Lab assignment 3: Radial basis functions neural networks

Academic year 2025/2026

Subject: Introduction to computational models
4th course Computer Science Degree (University of Córdoba)

4th November 2025

Abstract

This lab assignment serves as familiarisation for the student with radial basis functions (RBF) neural networks. In this way, a RBF neural network will be developed, using Python and the `scikit-learn` library [1]. In this sense, the assignment will also serve as familiarisation with external libraries, widely used in machine learning (`numpy`, `pandas`...). In addition, we will introduce the problem of bias in machine learning models through `fairlearn`[2]. The student must implement the algorithm and analyse the effect of different parameters over a given set of real-world datasets. Delivery will be made using the task in Moodle authorized for this purpose. All deliverables must be uploaded in a single compressed file indicated in this document. The deadline for the submission is **1st December 2025**. In case two students submit copied assignments, neither of them will be scored.

1 Introduction

The work to be done in this lab assignment consists in implementing a RBF neural network with a training stage divided into three steps:

1. Application of a clustering algorithm which will be used to establish the centres of the RBF (input-to-hidden-layer's weights).
2. The RBF radium adjustment is done by means of a simple heuristic (distance average to the rest of the centres).
3. Hidden-to-output's weights learning:
    - For regression problems, using the Moore-Penrose's pseudo-inverse.
    - For classification problems, using a logistic regression linear model.

The student should develop a Python's script able to train a RBF neural network with the aforementioned characteristics. This programme will be used to train models able to classify as accurate as possible a set of databases available in Moodle. Also, an analysis about the obtained results will be included. For the Triage dataset, we will also perform an algorithmic bias analysis to contrast the behaviour of the models in different demographic groups. **This analysis will greatly influence the qualification of this assignment.**

In the statement of the assignment, indicative values are provided for all parameters. However, it will be positively evaluated if the student finds other values for these parameters able to achieve better results.

[1] `http://scikit-learn.org/`
[2] `https://fairlearn.org/`

Section 2 describes a series of general guidelines when implementing the training algorithm for RBF neural networks. Section 3 explains the experiments to be carried out once the algorithm is implemented. Finally, section 4 specifies the files to be delivered for this assignment.

2 Implementation of the RBF neural network training algorithm

2.1 Model's architecture to be considered

The RBF neural network models should have the following architecture:

- An input layer with as many neurons as input variables the dataset has.
- A hidden layer with a number of neurons specified by the user. It is important to highlight that, in the two previous lab assignment, the number of hidden layer was variable. However, for this lab assignment, we are going to consider just one hidden layer. The type of all the neurons in the hidden layer will be RBF (in contrast to the sigmoidal neurons used in the previous lab assignments).
- An output layer with as many neurons as output variables the dataset has.
  - When considering regression datasets, all the output neurons will be linear (i.e. similar to the sigmoidal neurons without the application of the $\frac{1}{1+e^{-x}}$ transformation).
  - When considering classification datasets, all the output neurons will be softmax. The softmax function is already implemented by the logistic regression algorithm used for adjusting the weights of the output layer.

2.2 Weights adjustment

The instructions given in the class slides should be followed so that the training is carried out as follows:

1. Application of a clustering algorithm that will serve to establish the centres of the RBF (input-to-output layer weights). For classification problems, the centroid initialisation will be random and stratified, n_1 patterns[3]. For regression problems, n_1 will be randomly selected. After initialising the centroids, the `sklearn.cluster.KMeans` class will be used, with only one centroid initialisation (`n_init`) and a maximum of 500 iterations (`max_iter`).
2. To adjust the radium of the RBF, a simple heuristic will be applied (the half of the distance average to the rest of the centres). This is, the radium of the j-th neuron will be[4]:

$$\sigma_j = \frac{1}{2 \cdot (n_1 - 1)} \sum_{i \neq j} \|c_j - c_i\| = \frac{1}{2 \cdot (n_1 - 1)} \sum_{i \neq j} \sqrt{\sum_{d=1}^{n} (c_{jd} - c_{id})^2}. \quad (1)$$

