# RIGHT TO THE CITY AND URBAN NEURAL NETWORKS

Fernando **Visedo Manzanares**
Architect

**Right to the City and Urban Neural Networks**

**Abstract:**
Different disciplines formulate functions that relate economic and social variables. The New Science of Cities delves into the development of urban spatial syntax.

In a previous work on Urban Mechanics and Thermodynamics, the relationship between the Right to the City and the Urban System was explored, formulating the urban matrix that allows modeling the city and establishing the functions that explain the functioning of the Urban or Territorial System.

In a complementary way, the present work develops within the framework of AI (Artificial Intelligence) the treatment of urban big data analyzed by RNU (Urban Neural Networks) to find the hidden function between the different variables that model a Complex Urban System and predict the result of applying measures that contribute to correcting the urban vulnerability of the Right to the City.

**Keywords**: Right to the City; Neural Network; Urban Science; Artificial Intelligence

## 1. Urban Neural Networks

Urban Mechanics[1] allows us to deduce results from equations that relate different social, economic, and spatial variables. The functions that relate these different urban variables are derived from various theoretical and practical studies across different disciplines that analyze the economic, environmental, and spatial vectors aimed at addressing the vulnerability of the set of rights that constitute the Right to the City. These functions are derived after a detailed analysis of the systems. As the system becomes more complex, the difficulty in finding the relational functions also increases.
The current moment is characterized by the availability of data, obtained directly from any field of activity, any research discipline, and everyday life, at any time and in any place. Countless fixed and mobile devices, distributed unpredictably and without oversight, provide information on the habits and behaviors of humans and animals, the conduct of social groups, the state of the environment, and road conditions. This wealth of data penetrates to the most intimate, unimaginable depths of any nature. This set of data, or big data, which requires processing by computer applications for its analysis and application, opens a new field of research and analysis. It allows us to study the relationships between the causes and effects of a phenomenon, described through its variables, without needing to understand the underlying relationships between them. This is achieved by observing the results generated by a phenomenon that is part of a complex system.

Alternatively to the methods described in mechanics, where variables are related through proven functions, predictions of the target variables can be obtained from the input variables through the analysis of Urban Neural Networks within the framework of Deep Learning and Artificial Intelligence. The processing power of computer processors, combined with the methodological capabilities of neural networks, allows for the incorporation of a far greater number of input and output variables than the structured method of Urban Mechanics.
Indeed, the experience of Urban Mechanics allows for the identification of the variables that constitute the main components of each of the Urban Fields. The path taken by the set of submatrices of the Urban Matrix, described above, links all public and private investments to addressing the vulnerability of the Right to the City.

This path provides sufficient experience to develop analog or Boolean Urban Neural Networks (UNNs), which enable decision-making based on predictions obtained from an unlimited number of urban variables that define a complex urban system.
Two UNNs have been built in Visual Studio Code and Colab. The first network, called RNU8_2, is built with 8 input neurons and 2 output neurons to predict the population variation value and the (n0/n1) – (1/e) ratio, which indicates the city's tendency toward survival or decline. The second network, called RNU1_15, was built with 1 input neuron and 15 output neurons to predict the impact of increased public investment on addressing rights vulnerabilities.
The calculated RNUs were tested with 1, 2, and 3 layers of hidden neurons to verify the prediction error. This prediction was verified by inputting available urban values and comparing them with the corresponding output predictions based on the actual input values. A 5% error was considered acceptable, so the network with 3 hidden neurons provides results closer to reality.

In the first RNU, high learning is achieved from cycle 200 onwards. Normalization and optimization of training have been carried out with the Adam activation function with approximation 0.001 and loss through the squared error.

Reality, the same input can produce different outputs, as it depends on each city or urban system, where some variables may have coinciding values, but others may have disparate values.
This circumstance creates uncertainty in the forecast, since the disturbance of a single
variable is intended to predict the impact on 15 other variables. Therefore, it is necessary to adopt the necessary precautions as well as complementary checks with different models that are constructed with more input variables.

The result of the predictions allows us to compare the urban or territorial reality with them, in order to identify discrepancies or imbalances that contribute to the correction of said variable and/or directly or indirectly the rectification of the affected urban law.

[1] Urban Mechanics is developed in the work under the title "Life and Death of the City. Right to the City and Urban Science", formulated by the author, which is also summarized in the article of the same title.
https://www.fernandovisedo.com/entropia-y-entalpia-urbana/

## 2. Methodology for Developing Urban Neural Networks

Prior to developing the neural network, a table is prepared with a set of 90 urban variables from 36 provincial capitals in Spain, which at this initial stage allows the construction of the urban big data.
The development of the different neural networks is carried out following the scripts specific to this technique. In this regard, it is necessary to clarify that in the proposed models of RNU8_2 and RNU1_15, the neurons are grouped into layers. The different layers can perform different transformations on their inputs, from the first layer, or input layer, to the last layer, or output layer, through the different hidden layers.
The RNU Algorithm follows the following structure:
2.1. Import the necessary libraries
The process begins by incorporating the necessary libraries to run the program in Colab or Visual Studio Code:

tensorflow
numpy
matplotlib.pyplot
tensorflowjs
ipywidgets
IPython.display
MinMaxScaler
Seaborn

2.2. Defining the Features and Targets
The data was obtained from the platforms of the National Institute of Statistics and the Ministry of Finance. Most of the urban data related to Economy, Society and Population, and Environmental and Spatial data were obtained from the former. The data from the Ministry of Finance are related to municipal budgets. This was supplemented with custom-generated data related to the active and inactive population to calculate n0/n1.

