Contenido

[Objective 2](#_Toc150558840)

[Deliverables 2](#_Toc150558841)

[Part 1: Selecting and analyzing the datasets 2](#_Toc150558842)

[Analysis datasets 1 and 2 3](#_Toc150558843)

[Data normalization 3](#_Toc150558844)

[Analysis datasets 3: Data preprocessing 4](#_Toc150558845)

[Instance selection 4](#_Toc150558846)

[Outlier detection 4](#_Toc150558847)

[Missing values 5](#_Toc150558848)

[Data representation 5](#_Toc150558849)

[Data normalization 5](#_Toc150558850)

[Feature selection 5](#_Toc150558851)

[Feature Construction 5](#_Toc150558852)

[Part 2: Implementation of BP 7](#_Toc150558853)

[Part 3: Obtaining and comparing predictions using the three models (BP, BP-F, MLR-F) 8](#_Toc150558854)

[Part 3.1: Parameter comparison and selection 9](#_Toc150558855)

[Part 3.2: Model result comparison 9](#_Toc150558856)

# Objective

Prediction using the following algorithms:

* **Neural Network with Back-Propagation** (**BP**), implemented by the student.
* **Neural Network with Back-Propagation (BP-F**), using free software.
* **Multiple Linear Regression** (**MLR-F**), using free software.

# Deliverables

This assignment can be done alone or in pairs (groups of two)

For this activity you must deliver **one PDF document** that includes:

* A link to the Github repository where the code of all the activity is accessible. More details on the code in the following sections.
* Explanations of the analysis of the data preprocessing, and the results of the executions that are detailed in the section below.
* The name of the file should be **A1-Name1Surname1-Name2Surname2.pdf**

# Part 1: Selecting and analyzing the datasets

The predictions must be performed on three datasets:

1. File: *A1-turbine.txt*

* 5 features: the first 4 are the input variables, the last one is the value to predict.
* 451 patterns: use the first 85% for training and validation, and the remaining 15% for test.

1. File: *A1-synthetic.txt*

* 10 features: the first 9 are the input variables, the last one is the value to predict.
* 1000 patterns: use the first 80% for training and validation, and the remaining 20% for test.
  1. 3 Search a dataset from the Internet, with the following characteristics:
* At least 6 features, one of them used for prediction.
* The prediction variable must take real (float or double) values; it should not represent a categorical value (that would correspond to a classification task)
* At least 400 patterns
* Select randomly 80% of the patterns for training and validation, and the remaining 20% for test; it is important to shuffle the original data, to destroy any kind of sorting it could have.
* In case of doubt on the suitability of the selected data, please ask the professor.

As an output of this part, **you should include in your report the following analysis**:

## Analysis datasets 1 and 2

* For datasets 1 and 2, explain what type of data normalization you are going to apply to the input and output variables (the datasets are already cleaned, no need to preprocess them) and why you apply it.

### Data normalization

La decisión de normalizar o estandarizar datos depende del algoritmo de aprendizaje automático que estás utilizando y de las características de tus datos. Aquí hay algunas pautas generales:

**Normalización (Min-Max Scaling):**

Cuándo usar: La normalización es útil cuando los datos tienen diferentes escalas y deseas que estén en un rango específico, como [0, 1].

El proceso de normalización se utiliza generalmente cuando la distribución de los datos no sigue la distribución gaussiana.

Ejemplo: Algoritmos que utilizan medidas de distancia, como k-Nearest Neighbors (k-NN) o algoritmos basados en gradientes, pueden beneficiarse de la normalización.

|  |
| --- |
| from sklearn.preprocessing import MinMaxScaler  scaler = MinMaxScaler()  normalized\_data = scaler.fit\_transform(data) |

**Estandarización (Z-Score Standardization):**

Cuándo usar: La estandarización es útil cuando los datos tienen diferentes escalas y deseas que tengan una media de 0 y una desviación estándar de 1.

El proceso de estandarización se utiliza generalmente cuando sabemos que la distribución de los datos sigue la distribución gaussiana.