3. Learning the weights from hidden-to-output layer.
   - For regression problem, it is done using the Moore-Penrose pseudo-inverse. This is:

$$\boldsymbol{\beta}^{\mathrm{T}}_{((n_1+1)\times k)} = (\mathbf{R}^{+})_{((n_1+1)\times N)} \, \mathbf{Y}_{(N\times k)} = \quad (2)$$

$$= \left(\mathbf{R}^{\mathrm{T}}_{((n_1+1)\times N)} \times \mathbf{R}_{(N\times(n_1+1))}\right)^{-1} \mathbf{R}^{\mathrm{T}}_{((n_1+1)\times N)} \mathbf{Y}_{(N\times k)} \quad (3)$$

[3]For this, the `sklearn.model_selection.train_test_split` method can be used. It performs one or more stratified dataset partitions, this is, keeping the ratio of patterns belonging to each class in the original dataset `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

[4]Consider using the functions `pdist` and `squareform` of `scipy` to obtain the distances matrix

where $\mathbf{R}$ is the matrix containing the outputs of the RBF neurons, $\boldsymbol{\beta}$ is a matrix containing a vector of parameters for each of the outputs to be predicted, and $\mathbf{Y}$ is a matrix with the target outputs. To perform these operations, we will use the matrix functions of `numpy`, which is a dependence of `scikit-learn`.

- For classification problems, it is done using a logistic regression linear model. Using the `sklearn.linear_model.LogisticRegression` class, providing a value for the C parameter in order to apply regularisation. Note that in this library what we are specifying is the cost value C (importance of the approximation error versus the regularisation error), in such a way that $\eta = \frac{1}{C}$. We will use the `saga solver` and maximum of iterations (`max_iter`) of 10.

3 Experiments

We will test different configurations of the neural network and execute each configuration with five seeds (1, 2, 3, 4 and 5). Based on the results obtained, the average and standard deviation of the error will be obtained. For the regression problems, only the MSE will be shown. However, for classification problems, the CCR (the percentage of correct classified patterns) will be shown. To analyse algorithmic bias in the Triage dataset we will use the false positive rate, which is the most appropriate metric for this particular problem, but you can optionally include the false negative rate.

To assess how the implemented algorithm works, we will run it on three different regression datasets:

- *Sin-function dataset*: This dataset is composed of 120 training patterns and 41 testing patterns. It has been obtained by adding some random noise to the sin function (see Figure 1).

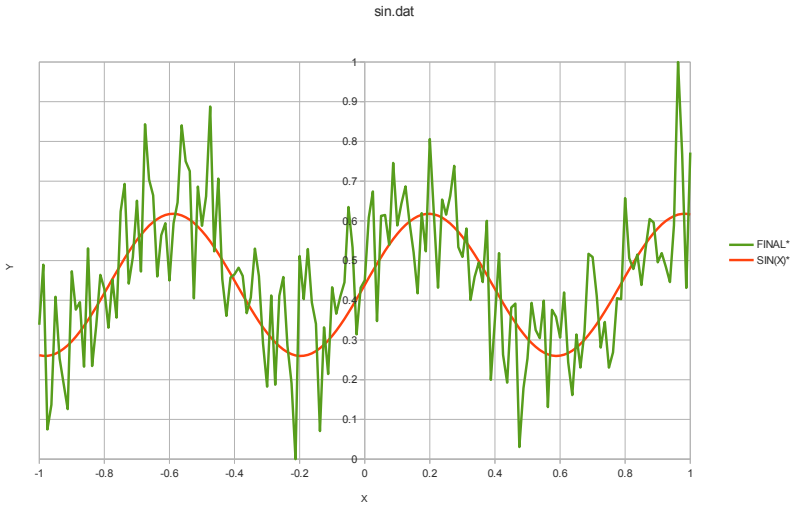

Figure 1: Data representation of the data included in the sin-function estimation problem.

- *Quake dataset*: this dataset is composed by 1633 training patterns and 546 testing patterns. It corresponds to a database in which the objective is to find out the strength of an earthquake (measured on the Richter scale). As input variables, we use the depth of focus, the latitude at which it occurs and the longitude [5].

[5] see `https://sci2s.ugr.es/keel/dataset.php?cod=75` to seek more information.