2.2.1. Features
The features are developed using the data from the prepared table. The input data are as follows.
It is possible to develop the features table outside the Colab algorithm environment or Visual Studio Code and link the execution to this table. Alternatively, the data matrix was imported into the model. To do this, the data was exported from xls to cvc and txt, taking care to separate decimals with periods and each data point in the row with commas.

2.2.2. Targets
The targets or output data for each of these cities were entered. The targets are the results that are intended to be predicted.

2.3. Normalization of Input Data
The data was normalized to reduce the possibility of error when interpreting the figures, in case there was a large disparity between them.

2.4. Defining the Neural Network Model
The number of input neurons and the number of output neurons are defined, as well as the number of hidden neurons. Furthermore, the activation function of the ReLU rectifier is defined.

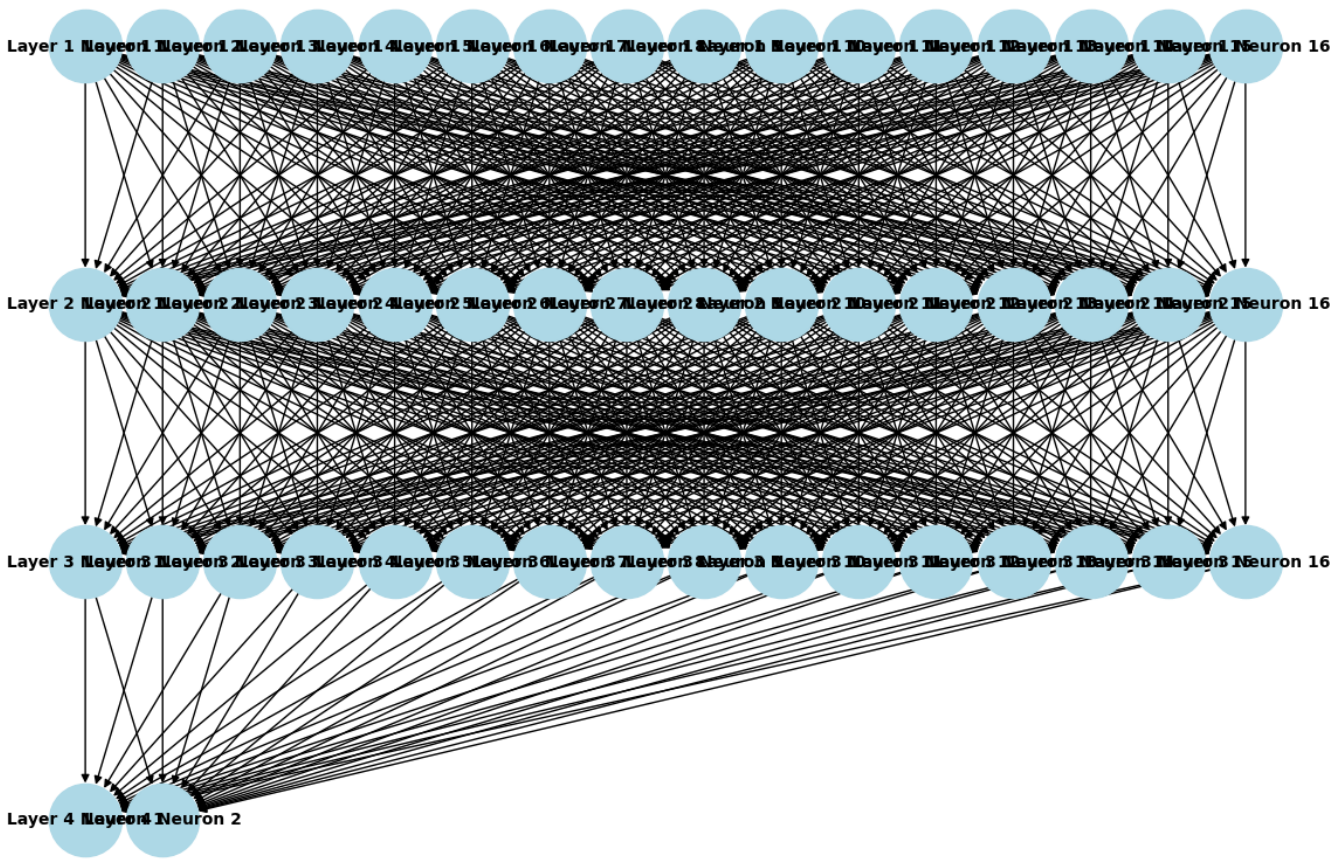

Layer 1 Neuron 1 … Layer 1 Neuron 16
Layer 2 Neuron 1 … Layer 2 Neuron 16
Layer 3 Neuron 1 … Layer 3 Neuron 16
Layer 4 Neuron 1
Layer 4 Neuron 2

*FIG. 1. Tested Urban Neural Network*
*Source: Author's own elaboration obtained from Algorithm RNU8_2 with 16 neurons in 3 hidden layers in Colab*

### 2.5. Compiling the Model

The ADAM Gradient Optimization Algorithm was used with an appropriate level of precision to reduce the probability of error without overlearning. RMSprop was also tested, but it did not improve prediction accuracy.