Ejemplo: Algoritmos que asumen una distribución normal de los datos, como regresión lineal o máquinas de soporte vectorial (SVM), pueden beneficiarse de la estandarización.

|  |
| --- |
| from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  standardized\_data = scaler.fit\_transform(data) |

### Se incluye en el repositorio la normalización y estandarización en el archivo siguiente: [Normalization\_synthetic\_turbine.xlsx](https://github.com/Cod3Frank/Activity-1-Prediction-with-Back-Propagation-and-Linear-Regression/blob/main/INPUT%20DATASETS/Normalization_synthetic_turbine.xlsx)

En este archivo se expone el proceso de normalización y estandarización realizado en Excel.

Para ambos datasets se escoge la **estandarización**, ya que tal y como se muestra en las gráficas del archivo [Normalization\_synthetic\_turbine.xlsx](https://github.com/Cod3Frank/Activity-1-Prediction-with-Back-Propagation-and-Linear-Regression/blob/main/INPUT%20DATASETS/Normalization_synthetic_turbine.xlsx) hace pensar que la aproximación de datos sigue la distribución gaussiana.

En el mismo repositorio adjunto un código con el que se puede realizar en Python la normalización y estandarización de un dataset.

### [standardizationVsNormalization.py](https://github.com/Cod3Frank/Activity-1-Prediction-with-Back-Propagation-and-Linear-Regression/blob/main/INPUT%20DATASETS/standardizationVsNormalization.py)

|  |
| --- |
| import pandas as pd  from sklearn.preprocessing import MinMaxScaler, StandardScaler  import matplotlib.pyplot as plt  import seaborn as sns  data = pd.read\_csv("dataset.csv").drop('target', axis = 1)  min\_max\_scaler = MinMaxScaler()  std\_scaler = StandardScaler()  # MinMaxScaler() class fit\_transform function first fits on data and then transform the given data  # and returns array, therefore we have to construct dataframe for normalized data  data\_normalized = min\_max\_scaler.fit\_transform(data)  df\_data\_normalized = pd.DataFrame(data\_normalized, columns= list(data.columns))  # StandardScaler() class fit\_transform function first fits on data and then transform the given data  # and returns array, therefore we have to construct dataframe for standardized data  data\_standardized = std\_scaler.fit\_transform(data)  df\_data\_standardized = pd.DataFrame(data\_standardized, columns= list(data.columns))  plt.figure(figsize = (20,20))  for idx, col in enumerate(list(data.columns)):      plt.subplot(5,3,idx + 1)      sns.kdeplot(data[col], color = 'g',label = f"Input data(Unscaled):{col}")      sns.kdeplot(df\_data\_normalized[col],color = 'b', label = f"Input data(Normalized):{col}" )      sns.kdeplot(df\_data\_standardized[col],color = 'r', label = f"Input data(Standardized):{col}" )      plt.legend()  plt.show() |

|  |
| --- |
| [¿Cómo normalizar datos en Excel? - GeeksforGeeks (en inglés)](https://www.geeksforgeeks.org/how-to-normalize-data-in-excel/)  [¿La estandarización y la normalización transforman los datos en una distribución normal? | por R. Gupta | Python en inglés sencillo (plainenglish.io)](https://python.plainenglish.io/do-standardization-and-normalization-transform-the-data-into-normal-distribution-cb5857ab9c63)  [Data Transformation: Standardization vs Normalization - KDnuggets](https://www.kdnuggets.com/2020/04/data-transformation-standardization-normalization.html)  [▷ Cómo normalizar datos en Excel en 2023 → STATOLOGOS®](https://statologos.com/normalizar-datos-excel/) |

## Analysis datasets 3: Data preprocessing

* For dataset 3, **add to the documentation of this assignment the link to the source webpage.** Then explain what techniques of data preprocessing you are going to apply: check for missing values, represent correctly categorical values, and look for outliers. You should also consider if you must apply data normalization to some (or all) of the input / output variables.

Se descargan los datos de “sistema de información agroclimática para el regadío” desde el siguiente enlace: [servicio.mapa.gob.es/websiar/SeleccionParametrosMap.aspx?dst=1](https://servicio.mapa.gob.es/websiar/SeleccionParametrosMap.aspx?dst=1)

La siguiente imagen muestra los datos solicitados

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

In the following file, all the preprocessing data is collected. Which will be explained in the following sections. [V01\_Pedralba\_04\_11\_2021\_11\_11\_2023.csv](https://github.com/Cod3Frank/Activity-1-Prediction-with-Back-Propagation-and-Linear-Regression/blob/main/INPUT%20DATASETS/V01_Pedralba_04_11_2021_11_11_2023.csv)

### Instance selection

Also known as data instance or data point selection, is a data preprocessing technique that involves choosing a subset of instances (data points) from a larger dataset. The goal is to reduce the size of the dataset while maintaining its representativeness and preserving key characteristics. Instance selection is often employed to address issues such as data redundancy, computational efficiency, and model performance improvement.