- Significant wave height (SWH): The dataset (buoy_46025) used in this study comes from time series of significant wave height (SWH) recorded by ocean buoys from the National Data Buoy Centre (NDBC) located on the west coast of the United States (see Figure 2). SWH forecasting is an active area in renewable energy research. For this assignement, one station has been considered (ID 46025). Each buoy provides hourly measurements of oceanic and meteorological variables, from which supervised examples were constructed where the target variable is wave height with a six-hour forecast horizon (WVHT-6h). Details of the dataset can be found in C. Peláez-Rodríguez et. al[6]. The predictive variables used are listed in Table 1.

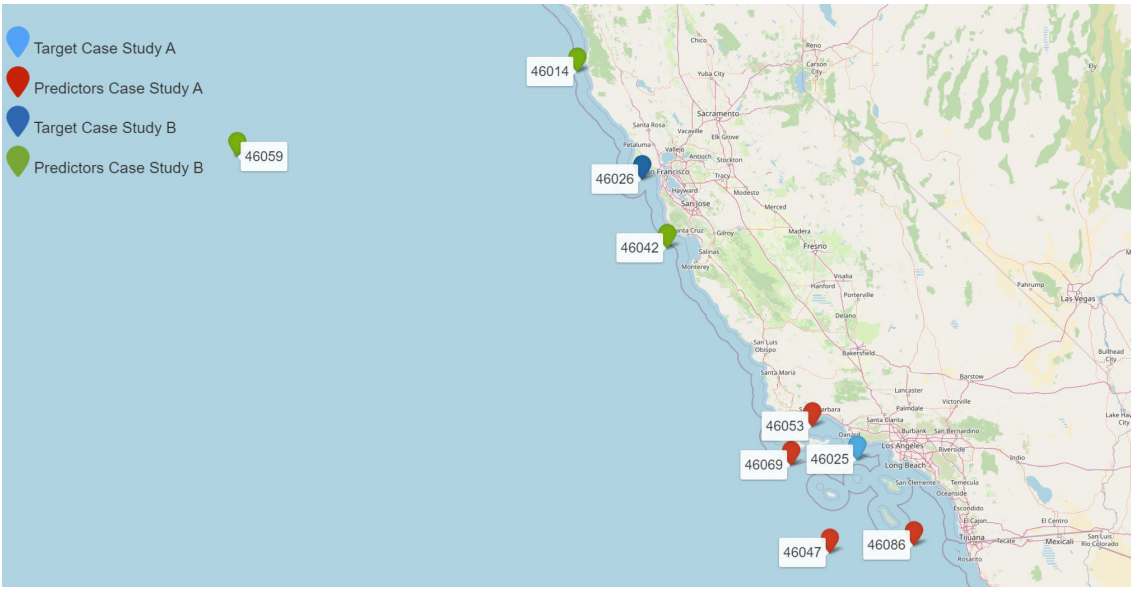

Figure 2: Geographic locations of the stations considered for target and predictors. Source https://doi.org/10.1016/j.apor.2024.104273

Table 1: Predictive variables used in case studies (46025 samples)

Variable	Description	Min	Max
air	Air temperature (K)	277.96	305.96
omega	Omega vertical velocity (Pa/s)	−0.37	0.30
pr_wtr	Precipitable water content (kg/m^2)	2.30	46.11
pres	Pressure (Pa)	95 300.82	98 856.65
rhum	Relative humidity (%)	18.98	100.00
uwnd	South-north wind speed (m/s)	−7.88	11.68
vwnd	West-east wind speed (m/s)	−7.47	12.28
WVHT	Current wave height (m)	0.42	3.26

And two classification datasets:

- *Triage*: this dataset was published on Kaggle in 2022 by Hossam Ahmed Aly [7]. It is based on three individual datasets from three urgent illnesses/injuries, each one with its own features and symptoms recorded for each patient. Then, they were merged to know what

[6] https://doi.org/10.1016/j.apor.2024.104273
[7] https://www.kaggle.com/datasets/hossamahmedaly/patient-priority-classification

are the most severe symptoms for each illness and give them priority of treatment. The three original datasets include:

- Diabetes: all types of diabetes can lead to excess sugar in the blood, and this can result in serious health problems.
- Heart Attack: it occurs when the flow of blood to the heart is severely reduced or blocked. The blockage is usually due to build-up of fat, cholesterol and other substances in the heart (coronary) arteries.
- Stroke: the blood supply to part of the brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Brain cells begin to die in minutes.