### 2.6. Training the Model

The model training was set to 1000 epochs, or learning cycles of the neural network. In both cases, increasing the number of learning cycles did not improve prediction accuracy and had the negative effect of consuming more system resources and increasing computation time.

### 2.7. Graphing the Magnitude of the Loss During Training

The training and validation graphs are plotted to illustrate the neural network's approach to the output values during its learning process.

### 2.8. Display the weights and biases of each layer

Weights represent the contribution to the final prediction of each of the relationships between the features or neurons of each layer of the neural network, from the input to the output predictions. Biases are bias neurons that contribute to the prediction, with a fixed input value of 1 and an associated weight. You have been asked to display the sequence of biases and weights for each learning epoch.

### 2.9. Save the complete model in an .h5 file

The .h5 file is responsible for storing the numerical, graphical, and text data hierarchically. The training data is stored in an .h5 file for use in a prediction script .html model with TensorFlow for internal use or for display in the browser after converting the .h5 file to .json and .bin using TensorFlowJS.

### 2.10. Convert and save the model in TensorFlow.js format

Saving and serialization are performed for sequential models with TensorFlow Keras in .json files.

### 2.11. Function to make predictions with the trained model

The task is to predict the outputs based on the inputs.

The function that relates the inputs to the output, where X is the vector of input features, $W(i)$ are the weights of the successive layers, and b(i) are the biases of the successive layers, is:

y=$W(3)$·ReLU ( $W(2)$)·ReLU ( $W(1)$X+$b(1)$) + $b(2)$ ) +$b(3)$

Each element of the function represents the following:

1. First hidden layer:
( $W(1)$X+$b(1)$) Calculates the linear combination of the input features with the weights $W(1)$ and biases $b(1)$
ReLU ( $W(1)$X+$b(1)$) Applies the ReLU activation function to the output of the linear combination, which introduces nonlinearity into the model. ReLU transforms negative values to zero and leaves positive values unchanged.

2. Second hidden layer:

( $W(2)$)·ReLU ( $W(1)$X+$b(1)$) + $b(2)$ ) Takes the output of the first layer (already transformed by ReLU) and applies another linear combination with the weights $W(2)$ and biases $b(2)$.

ReLU ( $W(2)$)·ReLU ( $W(1)$X+$b(1)$) + $b(2)$ ) Applies ReLU again to introduce more nonlinearity. 3.
Third Layer and Output Layer:
$W(3)$·ReLU ( $W(2)$)·ReLU ( $W(1)$X+$b(1)$) + $b(2)$ ) +$b(3)$ The third layer takes the transformed output of the second layer and applies a final linear combination with the weights $W(3)$ and biases $b(3)$ to obtain the final output y.

It is observed that the input features X are successively transformed through three hidden layers using linear combinations, biases, and the activation function ReLU, to finally produce an output y.

During training, the weights $W(1)$ of the first hidden layer, $W(2)$ of the second hidden layer, $W(3)$ of the output layer, and the biases $b(1)$, $b(2)$, $b(3)$ of the respective layers are adjusted using an optimization algorithm (gradient descent) to minimize the difference between the actual values and the predicted values.

2.12. Normalize the input data before predicting. In order to interpret the data correctly, it is necessary to "denormalize" it to recover the original units.

2.13. Display the network's internal calculations. Displaying the calculations is requested for verification purposes, although this operation is not necessary for prediction.

2.14. Calculate the prediction error (MSE). Displaying the prediction error calculated using the Mean Squared Error (MSE). This verification is essential to determine the probability of accuracy and verification of the prediction. The MSE must be greater than the figure considered appropriate for the calculation.

2.15. Create the input widgets for the data_prediction button_data output
For operational purposes, a table is created to input urban data for new municipalities in order to make predictions. In this case, all values are numeric. Activating the button will reset the system to make new predictions when urban data for new cities is included.

## 3. Urban Neural Networks RNU8_2

To develop this work, the following inputs, which describe the municipal urban system from the economic, social, and spatial perspectives, have been selected from among 90 variables:

