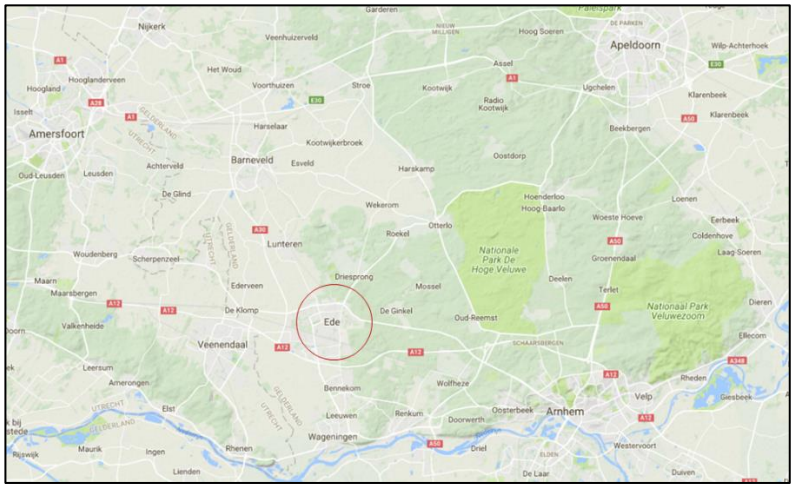
Exercise

A geo-exercise with regression algorithms

I.Garcia-Martí
July 2019

Context

Since 2006, a group of volunteers are sampling tick dynamics on a monthly basis in 17 locations of the Netherlands. Ticks are the main vector of Borrelia Burgdorferi bacteria, which is the main agent that causes the Lyme disease in humans. Every year in the Netherlands, 25.000 people are diagnosed with this disease after a tick bite, which often occurs while carrying out leisure activities in forests (e.g. hiking, picnicking, camping). In this block of exercises, we provide the time-series of one of this 17 locations, near the city of Ede. You can see the study area below, marked with a circle.



The sampling site in Ede is interesting from an ecological point of view, because it is close to the Veluwe national park, which is a natural habitat for ticks and a popular place for recreational purposes. The sampling site is composed of two transects of forest which are sampled by trained volunteers at the beginning of each month. You will use the data of the two transects (i.e. Ede_1 and Ede_2) to train ensemble models, and get the feature importance and the R2 score. The novelty here is that you will learn to test the model in the geographical space to visualize the results over a map.

We have enriched each of the observations in both transects with 7 weather variables, which are thought that can influence the activity of ticks. The value of these variables are from the same day of the sampling. The weather variables are: minimum and maximum temperature, precipitation, evapotranspiration, relative humidity, vapor pressure deficit and saturation deficit.

Note: Download the three required CSV files to do this block of exercises (i.e. *Tick_Dynamics_Ede_1*, *Tick_Dynamics_Ede_2*, and *NL_Weather_15_6_14*). Keep in mind that for these exercises we are not exhaustively giving instructions on how to proceed with the modelling. We trust that you are independent students capable of putting all pieces together! However, if you are stuck, do not hesitate in contacting the lecturer.

