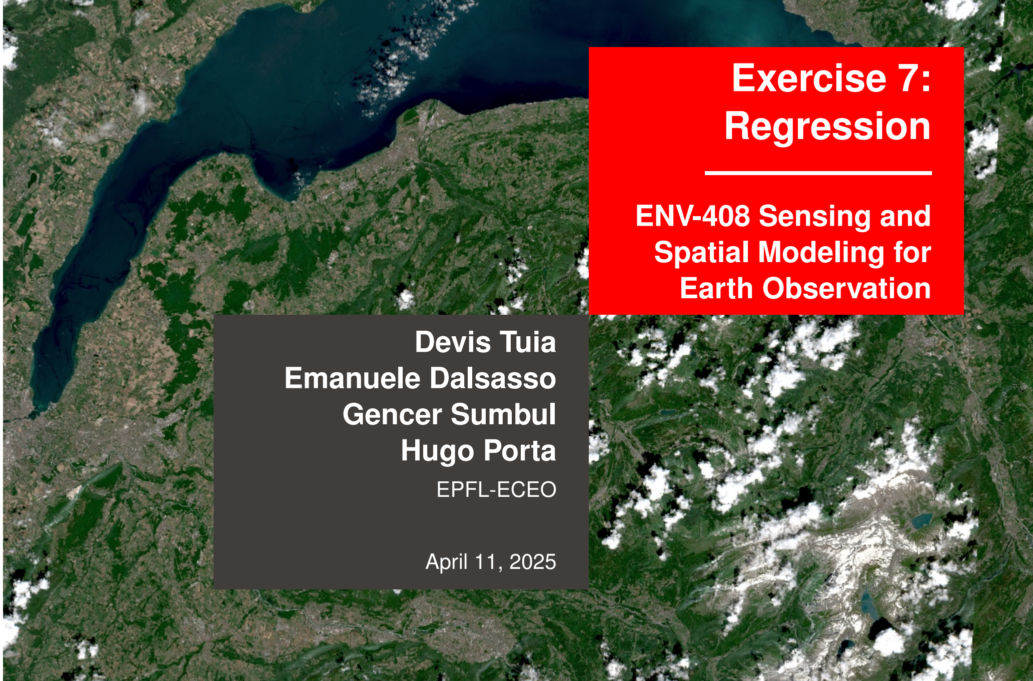


Exercise 7: Regression

ENV-408 Sensing and Spatial Modeling for Earth Observation

Devis Tuia
Emanuele Dalsasso
Gencer Sumbul
Hugo Porta
EPFL-ECEO

April 11, 2025

École polytechnique fédérale de Lausanne

GENCER SUMBUL & EMANUELE DALSASSO & HUGO PORTA

Welcome our Exercise on Regression

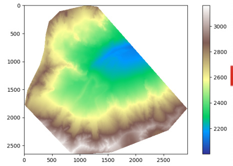
In this graded exercise 7:

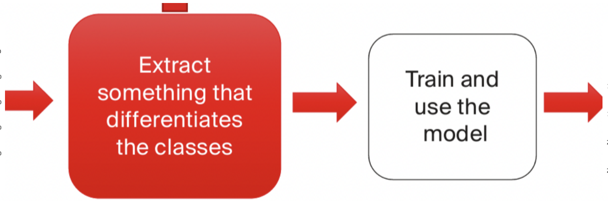
- we use a new DEM
- with the same feature types (DoG, Derivatives, Slope, Aspect) from Ex. 6
- while analyzing their relationship to target variables (average temperature and surface temperature) with regression models

Ex 1 (CRYOS)	Image measurements, coordinates and distorsion model
Ex 2 (ECEO)	Keypoint detection, description and matching
Ex 3 (CRYOS)	Absolute Orientation (Camera Pose)
Ex 4 (CRYOS)	Image Relative Orientation (Two Views)
Ex 5 (CRYOS)	Image Orientation, Orthophoto and DEM Creation
Ex 6 (ECEO)	Feature extraction from the DEM
Ex 7 (ECEO)	**Regression**

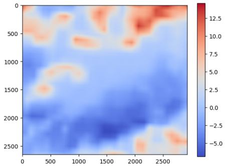
Machine Learning Pipeline

elevation model

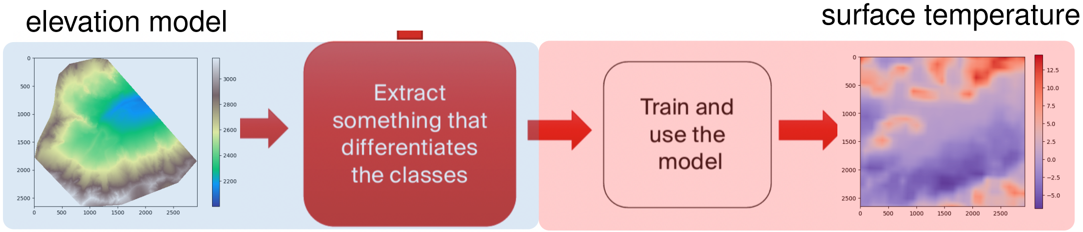




surface temperature



Machine Learning Pipeline



exercise 6 lead by H. Porta

exercise 7 lead by G. Sumbul

The overall outline is:

1. Setup
2. Data Exploration
3. Univariate Linear Regression
4. Multivariate Linear Regression
5. Feature Selection
6. Non-Linear Regression with Random Forest

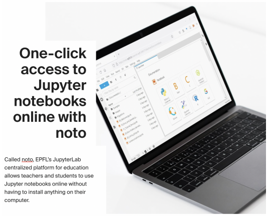
Jupyter Notebook Exercise

Download the notebook from Moodle and open it in your editor of choice

Jupyter

Visual Studio code

EPFL Noto



One-click access to Jupyter notebooks online with noto

Called noto, EPFL's JupyterLab centralized platform for education allows teachers and students to use Jupyter notebooks online without having to install anything on their computer.