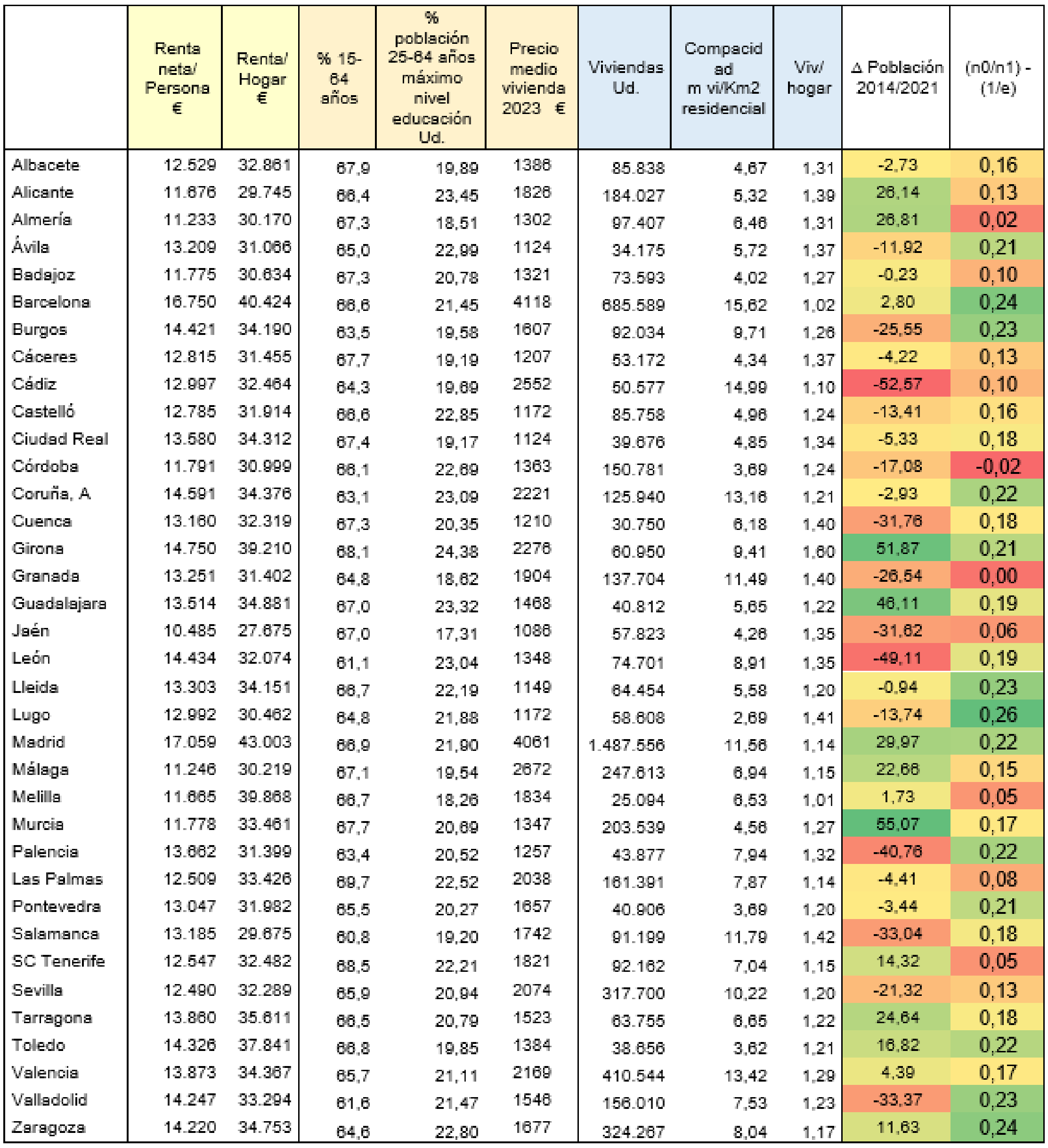
Net Income/Person
Income/Household
% of the working-age population between 15 and 64 years
Percentage of population aged 25-64 with the highest level of education
Average housing price in 2023

Number of dwellings
Compactness
Number of dwellings/Number of households

The selection of indicators was estimated for the sample of results from this exercise. The Urban Mechanics paper ("Life and Death of the City: Right to the City and Urban Science") proposes Principal Component Analysis to identify the fewest possible urban variables that best represent the urban system while losing the least amount of information. This analysis is beyond the scope of this exercise, and reference variables frequently used in urban analyses have been selected.

Furthermore, this study aims to connect with the aforementioned work to understand the city's vitality, that is, the value that reflects how close the city is to the "death" scenario, in which there is a high probability of the population abandoning the city due to the lack of incentives for citizen development with a high quality of life. It was shown that this scenario is related to the productivity of the active population n1 or rate of employed between 20 and 64 years and its relationship with the non-active population n0 or unemployment rate, in terms of urban entropy, figures obtained from INE whose sum is 100.

