Activity: Creating a clustering learner

Goal. Build an algorithm automatically for grouping similar results in an **election dataset**.

Data format

An **election dataset** is a dataset containing information from an elections process. It will always have the same structure. It will contain the results from an election process with K parties in different regions. It will be stored in a csv file (comma separated file). It will contain as many rows as regions, and it will contain $K + 1$ columns. The first K columns will contain the number of votes to each party in a region, the last column will contain the population in a region.

Examples

The following election dataset

```
57,11,5,3,170
271,14,25,10,594
1940,161,185,104,5515
121,0,4,5,251
298,26,15,16,826
```

contains information of an election process with *four* parties registered in five regions. The results in the first region were 57,11,5 and 3 to each party respectively, and the population in that region was 170.

The following election dataset

```
6,295,30,15,12,43,650
72,845,78,135,45,64,2351
8,78,3,8,4,14,192
0,32,5,6,4,6,78
```

contains information of an election process with *six* parties registered in four regions. The results in the first region were 6,295,30,15,12 and 43 to each party respectively, and the population in that region was 650.

Assignment

Create two scripts called `learner` and `predictor` (the extension will depend on the language used: R, py, jl, java, cpp, …) performing the following actions.

- The `learner` script should read an election dataset stored in a csv file called `training.csv`, and create a `param.out` file containing the hyperparameters and parameters that will be used by a clustering algorithm.
- The `predictor` script should read an election dataset stored in a csv file called `testing.csv` and the `param.out` file and should run the clustering algorithm by returning a csv file called `clustering.out` with the cluster assignment for each row in the election dataset.

The elections datasets provided to the `learner` and `predictor` script will be different, but both will have the same structure (same number of parties) and similar multivariate distribution (dataset obtained with similar conditions).

About the `param.out` file

The definition for the clustering algorithm should be stored in a text file called `param.out` specifying the hyperparameters and parameters needed to run the clustering algorithm (`predicor` script). The first row should always identify the number of clusters, the following rows will heavily depend on the script `predictor`.

About the `clustering.out` file

The clustering out file will contain as many rows as the `testing.csv` file, and each row should contain an integer indicating where each example in `testing.csv` should be assigned.

Examples

Example 1

The following are possible outputs produced by the `learner` script with the definition for the clustering algorithm:

```
2
21.5,312.5,29.0
225.75,1242.75,392.50
```

In this case, we have identified two clusters. The following rows are centres used to assign new observations in `predictor` script. For a new observation $x_{i1}, x_{i2}, x_{i3}, x_{i4}$ the algorithm will return 1 if (x_{i1}, x_{i2}, x_{i3}) is closer to $(21.5, 312.5, 29.0)$ and 2 if it is closer to $(225.75, 1242.75, 392.50)$.

Example 2

In this case,

```
1
```

we have identified one cluster. For a new observation, the algorithm will return 1.

Example 3

In this case,

```
3
-0.083 -0.982 -0.094 -0.052 -0.127
-853.5,-423.7,-112.1
```

we have identified three clusters. For a new observation $x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6}$ a dimensionality reduction is applied to $(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$ using direction $-(0.083, 0.982, 0.094, 0.052, 0.127)$, that is

$$x'_i = -0.083x_{i1} - 0.982x_{i2} - 0.094x_{i3} - 0.052x_{i4} - 0.127x_{i5}.$$

The algorithm will return 1 if x'_i is closer to -853.5231, 2 if it is closer to -423.7022 and 3 if it is closer to -112.1718.