The objective is to perform a triage based on the data provided. Triage is the prioritization of patient care (or victims during a disaster) based on illness/injury, symptoms, severity, prognosis and resource availability. The purpose of triage is to identify patients needing immediate resuscitation, to assign patients to a predesignated patient care area (prioritizing their care), and/or to initiate diagnostic/therapeutic measures as appropriate. For this dataset, the patients are assigned one of two classes: non-priority (class 0) and priority (class 1). There are total of 16 input variables: age, gender, chest pain type, blood pressure, cholesterol, max heart rate, exercise angina, plasma glucose, skin_thickness, insulin, bmi, diabetes_pedigree, hypertension, heart_disease, Residence_type, smoking_status. The target variable is whether the patient should have priority or not.

1. age: numerical age.
2. chest pain type: categorical
3. blood pressure: numerical
4. cholesterol: numerical
5. max heart rate: numerical
6. exercise angina: binary
7. plasma glucose: numerical
8. skin_thickness: numerical
9. insulin: numerical
10. bmi: numerical
11. diabetes_pedigree: numerical
12. hypertension: binary
13. heart_disease: binary
14. Residence_type: binary (0 means 'Urban' and 1 means 'Rural')
15. smoking_status: categorical (0 means 'never smoked', 1 means 'formerly smoked', 2 means 'smokes' and 3 means 'Unknown')
16. gender: binary gender (0 means 'Male' and 1 means 'Female').

We will have to make an algorithmic bias analysis to see if there are notable differences according to the gender variables (differences between males and females), when assigning priority. You should also interpret what that bias is telling us. Note that the input variables of this dataset have not been previously standarized, therefore you have to do it. The gender variable is the last one of the input variables in the `csv`.

- *noMNIST dataset*: originally, this dataset was composed by 200.000 training patterns and 10.000 test patterns, with a total of 10 classes. Nevertheless, for this lab assignment, the size of the dataset has been reduced in order to reduce the computational cost. In this sense, the dataset is composed by 900 training patterns and 300 test patterns. It includes a set of

letters (from a to f) written with different typologies or symbols. They are adjusted to a squared grid of 28×28 pixels. The images are in grey scale in the interval $[-1.0; +1.0]$[8]. Each of the pixels is an input variable (with a total of $28 \times 28 = 784$ input variables) and the class corresponds to a written letter (a, b, c, d, e y f, with a total of 6 classes). Figure 3 represents a subset of 180 training patterns, whereas figure 4 represents a subset of 180 letters from the test set. Moreover, all the letters are arranged and available in Moodle in the files `train_img_nomnist.tar.gz` and `test_img_nomnist.tar.gz`, respectively.

Figure 3: Subset of letters belonging to the training dataset.

Figure 4: Subset of letters belonging to the test dataset.

All these datasets are in csv format. The number of holdout partitions is equal to the number of seeds included by the user (parameter n). For this, the class `sklearn.model_selection.train_test_split` can be used. The size of the test set will be 25% (i.e., `test_size=0.25`). Moreover, patters will be shuffled using the parameter `suffle=True` and, for classification problems, train and test partitions have to be stratified.

The average and standard deviation of two measures (regression) or four measures (classification) should be computed:

- Regression: average and standard deviation of training and testing MSE.
- Classification: average and standard deviation of training and testing CCR.

At least, the following configurations should be tried:

- *Network architecture*:
  - For all the datasets, consider a number hidden neurons (n_1) equal to the 5%, 15%, 25% and 50% of the total number of patterns of the dataset. In this stage, for classification problems use L1 regularisation and $\eta = 10^{-5}$.
- For the classification problems, once decided the best architecture, try the following values for η: η: $\eta = 10^{-3}$, $\eta = 10^{-2}$, $\eta = 10^{-1}$, $\eta = 1$, $\eta = 10^{1}$, $\eta = 10^{2}$, $\eta = 10^{3}$, along with the two types of regularisation (L2 y L1). What is happening? Compute the difference in number of

[8]Check `http://yaroslavvb.blogspot.com.es/2011/09/notmnist-dataset.html` for more information.

coefficients for Triage and noMNIST dataset when the regularisation type is modified (L2 vs L1)[9].