$$\frac{n0}{n1} = e^{-1}$$



| | Renta neta/ Persona € | Renta/ Hogar € | % 15-64 años | % población 25-64 años máximo nivel educación Ud. | Precio medio vivienda 2023 € | Viviendas Ud. | Compacidad m vi/Km2 residencial | Viv/ hogar | Δ Población 2014/2021 | (n0/n1) - (1/e) |
|---|---|---|---|---|---|---|---|---|---|---|
| Albacete | 12.529 | 32.861 | 67,9 | 19,89 | 1388 | 85.838 | 4,67 | 1,31 | -2,73 | 0,16 |
| Alicante | 11.676 | 29.745 | 66,4 | 23,45 | 1828 | 184.027 | 5,32 | 1,39 | 26,14 | 0,13 |
| Almería | 11.233 | 30.170 | 67,3 | 18,51 | 1302 | 97.407 | 6,46 | 1,31 | 26,81 | 0,02 |
| Ávila | 13.209 | 31.066 | 65,0 | 22,99 | 1124 | 34.175 | 5,72 | 1,37 | -11,92 | 0,21 |
| Badajoz | 11.775 | 30.634 | 67,3 | 20,78 | 1321 | 73.593 | 4,02 | 1,27 | -0,23 | 0,10 |
| Barcelona | 16.750 | 40.424 | 66,6 | 21,45 | 4118 | 685.589 | 15,62 | 1,02 | 2,80 | 0,24 |
| Burgos | 14.421 | 34.190 | 63,5 | 19,58 | 1607 | 92.034 | 9,71 | 1,26 | -25,55 | 0,23 |
| Cáceres | 12.815 | 31.455 | 67,7 | 19,19 | 1207 | 53.172 | 4,34 | 1,37 | -4,22 | 0,13 |
| Cádiz | 12.997 | 32.464 | 64,3 | 19,69 | 2552 | 50.577 | 14,99 | 1,10 | -52,57 | 0,10 |
| Castelló | 12.785 | 31.914 | 66,6 | 22,85 | 1172 | 85.758 | 4,96 | 1,24 | -13,41 | 0,16 |
| Ciudad Real | 13.580 | 34.312 | 67,4 | 19,17 | 1124 | 39.676 | 4,85 | 1,34 | -5,33 | 0,18 |
| Córdoba | 11.791 | 30.999 | 66,1 | 22,69 | 1363 | 150.781 | 3,69 | 1,24 | -17,08 | -0,02 |
| Coruña, A | 14.591 | 34.376 | 63,1 | 23,09 | 2221 | 125.940 | 13,16 | 1,21 | -2,93 | 0,22 |
| Cuenca | 13.160 | 32.319 | 67,3 | 20,35 | 1210 | 30.750 | 6,18 | 1,40 | -31,76 | 0,18 |
| Girona | 14.750 | 39.210 | 68,1 | 24,38 | 2276 | 60.950 | 9,41 | 1,60 | 51,87 | 0,21 |
| Granada | 13.251 | 31.402 | 64,8 | 18,62 | 1904 | 137.704 | 11,49 | 1,40 | -26,54 | 0,00 |
| Guadalajara | 13.514 | 34.881 | 67,0 | 23,32 | 1468 | 40.812 | 5,65 | 1,22 | 46,11 | 0,19 |
| Jaén | 10.485 | 27.675 | 67,0 | 17,31 | 1086 | 57.823 | 4,26 | 1,35 | -31,62 | 0,06 |
| León | 14.434 | 32.074 | 61,1 | 23,04 | 1348 | 74.701 | 8,91 | 1,35 | -49,11 | 0,19 |
| Lleida | 13.303 | 34.151 | 66,7 | 22,19 | 1149 | 64.454 | 5,58 | 1,20 | -0,94 | 0,23 |
| Lugo | 12.992 | 30.462 | 64,8 | 21,88 | 1172 | 58.608 | 2,69 | 1,41 | -13,74 | 0,26 |
| Madrid | 17.059 | 43.003 | 66,9 | 21,90 | 4061 | 1.487.556 | 11,56 | 1,14 | 29,97 | 0,22 |
| Málaga | 11.246 | 30.219 | 67,1 | 19,54 | 2672 | 247.613 | 6,94 | 1,15 | 22,66 | 0,15 |
| Melilla | 11.665 | 39.868 | 66,7 | 18,26 | 1834 | 25.094 | 6,53 | 1,01 | 1,73 | 0,05 |
| Murcia | 11.778 | 33.461 | 67,7 | 20,69 | 1347 | 203.539 | 4,56 | 1,27 | 55,07 | 0,17 |
| Palencia | 13.662 | 31.399 | 63,4 | 20,52 | 1257 | 43.877 | 7,94 | 1,32 | -40,76 | 0,22 |
| Las Palmas | 12.509 | 33.426 | 69,7 | 22,52 | 2038 | 161.391 | 7,87 | 1,14 | -4,41 | 0,08 |
| Pontevedra | 13.047 | 31.982 | 65,5 | 20,27 | 1657 | 40.906 | 3,69 | 1,20 | -3,44 | 0,21 |
| Salamanca | 13.185 | 29.675 | 60,8 | 19,20 | 1742 | 91.199 | 11,79 | 1,42 | -33,04 | 0,18 |
| SC Tenerife | 12.547 | 32.482 | 68,5 | 22,21 | 1821 | 92.162 | 7,04 | 1,15 | 14,32 | 0,05 |
| Sevilla | 12.490 | 32.289 | 65,9 | 20,94 | 2074 | 317.700 | 10,22 | 1,20 | -21,32 | 0,13 |
| Tarragona | 13.860 | 35.611 | 66,5 | 20,79 | 1523 | 63.755 | 6,65 | 1,22 | 24,64 | 0,18 |
| Toledo | 14.326 | 37.841 | 66,8 | 19,85 | 1384 | 38.656 | 3,62 | 1,21 | 16,82 | 0,22 |
| Valencia | 13.873 | 34.367 | 65,7 | 21,11 | 2169 | 410.544 | 13,42 | 1,29 | 4,39 | 0,17 |
| Valladolid | 14.247 | 33.294 | 61,6 | 21,47 | 1546 | 156.010 | 7,53 | 1,23 | -33,37 | 0,23 |
| Zaragoza | 14.220 | 34.753 | 64,6 | 22,80 | 1677 | 324.267 | 8,04 | 1,17 | 11,63 | 0,24 |

*FIG. 2. RNU 8_2 database: 36 Spanish cities.*
*Source: Prepared by the author using INE data*

The code is a complete implementation of a neural network in Python using TensorFlow to analyze a socioeconomic dataset. Each part of the code is explained in detail below, including the process described in the previous section: setup, data normalization, definition of the neural network model, training, visualization of the network structure and performance, and prediction of new data entered manually. The interpretation, with its fundamental parts and detailed explanations, is provided here.