In this case, the selected dataset is not excessively long so it is not necessary to make subsets. Furthermore, there are no gaps of non-existent values.

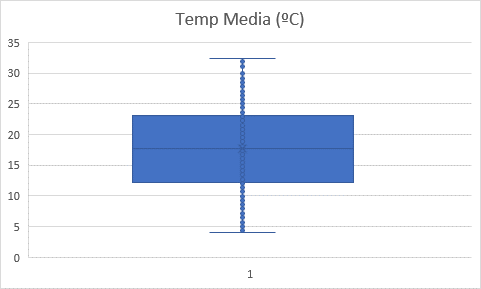
In the "instance selection" tab of the downloaded file "V01\_Pedralba\_04\_11\_2021\_11\_11\_2023" we can see the result of this processing.

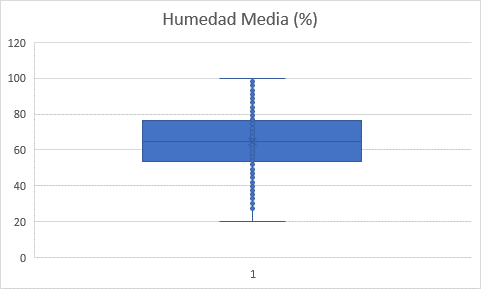
### Outlier detection

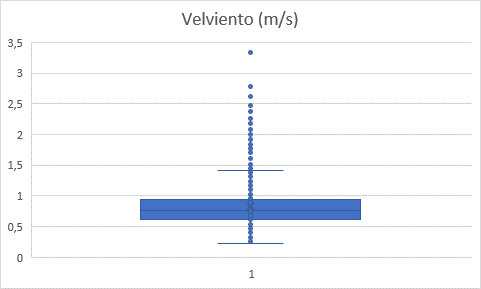
Outlier detection is an essential step in data preprocessing, as outliers can significantly impact the performance and results of machine learning models. Outliers are data points that deviate significantly from the majority of the data. Here are some common techniques for detecting outliers:

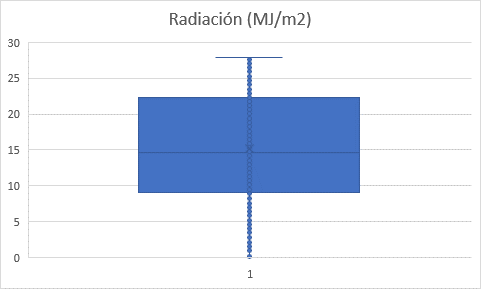
* Visual Inspection:

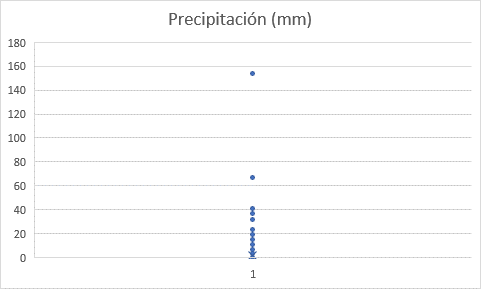
Box Plots: Visualize the distribution of a variable using a box plot, which highlights potential outliers as points beyond the whiskers. The graphs are shown for our specific case.











### Missing values

Decide how to proceed with patterns which have missing values.

Approaches

* Ignore pattern
* Assign value to the missing value: Average, Median, Mode, Regression, New value

In this case there are no missing values, so the sample is taken as is.

### Data representation

Data representation in data preprocessing involves transforming raw data into a format that is suitable for analysis or input into machine learning models. The choice of data representation can significantly impact the performance of the analysis or model.