Google Colab

Instructions are in the notebook. Ask your colleagues and us if you have technical issues or questions.

2.3 Extract Tabular Data from Rasters

Each pixel location in the above maps defines one data sample (x^i, y^i).

- x^i will be a vector of input features (dem, slope, etc)
- y^i will be either average temperature, or surface temperature.

The overall dataset $(x^i, y^i)_{i=0}^{N}$ contains all N (non-nan) pixels in the above images.

As we do not consider the spatial relationships of pixels to their neighbors, it is easiest to flatten the images and build one long table:

3 Univariate Linear Regression

Let's start simple with

1. a single input: elevation (`dem`) and
2. a single target variable: temperature average: (`tave`)

3.1 Definition of a Linear Model

We can express any linear relationship with this model:

$$\hat{y} = f_\beta(x) = \beta_0 + \beta_1 x$$

with average temperature y, and elevation x. It has the (unknown) parameters: slope β_1 in °C/m and y-intercept β_0 in °C.

3.2 Ordinary Least Squares (OLS)

Let's automate this process of finding $\hat{\beta}_0, \hat{\beta}_1$ with the closed form solution of ordinary least squares:

Overall, we want to find the $\hat{\beta}_0, \hat{\beta}_1$ that minimize the Mean Squared Error:

$$\hat{\beta}_0, \hat{\beta}_1 = \arg\min_{\hat{\beta}_0, \hat{\beta}_1} \mathrm{MSE}(\hat{\beta}_0, \hat{\beta}_1) = \arg\min_{\hat{\beta}_0, \hat{\beta}_1} \frac{1}{N} \sum_{i=0}^{N} \left(f_\beta(x^i) - y^i\right)^2$$

Calculating the derivative of this error with respect to β_0 and β_1, setting it to zero, and solving for β_0 and β_1 brings us to the closed form solution:

$$\hat{\beta}_1 = \frac{\bar{xy} - \bar{y}\bar{x}}{\bar{x^2} - \bar{x}\bar{x}} = \frac{\frac{1}{N}\sum_i (x^i - \bar{x})(y^i - \bar{y})}{\frac{1}{N}\sum_i (x^i - \bar{x})^2} \tag{1}$$

$$\hat{\beta}_0 = \frac{1}{n}\sum_{i=1}^{N} y^i - \hat{\beta}_1 \frac{1}{n}\sum_{i=1}^{N} x^i = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2}$$

TODO Implement the closed form solution in the cell below:

Note, $\bar{x}$ indicates the sample mean: In numpy, you can call the `x.mean()` function on any array

3.3 Evaluation Metrics

Let's test the accuracy of the prediction by implementing the following accuracy metrics. N refers to the number of samples.

Residual Sum of Squares

$$\mathrm{RSS} = \sum_{i=1}^{N} (y^i - \hat{y}^i)^2$$

Residual Standard Error

$$\mathrm{RSE} = \sqrt{\frac{1}{N-2}\mathrm{RSS}}$$

R2 Score: Coefficient of Determination

$$\mathrm{R}^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

using the Total Sum of Squares (TSS)

$$\mathrm{TSS} = \sum_{i=1}^{N} (y^i - \bar{y})^2$$

Hint: Make use of numpy's `.mean()` `.sum()` and `np.sqrt(arr)` functions

4.1 Definition of the Linear Model

The linear model is defined for one sample (indexed i) as: $\hat{y}^i = \mathbf{x}^i \hat{\beta} = \hat{\beta}_0 + x_1^i \hat{\beta}_1 + x_2^i \hat{\beta}_2 + \cdots + x_p^i \hat{\beta}_p$ with p features.

We can also predict the entire dataset with a matrix-vector multiplication $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$, where $\hat{\mathbf{y}}$ is a vector of n samples, $\mathbf{X}$ is a $n \times p$ matrix, and $\hat{\beta}$ a vector of p coefficients.

4.1.1 Data Preparation

First, we need to represent the data as matrices and vectors

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^1 & x_2^1 & x_3^1 & \dots & x_p^1 \\ 1 & x_1^2 & x_2^2 & x_3^2 & \dots & x_p^2 \\ 1 & x_1^3 & x_2^3 & x_3^3 & \dots & x_p^3 \\ \vdots & \vdots & \vdots & \ddots & & \\ 1 & x_1^n & x_2^n & x_3^n & \dots & x_p^n \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y^1 \\ y^2 \\ y^3 \\ \vdots \\ y^n \end{pmatrix}. \tag{1}$$

Important: Notice the 1-column at $\mathbf{X}$: we need to add this to have a y-intercept

4.2 Solution of Ordinary Least Squares

Identically to the uni-variate case, we are looking for the $\boldsymbol{\beta}$ coefficient vector that minimizes the Mean Squared Error (MSE)

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \text{MSE}(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} \|\overbrace{\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}}^{\text{errors } \boldsymbol{\varepsilon}}\|^2$$

Calculating the derivative of this error with respect to $\boldsymbol{\beta}$, setting it to zero, and solving for $\boldsymbol{\beta}$ brings us to the closed form solution:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{1}$$

TODO Implement the closed-form solution

for implementation in numpy, you can use:

- `X.T` to transpose a matrix $\mathbf{X}^T$
- the Moore-Penrose Pseudoinverse `np.linalg.pinv` for the inverse $(\mathbf{X}^T\mathbf{X})^{-1}$