1. Library Import and Environment Setup
The code begins by importing the necessary libraries.

```
!pip install tensorflow
!pip install tensorflowjs
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import tensorflowjs as tfjs
import ipywidgets as widgets
from IPython.display import display
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.utils import plot_model
import matplotlib.image as mpimg
import networkx as nx
```

The first lines install TensorFlow and TensorFlow.js to manage neural networks and make predictions in web applications. The other libraries are used for data manipulation, visualization, feature scaling, and graphical visualization of the neural network structure.

2. Definition of Input and Output Data (Features and Targets)
The following block contains two matrices, features and targets, which represent the independent variables (input data) and the target variables (output data), respectively.

```
features = np.array([
    (12.53, 32.86, 67.9, 19.9, 13.86, 85.84, 4.67, 1.31),
    ...])
targets = np.array([
    [-2.73, 0.16],
    ...])
```

These matrices contain socioeconomic and demographic data, where each row of features represents a set of independent values, and each row of targets represents two target output values for that particular input row.

3. Normalization of Input Data
To improve model performance, the input data (features) are normalized using MinMaxScaler, which brings all values to a range between 0 and 1.

```
scaler = MinMaxScaler()
features_normalized = scaler.fit_transform(features)
```

Normalization helps make the learning process more efficient, preventing the Urban Neural Network (UNN) from encountering problems when dealing with data of very large scales.
In the case under analysis, the input of values was performed using an approximation of values, a kind of pre-normalization, to facilitate the normalization process.

4. Definition of the UNN Model (Urban Neural Network)

The designed neural network contains three hidden layers, each with 64 neurons and an activation function (ReLU), and an output layer with two neurons to produce the two expected outputs for the targets. Other options with one and two hidden layers were tested, yielding results with worse predictions. The prediction is verified by calculating the difference between the output value and the actual value, which is related to the eight available city inputs. It is important to clarify that an exact match has a practically zero probability, since two cities with identical inputs can have different actual reference values for different outputs.

```
modelo = tf.keras.Sequential([
    tf.keras.layers.Dense(units=64, input_shape=[8], activation='relu'),  # Primera capa oculta
    tf.keras.layers.Dense(units=64, activation='relu'),  # Segunda capa oculta
    tf.keras.layers.Dense(units=64, activation='relu'),  # Tercera capa oculta
    tf.keras.layers.Dense(units=2)  # Capa de salida con dos neuronas])
```

5. Visualizing the Neural Network Structure
A function called draw_neural_network allows you to graphically visualize the neural network architecture using the networkx graphics library. This function creates an intuitive network graph, showing layers and connections between neurons in different layers.

```
def dibujar_red_neuronal(modelo):
    G = nx.DiGraph()
    capas = []
    for i, layer in enumerate(modelo.layers):
        capas.append([f'Layer {i+1} Neuron {j+1}' for j in range(layer.units)])
    ...
    nx.draw(G, pos, with_labels=True, node_size=2000, node_color='lightblue', font_size=10,
font_weight='bold', arrows=True)
    plt.title('Esquema de la Red Neuronal')
    plt.show()
```

6. Model Compilation and Training
The model is compiled with the Adam optimizer and the appropriate loss function for mean-squared-error regression problems.

```
modelo.compile(
    optimizer=tf.keras.optimizers.Adam(0.0001),
    loss='mean_squared_error' )
historial = modelo.fit(features_normalized, targets, epochs=1400, verbose=False)
```

The model is trained with 1400 learning epochs. It has been tested with 500 and 1500 cycles, with worse prediction results due to underlearning and overlearning, respectively.
The training data were normalized beforehand, and at the end of the training, the loss is plotted to visualize the model's progress.

```
plt.xlabel('#Época')
plt.ylabel('Magnitud de pérdida')
plt.plot(historial.history['loss'])
plt.show()
```