- Implement the option `-v` so that, in classification problems, the classifier is trained using `sklearn.linear_model.LogisticRegressionCV` instead of using the class `sklearn.linear_model.LogisticRegression`. This will automatically calculate (without looking at the test values), the optimal value of η. Set the object so that the values to be tested (`Cs`) are $\eta = 10^{-3}$, $\eta = 10^{-2}$, $\eta = 10^{-1}$, $\eta = 1$, $\eta = 10^{1}$, $\eta = 10^{2}$ and $\eta = 10^{3}$ and that the optimization is done using a stratified *k-fold* with $k = 3$. Compare, on the two classification datasetss, the results you get using this strategy (under the assumption of L1 and the best ratio of RBFs obtained above). Compare also the values you choose for eta with those obtained by the experimentation of the previous point. What can the differences be due to?

- For one of the classification problems, run the script considering the problem as if it was regression (i.e. the classification parameter is False and compute the CCR rounding the predictions to the closest integer). What is happening for this situation?

As a guideline, the training and generalisation errors over 5 runs achieved by a linear regression (using sklearn) over the three regression datasets is shown:

- *sin dataset*: $MSE_{\text{train}} = 0.029729$; $MSE_{\text{test}} = 0.036545$.
- *Quake dataset*: $MSE_{\text{train}} = 0.029648$; $MSE_{\text{test}} = 0.028939$.
- *SWH dataset*: $MSE_{\text{train}} = 0.002194$; $MSE_{\text{test}} = 0.001842$.

Also, the training CCR and the test CCR, also over 5 runs achieved by a logistic regression (using sklearn) over the two classification datasets is shown:

- *Triage dataset*: $CCR_{\text{train}} = 95.4407\%$; $CCR_{\text{test}} = 95.9707\%$.
- *noMNIST dataset*: $CCR_{\text{train}} = 100.000000\%$; $CCR_{\text{test}} = 82.666667\%$.

The student should be able to improve this error values with some of the configurations.

3.1 File format

The files containing the datasets will be CSV, in such a way that the values will be separated by commas. In this sense, there are no headers. In order to read the files properly, the function `read_csv` from `pandas` should be used. In the Triage database, the 'gender' variable has been placed in the last column so that it can be easily processed and integrated with `fairlearn`.

4 Deliverables

The files to be submitted will be the following:

- Report in a `pdf` file describing the programme implemented, including results, tables and their analysis.
- Python script.

[9] The coefficients are in the `coef_` attribute of the logistic regression object. Consider that if the absolute value of a coefficient is lower than 10^{-5}, then the coefficient is null

4.1 Report

The report for this lab assignment must include, at least, the following content:

- Cover with the lab assignment number, its title, subject, degree, faculty department, university, academic year, name, DNI and email of the student
- Index of the content with page numbers.
- Description of the steps for the RBF training stage (**1 page maximum**).
- Experiments and results discussion:
  - Brief description of the datasets used.
  - Brief description of the values of the parameters considered.
  - Results obtained, according to the format specified in the previous section.
  - Discussion/analysis of the results. The analysis must be aimed at justifying the results obtained instead of merely describing the tables. This analysis should include algorithmic bias assessment in Triage. Take into account that this part is extremely decisive in the lab assignment qualification. The inclusion of the following comparison items will be appreciated:
    * Test confusion matrix of the best neural network model achieved for the *noMNIST* database. Analysing the errors, **including the images of some letters for which the model mistakes**, to visually check if they are confusing. Comparison between the confusion matrix obtained for this assignment against the one obtained in the previous lab assignment.
    * Computational time needed for the training step for *noMNIST* dataset and comparison against the computational time spent in the previous lab assignment.
- Bibliographic references or any other material consulted in order to carry out the lab assignment different to the one provided by the lecturers (if any).

