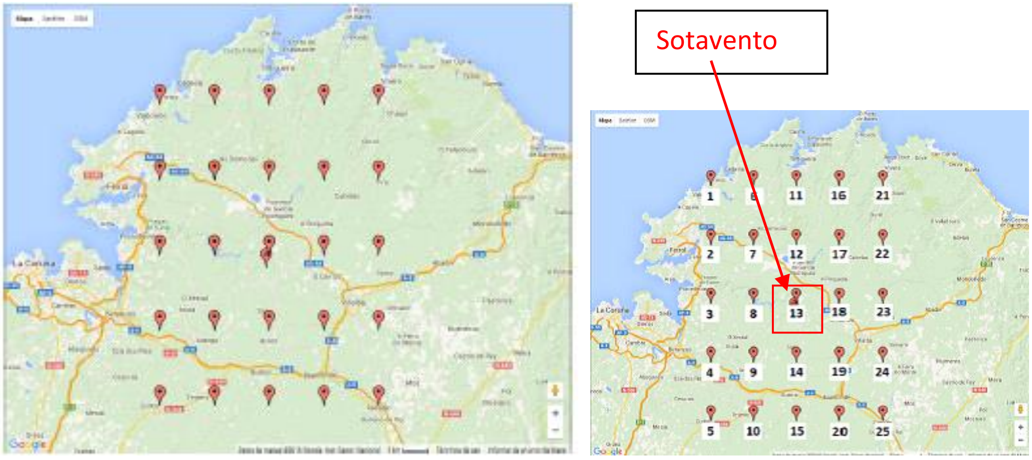
In particular, we are going to use meteorological variables forecasted by ECMWF (http://www.ecmwf.int/) as input attributes to a machine learning model that is able to estimate how much energy is going to be produced at the Sotavento experimental wind farm (http://www.sotaventogalicia.com/en).

Sotavento wind farm.

More concretely, we intend to train a machine learning model *f*, so that:

- Given the 00:00am ECMWF forecast for variables $A_{6:00}$, $B_{6:00}$, $C_{6:00}$, … at 6:00 am (i.e. six hours in advance)
- $f(A_{6:00}, B_{6:00}, C_{6:00}, …)$ = electricity generated at Sotavento at 6:00

We will assume that we are not experts on wind energy generation (not too far away from the truth, actually). This means we are not sure which meteorological variables are the most relevant, so we will use many of them, and let the machine learning models and attribute selection algorithms select the relevant ones. Specifically, 22 variables will be used. Some of them are clearly related to wind energy production (like “100 metre U wind component”), others not so clearly (“Leaf area index, high vegetation”). Also, it is common practice to use the value of those variables, not just at the location of interest (Sotavento in this case), but at points in a grid around Sotavento. A 5x5 grid will be used in this case.



5x5 grid around Sotavento.

Therefore, each meteorological variable has been instantiated at 25 different locations (location 13 is actually Sotavento). That is why, for instance, attribute iews appears 25 times in the dataset (*iews.1, iews.2, …, iews.13, …, iews.25*). Therefore, the dataset contains 22*25 = 550 input attributes.

• WHAT TO DO:

- General issues:
  a. Follow the preliminaries of the notebook. In that section, data is going to be read, missing values are going to be put at random, and then the file is going to be saved again on file ‘wind_pickle_with_nan.pickle’. This is the file that you will use for the remaining of the assignment.
  b. Historical data is available both from ECMWF (for the meteorological variables) and Sotavento (for energy production) from 2005 to 2010. The dataset has many columns. **Energy** is the response variable and must be kept. **Steps, month, day, hour, year should be removed, they cannot be used for training the models**. The remaining variables are the input attributes. They are defined for the 25 locations in the grid.
  c. We are going to use train/test for model evaluation (outer) and train/validation for hyper-parameter tuning (inner), as follows:
    i. Train partition: the first two years of data. Given that there are 6 years worth of data, we will use the first 2/6 of the instances for training.
    ii. Validation partition: the second two years of data.
    iii. Test partition: the remaining data
  d. Convert the training and test sets from Pandas dataframes to Numpy matrices, so that they can be used by scikit-learn.
  e. Please, note that in this occasion, we are not going to use cross-validation for hyper-parameter tuning, but train/validation. An array *tr_val_partition* must be defined with -1 if the instance is to be used for training, and 0 for validation. For instance, ***tr_val_partition* = PredefinedSplit([-1, -1, 0, 0])** means that the first two instances will be used for training

and the last two, for validation. Then, this can be used in, for instance, BayesSearch, as follows:

```
clf = BayesSearchCV(neighbors.KNeighborsRegressor(), param_grid,
scoring='neg_mean_absolute_error',  cv= tr_val_partition, verbose=1)
```

- **(1) Model selection and hyper-parameter tuning (1.25 points):**
    a. Here, please train KNN, Random Forest, and Gradient Boosting models, with and without hyper-parameter tuning. Both BayesSearch and Optuna can be used (but Optuna will be graded better). If you use advanced implementations of GradientBoosting such as XGboost or LightBoost, your work will get higher grades. Compare all of them using the test partition, so that we can conclude what is the best method. Also, compare results with those of trivial (dummy) models.
- **(2) Attribute selection (1.25 points) You have to answer the following two questions:**
    a. Are all 550 input attributes actually necessary in order to get a good model? Is it possible to have an accurate model that uses fewer than 550 variables? How many?
    b. Is it enough to use only the attributes for the actual Sotavento location? (13th location in the grid)

    In order to answer these questions, you should read the "Attribute Selection" notebook, and understand the main ideas about *SelectKBest* and *Pipeline*. Use *SelectKBest* and *Pipeline* (and whatever else you need) in order to find a subset of attributes that allows to build an accurate Decision Tree model. Use the best model of the previous section, but if it is too slow, use Decision Trees (they are faster). Use the test partition in order to compare different models.

3. WHAT TO HAND IN: All you need to hand in is your ipython notebook. But please, use some of the cells to make comments about what you are doing, your results, and the conclusions of your results. A good report, with tables and conclusions will be graded better. 0.25 points / 0.25 points are reserved in the first and second sections for extra effort beyond a straightforward answer, original student work, etc.

If your group has two members, please write your names at the beginning of the notebook.

ATTRIBUTE NAMES

2 metre temperature

10 metre U wind component

10 metre V wind component

100 metre U wind component

100 metre V wind component

Convective available potential energy

Forecast logarithm of surface roughness for heat

Forecast surface roughness

Instantaneous eastward turbulent surface stress

Instantaneous northward turbulent surface

Leaf area index, high vegetation

Leaf area index, low vegetation

Neutral wind at 10 m u-component

Neutral wind at 10 m v-component

Soil temperature level 1

Soil temperature level 2

Soil temperature level 3

Soil temperature level 4

Surface pressure

Vertical integral of temperatura

Vertical integral of divergence of kinetic energy

Vertical integral of water vapour