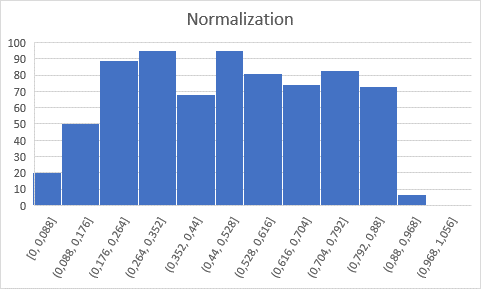
**Numeric Representation:** Many machine learning algorithms require numeric input. Convert categorical variables into numerical representations using techniques such as:

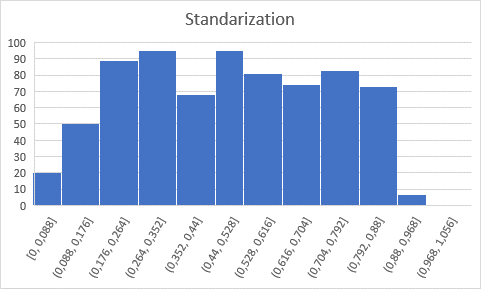
* One-Hot Encoding: Represent each category with a binary column (0 or 1).
* Label Encoding: Assign a unique integer to each category.
* Ordinal Encoding: Assign integers to categories based on their order.

### Data normalization

As was done for the other two previous datasets. The "Data normalization" tab shows the data extracted in the normalization process and by standardization.

As seen in the graphs, these data follow a Gaussian distribution, which is why we opted for the standardization process.





### Feature selection

* Objective: Leave only the relevant variables
* Approaches

Removal of variables: Redundant, unrelated to the objective of the analysis, low quality (missing data, outliers, noisy, etc)

Selection of variables: Identify the relevant variables

### Feature Construction

* Objective: Add new synthetic variables which could facilitate the analysis
* Approaches

Build new variables from the existing ones

Linear or non-linear combinations (e.g., add the squares of some variable)

Change of reference (e.g., from cartesian to polar coordinates)

Once this part is completed, you should generate the preprocessed / normalized files that are going to be the input of your analysis part.

# Part 2: Implementation of BP

[CarlosJesusGH/URV\_NEC\_MachineLearning\_BackProp-LinReg: Prediction with Back-Propagation and Linear Regression (github.com)](https://github.com/CarlosJesusGH/URV_NEC_MachineLearning_BackProp-LinReg)

[GenevieveMasioni/NEC-Back-Propagation: Prediction with Back-Propagation (BP) and Multiple Linear Regression (MLR). (github.com)](https://github.com/GenevieveMasioni/NEC-Back-Propagation)

[How to Code a Neural Network with Backpropagation In Python (from scratch) - MachineLearningMastery.com](https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/)

[Crear una Red Neuronal en Python desde cero | Aprende Machine Learning](https://www.aprendemachinelearning.com/crear-una-red-neuronal-en-python-desde-cero/)

[Guía de Aprendizaje | Aprende Machine Learning](https://www.aprendemachinelearning.com/guia-de-aprendizaje/)

The most important part of this activity is the programming of a neural network with back propagation from scratch. Here are some instructions on how to code the project.

* You should create a project in a github account where you are going to develop the activity. You should do regular commits to this project so we can observe the evolution of the project from the initial file to the final delivery. We will apply a penalization to the grade if there is one single commit in the project.
* We recommend the implementation of the project using Python (version >= 3.6). We are providing a base code that can be used to start the project (MyNeuralNetwork.py).
* In this part you must implement all the methods necessary for the network to learn, you cannot use external libraries that already implement BP or Neural Networks.
* The implementation must be based on the algorithm and equations in document [G] of Unit 3 at Moodle.