Although the content is important, the presentation, including the style and structure of the document will also be valued. The presence of too many spelling mistakes can decrease the grade obtained.

4.2 Executable and source code

Together with the report, the two files (main and class) prepared to be run in the UCO's machines (concretely, test using `ssh` on `ts.uco.es`) must be included. In addition, all the source code must be included. The script developed should receive the following command-line arguments[10]:

- Argument `-d`, `--dataset_filename`: Indicates the name of the file that contains the dataset to be used. This argument is compulsory, and without it, the program can not work.
- Argument `-s`, `--standarize`: Boolean indicating whether the data sets are to be standardized (inputs and outputs for regression, only inputs for classification). If not specified, we will assume that it is not standardized.
- Argument `-c`, `--classification`: Boolean that indicates whether it is a classification problem. If it is not specified, we will suppose that it is a regression problem.
- Argument `-r`, `--ratio_rbf`: Indicates the radium (by one) of RBF neurons with respect to the total number of patterns in training. If not specified, use 0.1.

[10]To process the input sequence, the `click` library will be used.

- Argument `-l`, `--l2`: Boolean that indicated if L2 regularisation is used, instead of L1. If it is not specified, L1 will be used.
- Argument `-e`, `--eta`: Indicates the value for the eta (η) parameter. By default, use $\eta = 1e-2$.
- Argument `-f`, `--fairness`: Boolean that indicated if fairness metrics should be extracted from predictions. Assumes that the group is stored as the last variable of the input variables. By default, it is disabled.
- Argument `-v`, `--logisticcv`: Boolean for activating the use of the class `LogisticRegressionCV` for classification problems. By default, it is assumed that it will not be applied (therefore, `LogisticRegression` is used).
- Argument `-n`, `--seeds`: Number of different seeds to be used. By default, $n = 5$ and the seeds to be used would be from 0 to 4 (both included).
- Argumento `-cm`, `--cm_out_folder`: Indicates the directory in which the confusion matrices are saved. This argument is not required, if not specified, the calculation of the confusion matrices will be omited.
- (Kaggle) Argument `-p`, `--pred`: Integer that indicates the model to be used (the one that has been trained with that seed). Also, the file for which the predictions have to be obtained has to be named as dataset_kaggle.csv. This file has to have the same structure than dataset.csv, the one used to train the models.
- (Kaggle) Argument `-m`, `--model_file`: Indicates the directory in which the trained models are saved (in the training mode, without the flag p) or the file containing the model that will be used (in the prediction mode, with the flag p).
- Argument `--help`: It shows the help of the program (use the one automatically generated by the `click` library).

An example of execution can be seen in the following output[11]:

```
(python3.11) maccordoba@srvrrycarn04:~/IMCP3$ python main.py --help
Usage: main.py [OPTIONS]

Run several executions of RBFNN training and testing.

RBF neural network based on hybrid supervised/unsupervised training. Every
run uses a different seed for the random number generator. The results of
the training and testing are stored in a pandas DataFrame.

[...]

Options:
-d, --dataset\textbackslash{}\_filename TEXT  Name of the file with training data.
[required]
-s, --standarize             Standarize input variables.
-c, --classification         Use classification instead of regression.
-r, --ratio\textbackslash{}\_rbf FLOAT        Ratio of RBFs with respect to the total
    number
of patterns.  [default: 0.1]
-l, --l2                     Use L2 regularization for logistic regression.
-e, --eta FLOAT              Value of the regularization factor for logistic
regression.  [default: 0.01]
```