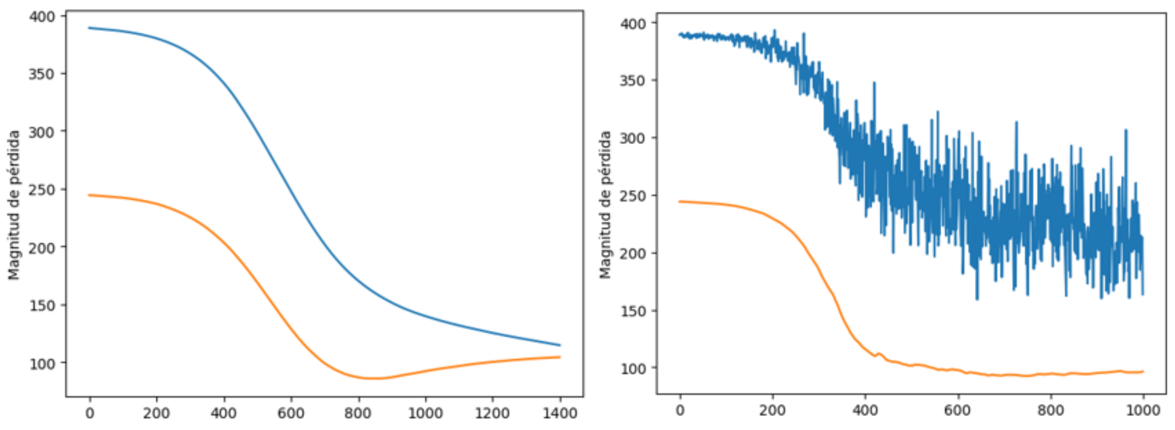


*FIG. 3. LEARNING CURVE (BLUE) AND VALIDATION (ORANGE) OF RNU8_2 WITHOUT DROPOUT AND WITH DROPOUT 0.3*
*SOURCE: AUTHOR'S OWN ELABORATION OBTAINED FROM RNU8_2 ALGORITHM WITH 16 NEURONS IN 3 HIDDEN LAYERS IN COLAB*

## 7. Extracting Weights and Biases from the Neural Network

The weights and biases are requested. The code prints the weights and biases that the model learned during training for each layer of the neural network.

```
for i, layer in enumerate(modelo.layers):
    pesos, sesgos = layer.get_weights()
    print(f"Capa {i+1}:")
    print(f"Pesos:\n{pesos}")
    print(f"Sesgos:\n{sesgos}")
```

It allows you to analyze the adjustment of the RNU's internal parameters to diagnose the predictions.

## 8. Save the Model in Keras and TensorFlow.js Formats

The trained model is saved in .h5 format for local use and in TensorFlow.js format for use in the web application. Obtaining this file is necessary for the tests performed with Visual Studio Code.

```
modelo.save('/content/model.h5')
tfjs.converters.save_keras_model(modelo, '/content/MODELOS/Urbana')
```

## 9. Prediction on New Manually Entered Data

The make_prediction function allows predictions to be made based on new update data after normalization.

```
def make_prediction(RentanetaPersonamE, RentaHogarmE, activa, edupoblacion64,
Preciomediovivienda2023mE, mViviendas, Compacidad, Numviv_Numhogares):
    input_data = np.array([[float(RentanetaPersonamE), float(RentaHogarmE), ...]])
    input_data_normalized = scaler.transform(input_data)
    prediction = modelo.predict(input_data_normalized)
    return prediction[0][0], prediction[0][1]
```

## 10. Calculation of the Prediction Error (MSE)

The code includes the calculate_error function, which calculates the Mean Squared Error (MSE) between the model's predictions and the actual values included in the targets.

```
def calcular_error():
    predicciones = modelo.predict(features_normalized)
    mse = np.mean(np.square(predicciones - targets))
    print(f"Error cuadrático medio (MSE): {mse}")
```

11. Graph of Predictions and Actual Values
The graph_predictions function allows you to visually compare the model's predictions with the actual values, creating a scatter plot for each target variable.

```
def graficar_predicciones():
    predicciones = modelo.predict(features_normalized)
    plt.figure(figsize=(10, 6))
    for i in range(targets.shape[1]):
        plt.subplot(1, 2, i + 1)
        plt.scatter(targets[:, i], predicciones[:, i], alpha=0.7)
        plt.plot([-100, 100], [-100, 100], 'r--')
    plt.tight_layout()
    plt.show()
```

12. User Input Widgets
Finally, the code creates interactive widgets that allow users to enter values for the different parameters in order to predict the result, which in this case is n0/n1. This set of widgets is useful for implementing the model in environments such as Visual Studio Code or Google Colab with a comprehensive interface, as shown in Fig. 4.

```
RentaPersonamE = widgets.FloatText(description='Renta neta/Persona m€', value=0.0)
...
button = widgets.Button(description="Hacer Predicción")
output = widgets.Output()

def on_button_click(b):
    with output:
        output.clear_output()
        result1,    result2    =    make_prediction(RentaPersonamE.value,    RentaHogarmE.value,
activa.value, ...)
        print(f'Predicción (Δ Poblacional): {result1}')
        print(f'Predicción ((n0/n1)-(1/e)): {result2}')

button.on_click(on_button_click)
display(RentaPersonamE,   RentaHogarmE,   activa,   edupoblacion64,   Preciomediovivienda2023mE,
mviviendas, Compacidad, Numviv_Numhogares, button, output)
```

It includes the prediction of the target outcome in the RNU (Real-Time Neural Network), allowing the input of new values to generate new predictions.