Modelling tick dynamics

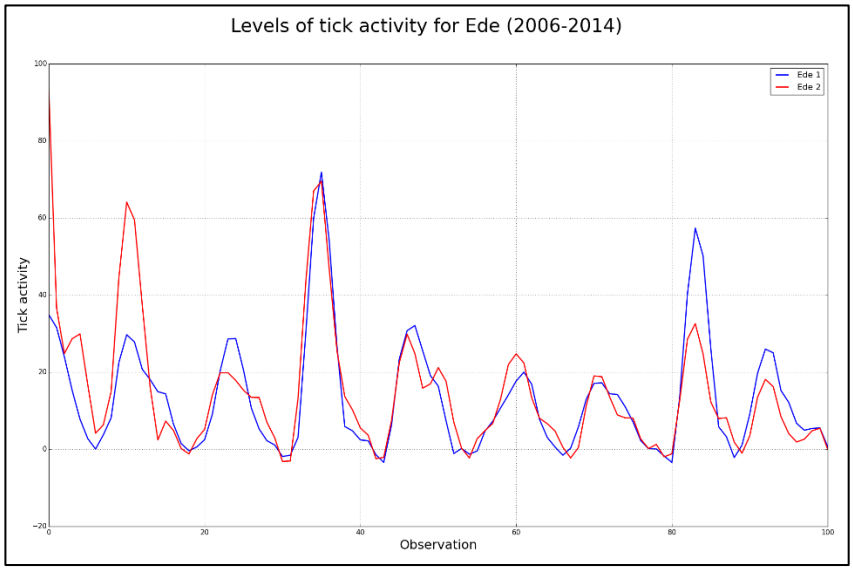
Exercise 1: Exploratory data analysis 1

Load into two numpy arrays the datasets for Ede (i.e. Tick_Dynamics_Ede_X.csv) using the parameter “usecols=range(3, 11)” to ignore the metadata columns corresponding to a row id, the name of the transect and the sampling date. You will use Ede_1 to train your models and Ede_2 to test your models. Extract the target variable of both files, which is stored in the first column of the datasets (i.e. that is index 0) and create a basic plot to explore both tick activity signals.

Hint: to plot two signals in the same plot, you can use the following code.

```
xlinspace = np.linspace(0, len(ede1)-1, len(ede1))
plt.plot(xlinspace, ede1, "b-", label="Ede 1", linewidth=2)
plt.plot(xlinspace, ede2, "r-", label="Ede 2", linewidth=2)
```

You should see something like:



As you can see, the signal of both transects is quite well-behaved, except at the extremes of the time-series. Think about these peaks: will they be a consequence of weather conditions?

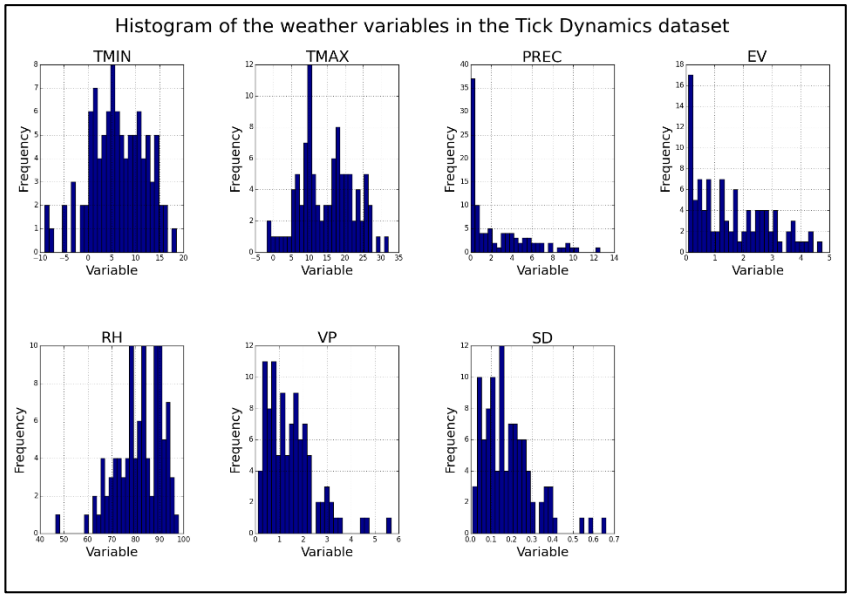
Exercise 2: Exploratory Data Analysis 2

A good practice when you receive a dataset is conducting an exploratory data analysis of your predictor values. A classic approach to do so is plotting a histogram per each individual predictor. This provides a quick overview of the data without printing it on the console.

Complete the following code in your Python script to draw a figure with 7 subplots, each one depicting a histogram of the weather variables captured in the tick dynamics dataset.

```
labels = ["TMIN", "TMAX", "PREC", "EV", "RH", "VP", "SD"]
plt.suptitle("Histogram of the weather variables”)
plt.subplots_adjust(hspace=0.5, wspace=0.5)
for i in range(1, 8):
     # Store in a variable called “curr” the i-th colum
     plt.subplot(2, 4, i)
     # Call the function plt.hist() with “curr” and 25 bins
     # Add proper labels to the subplot
     # Add grid lines to the subplot
     # Call the plt.title() function to add the (i – 1) label
# Show the chart
```

You should get as an output something like:



Make sure that the weather variables fall within reasonable ranges for the phenomenon measured.

Exercise 3: Modelling with ensemble algorithms

Import from the scikit-learn library the modules RandomForestRegressor and GradientBoostingRegressor. Create a "rf" object and a "gb" object and fine tune its parameters as done in the two first blocks of this practical session. Use the observations belonging to "Ede_1" to train each model and the ones in "Ede_2" to test de model. Calculate the R2 score to see the goodness of fitting of each model. Which model is the one with a higher R2? Now print the feature importance and identify which are the 3 most important features for each of the models. Do the models agree in the selected features?