[11] To make the developed code to work in the UCO machines, the packages `click` and the last version of the package `scikit-learn` should be installed, using the following commands:
`pip install scikit-learn --user --upgrade`
`pip install click --user --upgrade`

```
-f, --fairness               Calculate fairness metrics.
-v, --logisticcv             Use LogisticRegressionCV.
-n, --seeds INTEGER          Number of seeds to use.  [default: 5]
-m, --model\textbackslash{}\_filename TEXT    Directory name to save the models (or name of
the file to load the model, if the prediction
mode is active).
-p, --pred INTEGER           Specifies the seed used to predict the output
of the dataset\textbackslash{}\_filename.
-cm, --cm_out_folder TEXT    Name of the folder to save confusion matrices.
--help                       Show this message and exit.

# Example with Triage dataset and fairness parameter
(python3.11) maccordoba@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/triage.csv -c --l2 -f
Running on triage - seed: 0.
Running on triage - seed: 1.
Running on triage - seed: 2.
Running on triage - seed: 3.
Running on triage - seed: 4.
******************
Summary of results
******************
                      MSE        CCR        FN0        FN1       FP0       FP1
seed partition
0    Train      0.060045  93.995522  66.666667  76.987448  0.604933  0.830565
     Test       0.068376  93.162393  75.675676  82.926829  0.664011  1.436031
1    Train      0.063709  93.629147  68.224299  79.919679  0.826067  0.967199
     Test       0.062271  93.772894  80.851064  75.000000  0.414938  0.879397
2    Train      0.065337  93.466314  74.137931  85.416667  0.547945  0.760456
     Test       0.061661  93.833944  71.052632  75.308642  0.702247  0.991326
3    Train      0.063098  93.690210  73.553719  80.425532  0.504356  0.883838
     Test       0.067766  93.223443  87.878788  82.558140  0.554785  0.877193
4    Train      0.063912  93.608793  70.940171  80.334728  0.732265  0.969646
     Test       0.063492  93.650794  78.378378  82.926829  0.278940  0.623441
Mean Train      0.063220  93.677997  70.704557  80.616811  0.643113  0.882341
     Test       0.064713  93.528694  78.767307  79.744088  0.522984  0.961478
Std  Train      0.001955   0.195548   3.256437   3.033688  0.133406  0.089947
     Test       0.003143   0.314274   6.256732   4.193987  0.176257  0.297574

# En los siguientes ejemplos se lanzan varios problemas de regresión:
(python3.11) maccordoba@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/bouys.csv -r 0.5 -s -n 3
Running on bouys - seed: 0.
Running on bouys - seed: 1.
Running on bouys - seed: 2.
******************
Summary of results
******************
                      MSE
seed partition
0    Train      0.063646
     Test       0.514725
1    Train      0.064947
     Test       0.704460
2    Train      0.066099
     Test       0.664160
Mean Train      0.064898
     Test       0.627781
Std  Train      0.001227
     Test       0.099962

(python3.11) maccordoba@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/bouys.csv -r 0.15 -s
Running on bouys - seed: 0.
Running on bouys - seed: 1.
Running on bouys - seed: 2.
```

```
Running on bouys - seed: 3.
Running on bouys - seed: 4.
******************
Summary of results
******************
                      MSE
seed partition
0    Train      0.137546
     Test       0.191664
1    Train      0.135316
     Test       0.254717
2    Train      0.142577
     Test       0.209315
3    Train      0.141903
     Test       0.202015
4    Train      0.137473
     Test       0.213062
Mean Train      0.138963
     Test       0.214155
Std  Train      0.003132
     Test       0.024099

(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/sin.csv -r 0.15
Running on sin - seed: 0.
Running on sin - seed: 1.
Running on sin - seed: 2.
Running on sin - seed: 3.
Running on sin - seed: 4.
******************
Summary of results
******************
                      MSE
seed partition
0    Train      0.012505
     Test       0.022467
1    Train      0.012703
     Test       0.025110
2    Train      0.012367
     Test       0.023760
3    Train      0.013290
     Test       0.022022
4    Train      0.012436
     Test       0.022403
Mean Train      0.012660
     Test       0.023152
Std  Train      0.000374
     Test       0.001276
```

4.3 Error control

It is necessary to introduce a quite detailed error control. For example, if we are in the prediction mode for Kaggle, we have to indicate both the file where the model is and the seed we want to use for prediction.

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/nomnist.csv -r
    0.05 -s -n 2 -p 1
Traceback (most recent call last):
File "/home/dguijo/IMCP3/main.py", line 530, in <module>
main()

[...]