Texto

Descripción generada automáticamente con confianza media

* The implementation must use the following variables to hold all the information about the structure of the multilayer neural network and the BP:
  1. L: number of layers
  2. n: an array with the number of units in each layer (including the input and output layers)
  3. h: an array of arrays for the fields (h)
  4. xi: an array of arrays for the activations (ξ)
  5. w: an array of matrices for the weights (w)
  6. theta: an array of arrays for the thresholds (θ)
  7. delta: an array of arrays for the propagation of errors (Δ)
  8. d\_w: an array of matrices for the changes of the weights (δw)
  9. d\_theta: an array of arrays for the changes of the weights (δθ)
  10. d\_w\_prev: an array of matrices for the previous changes of the weights, used for the momentum term (δw(prev))
  11. d\_theta\_prev: an array of arrays for the previous changes of the thresholds, used for the momentum term (δθ(prev))
  12. fact: the name of the activation function that it will be used. It can be one of these four: sigmoid, relu, linear, tanh.
* For example, the weight 𝑤𝑖𝑗(𝐿) between unit *j* in layer *L-1* and unit *i* in layer *L* is accessed as w[L][i,j]
* The idea behind this structure is that the code must be able to deal with arbitrary multilayer networks. For example, a network with architecture 3:9:5:1 (4 layers, 3 input units, 1 output unit, and two hidden layers with 9 and 5 units, respectively), would have n=[3; 9; 5; 1], and xi would be an array of length 4 (one component per layer), with xi[1] and array of real numbers of length 3, xi[2] and array of real numbers of length 9, xi[3] and array of real numbers of length 5, and xi[4] and array of real numbers of length 1. Similarly, w[2] would be an array 9x3, w[3] an array 5x9, and w[4] and array 1x5; w[1] is not used.
* Additionally, the use of this structure, name conventions and array dimensions make it easy to convert the equations into code.
* The code will receive one input dataset, and using the percentage of data that is passed as a parameter in the class constructor, should divide this dataset into training and validation. If the percentage is 0, then we consider that there is no validation, and all the input data is used for training.

The class MyNeuralNetwork **will receive all these parameters in the class constructor**:

* Number of layers
* Number of units in each layer
* Number of epochs
* Learning rate and momentum
* The selected activation function (sigmoid, relu, linear, tanh)
* The percentage of data that should be used as the validation set

The class MyNeuralNetwork **will provide three public functions** that can be called externally:

* + A function **fit(X, y)** that has 2 parameters: an array X of size (n\_samples, n\_features), which holds the training samples represented as floating point feature vectors; and a vector y of size (n\_samples), which holds the target values (class labels) for the training samples. This method allows us to train the network with this data.
  + A function **predict(X)** that has 1 parameter, an array X of size (n\_samples, n\_features) that contains the samples. This method returns a vector with the predicted values for all the input samples.
  + A function **loss\_epochs()** that returns 2 arrays of size (n\_epochs, 2) that contain the evolution of the training error and the validation error for each of the epochs of the system, so this information can be plotted.

# Part 3: Obtaining and comparing predictions using the three models (BP, BP-F, MLR-F)

This part is focused on comparing the results of the backpropagation model that has been developed against two open-source implementations of machine learning methods.

The coding corresponding on this part should be done using jupyter notebooks, which should also be included in the github of the project.

To study the quality of the predictions in this section, we are going to use two elements:

1. Although BP and MLR are based on the minimization of the mean squared error, it does not constitute a useful measure of the prediction error. Therefore, to compute the quality of a prediction, we will use the mean absolute percentage error (MAPE), given by:

Imagen que contiene Logotipo

Descripción generada automáticamente

1. The best way to visualize the results is with scatter plots of the prediction value  compared with the real value *z*. The closer the points are to the diagonal, the better the prediction.

## Part 3.1: Parameter comparison and selection

First, we need to choose the correct set of parameters that will provide the best prediction for the neural network that we have implemented. For this reason, for each of the three datasets we will have to explore what are the optimal values for the network parameters.

We have to explore some of the space of parameters, and the result of the prediction using those parameters. For this reason, you should include in the document **the following information for each of the 3 datasets**:

* A table that summarizes the quality of the prediction of each set of parameters using the MAPE value:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of layers** | **Layer Structure** | **Num epochs** | **Learning Rate** | **Momentum** | **Activation function** | **MAPE** |
|  |  |  |  |  |  |  |

* 2 or 3 representative scatter plots of the prediction value vs real value of some rows of this table (including the set of parameters that gives the minimum MAPE value).
* 2 or 3 plots of the evolution of the training and validation error as a function of the number of epochs (the information that can be obtained calling (the function loss\_epochs).
* A discussion/opinion of 1-2 paragraphs of why you think these parameters are the most adequate.

## Part 3.2: Model result comparison

Second, we are going to compare the results obtained in our network against two already implemented models. We are going to compare the best results obtained in the previous section against the following models:

* A multi-linear regression from scikit-learn.
* A neural network model, which can be used from Tensorflow, Scikit-learn or any other python library.

For this part **you should include in the document**, for each of the three datasets, the following information:

* The description of the parameters used in each of the two additional models.
* A table comparing the prediction quality (MAPE) of the three models.
* The scatter plot of predicted vs real values for the three models.