Exercise 4: Testing the model at the country level

In this exercise you will apply the trained Random Forest model to all forested areas in the Netherlands for a particular date (i.e. 15th June 2014). To do so, we provide a new testing dataset that will replace your "xtest" variable. This testing dataset comes in CSV format and contains 3,541 rows and 10 columns. The first three columns contain a row id, and the pixel coordinates in a country raster. The columns contain the value of the weather features stated in the "Context" section, feel free to explore this CSV.

Load the country CSV file using the function np.loadtxt() as follows:

```
mnl = np.loadtxt( path_nl, usecols=list(range(3, 10)),
                  dtype=np.float, delimiter=";")
metadata = np.loadtxt(path_nl, usecols=(0, 1, 2),
                  dtype=np.float, delimiter=";")
```

The variable "mnl" contains the weather variables for each of the selected pixels, whereas the variable "metadata" contains the row id and the pixel coordinates. You will use rf.predict() with the variable "mnl" and stack the prediction with the metadata. To do so, you can use the following code:

```
yprednl = rf.predict(xtestnl).reshape(-1, 1)
stack = np.hstack((metadata, yprednl))
```

The variable "stack" now contains the metadata plus the predicted tick activity for each individual pixel. After this step, you are going to visualize the predictions as a raster. However, first it is necessary to place each of the predictions in a raster of 350 rows and 300 columns, which corresponds to the projected map of the Netherlands (EPSG:28992, RD_New) at 1km of pixel resolution.

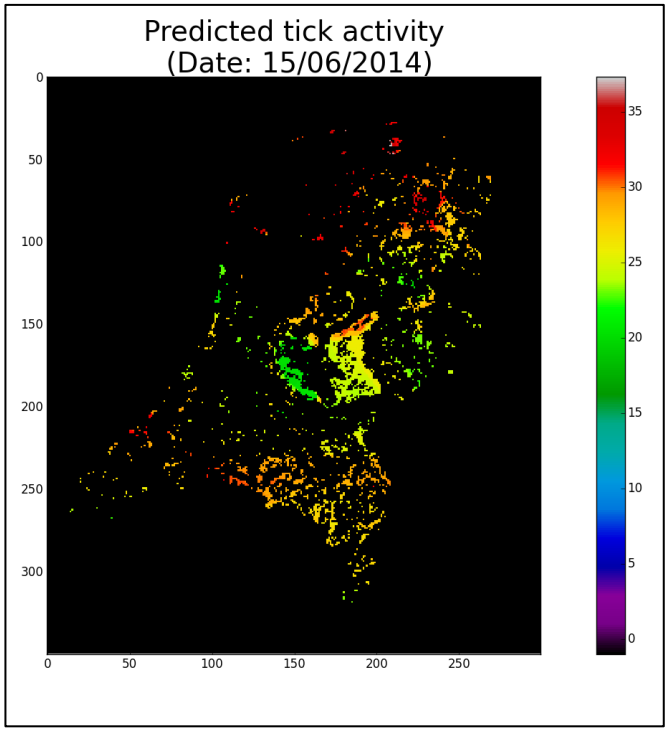
To do this positioning of the CSV coordinates, we provide the following function that you must copy right after the import section of your Python script:

```
def place_pixel(stack):
    placed = np.multiply(np.ones((350, 300)), -1)
    for position in stack:
        row = position[1]
        col = position[2]
        value = position[3]
        placed[row, col] = value
    return np.round(placed, decimals=2)
```

Call this function at the end of your main program and collect the placed numpy array in a variable called “raster”. To visualize this raster, you can use the function plt.imshow() as follows:

```
plt.title("Predicted tick activity \n (Date: 15/06/2014)",
          size=28)
plt.imshow(map, interpolation="None",
          cmap=plt.get_cmap("nipy_spectral"))
plt.colorbar()
plt.show()
```

You should get an output similar to the image below. As you can see, this map visually depicts the results of a complex model. For that date, the highest tick activity was concentrated around the Frisian Islands along the coast. Moderate tick activity is spotted around the Veluwe National Park in the center of the image.



Congratulations! You have completed you first spatio-temporal analysis with machine learning algorithms!