File "/home/dguijo/IMCP3/main.py", line 166, in main
raise ValueError("You have not specified the model directory (-m).")
ValueError: You have not specified the model directory (-m).
```

Another case may be to indicate a folder that does not contain pre-trained models. This will indicate that we have to train the model and therefore we have to remove the prediction mode for Kaggle.

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/nomnist.csv -r
    0.05 -s -n 2 -m model5 -p 1
Running on nomnist - seed: 1.
Prediction mode. Using dataset for Kaggle in ./datasets/nomnist_kaggle.csv
Traceback (most recent call last):
File "/home/dguijo/IMCP3/main.py", line 530, in <module>
main()

[...]

File "/home/dguijo/IMCP3/main.py", line 404, in load
raise ValueError(
ValueError: The model file model5/nomnist/1.p does not exist.
You can create it by firstly using the parameter (n = 1) and removing the flag P (for
    pred) to train the model.
```

Or calculate fairness metrics for a dataset that is not appropriate:

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/nomnist.csv -r
    0.05 -s -n 2 -m models -c -f -p 3
Traceback (most recent call last):
File "/home/dguijo/IMCP3/main.py", line 533, in <module>
main()

[...]

File "/home/dguijo/IMCP3/main.py", line 175, in main
raise ValueError(
ValueError: You can only calculate fairness metrics when using the compas dataset.
```

4.4 [OPTIONAL] Save the model to a file.

During the training stage, the script can save the model trained as a pickle[12]. This will allow to use the trained model to predict the outputs of the **Kaggle** dataset.

To save the model, it is necessary to use the `-m` parameter with the path where the models will be saved. An execution example is as follows:

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/nomnist.csv -r
    0.5 -s -n 3 -m models -c
Running on nomnist - seed: 0.
Model saved in models/nomnist/0.p
Running on nomnist - seed: 1.
Model saved in models/nomnist/1.p
Running on nomnist - seed: 2.
Model saved in models/nomnist/2.p
******************
Summary of results
******************
                      MSE        CCR
seed partition
0    Train       0.647778  90.111111
     Test        0.630000  88.333333
1    Train       0.601111  90.555556
     Test        1.043333  84.666667
2    Train       0.616667  90.777778
     Test        0.633333  88.333333
Mean Train       0.621852  90.481481
     Test        0.768889  87.111111
Std  Train       0.023762   0.339450
```

[12] https://docs.python.org/3/library/pickle.html

```
     Test       0.237682   2.116951
```

Once the execution is finished, there will be a folder named "models" containing 3 pickles (`n=3`). Each one corresponds with the generated model for each seed. In order to obtain the predictions, one of these pickles should be chosen.

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ ls -l models
total 1
drwxr-xr-x 2 dguijo ayrna 6 nov 18 15:40 nomnist

(python3.11) dguijo@srvrrycarn04:~/IMCP3$ ls -l models/nomnist
total 9852
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 0.p
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 1.p
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 2.p
```

4.5 [OPTIONAL] Obtaining the predictions for Kaggle.

Once the model is saved to a pickle, it is possible to obtain the output predictions for the Kaggle dataset. For this, `-m` and `-p` parameters should be used. Below is an example when the model selected is the one trained with the seed 2, which was the one achieving the best results (along with seed 0):

```
(python3.11) dguijo@srvrrycarn04:~/IMCP3$ python main.py -d ./datasets/nomnist.csv -r
    0.5 -s -n 3 -m models -c -p 2
Running on nomnist - seed: 2.
Prediction mode. Using dataset for Kaggle in ./datasets/nomnist_kaggle.csv
Predictions saved in models/nomnist/predictions_2.csv.
******************
Summary of results
******************
                     MSE        CCR
seed partition
2    Train      0.616667  90.777778
     Test       0.633333  88.333333

(python3.11) dguijo@srvrrycarn04:~/IMCP3$ ls -l models/nomnist
total 13807
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 0.p
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 1.p
-rw-r--r-- 1 dguijo ayrna 5675042 nov 18 15:41 2.p
-rw-r--r-- 1 dguijo ayrna   15012 nov 18 15:41 predictions_2.csv

(python3.11) dguijo@srvrrycarn04:~/IMCP3$ head ./models/nomnist/predictions_2.csv
Id,Category
0,2
1,1
2,1
3,4
4,0
5,4
6,3
7,2
8,4
```

This file is ready to be uploaded to Kaggle.