The described code generates a complete workflow for developing a neural network for prediction on socioeconomic, environmental, and spatial data, including the database, model generation and training, and the integration of the interactive graphical interface for making real-time predictions of expected outputs, in this case (n0/n1) – (1/e). This value reflects the urban system's proximity to the risk of its "death" or abandonment due to a lack of opportunities for the population. In this case, eight values have been identified that characterize the "vitality of the city." As the value increases between 0 and 1, the city's vitality also increases. A negative value indicates that the urban system has passed the point of recovery and requires an injection of extraordinary external resources for its survival. The urban variables related to this urban vitality indicator can be replaced by others deemed more appropriate, based on Urban Mechanics studies or the availability of values from other urban indicators.

Interpretation of the parameters of the 3 hidden layers 1, 2, and 3
Each of the three hidden layers has 16 neurons, each with a set of weights and an associated bias. The weights are the values that determine the influence of each input on each neuron in the layer. These weights are adjusted during model training to minimize the error between the prediction and the actual values. The biases help adjust the activation function, shifting it to activate or deactivate a given neuron.

Interpretation of the parameters of layer 4 (Output Layer)
In the output layer, each neuron represents an expected output of the model. In this case, 15 neurons are projected onto the output, corresponding to 15 outputs.

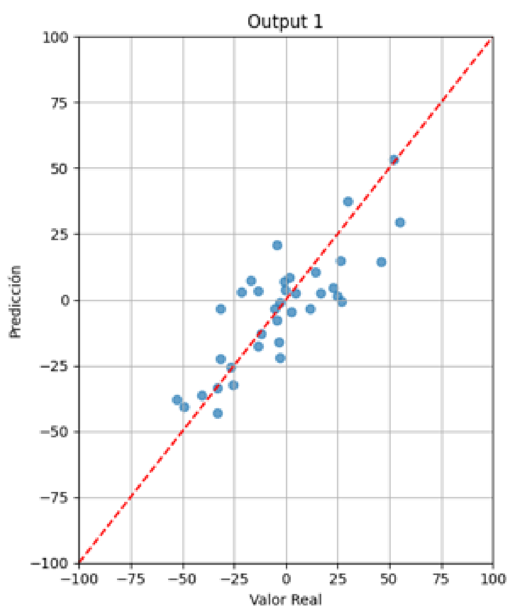
General Model Analysis
Although the model demonstrates sufficient learning, the extreme values of some weights suggest that certain neurons may be more influenced than others, validating the predicted relationships between input data and outputs. However, excessively high weight values indicate that the model is overfitted to certain patterns in the training data.

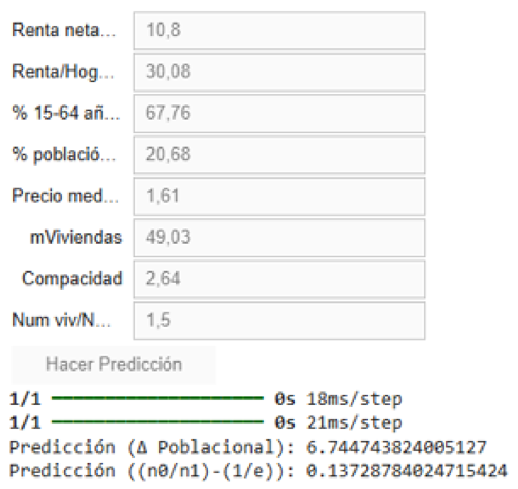
It has been observed that the validation curve diverges from the learning curve around cycle 600, at which point the prediction error remains high. It is possible to regularize the model by applying L2 or dropout techniques to reduce the dependence on extreme weights and improve the RNU's performance. The result of L2 regularization is to reduce the weight values to small values; this has not improved the magnitude of training loss (blue line), although it has improved validation (orange line). Regularization with dropout, which involves randomly removing neurons, initially shows a decrease in both lines up to 200 cycles, where training and validation loss decrease, indicating that the model is learning patterns in both the training and validation data. Subsequently, training loss begins to oscillate and shows a general increase instead of continuing to decrease due to significant fluctuations in the fit of the training data. Validation loss begins to increase after reaching a minimum due to overfitting to the training data, suggesting that the model does not generalize well to new data and is adapting to the values in the training data. On the other hand, the oscillations in training loss indicate a difficulty for the model in stabilizing its learning on the training data.

Quantitative Error Assessment
A Mean Squared Error (MSE) and $R^2$ close to 1 relative to the scale of the output data indicate that the model is making accurate predictions, while high MSE values and a low $R^2$ indicate problems with accuracy. The model has shown a Mean Squared Error (MSE) of 112.

If the output values are significantly higher than the MSE value, the error is acceptable. However, if the output values are in a lower range, the deviation between predictions and actual values may be excessive. If high errors are obtained for specific outputs, the model does not accurately capture the relationships for those particular outputs. In the analysis performed, the MSE is compared with a visual assessment of the error to determine the model's validity.

Visual Assessment of the Error: Comparing the actual and predicted values in the scatter plot shows that they align.





| | |
|---|---|
| Renta neta... | 10,8 |
| Renta/Hog... | 30,08 |
| % 15-64 añ... | 67,76 |
| % població... | 20,68 |
| Precio med... | 1,61 |
| mViviendas | 49,03 |
| Compacidad | 2,64 |
| Num viv/N... | 1,5 |

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Predicción (Δ Poblacional): 6.744743824005127
Predicción ((n0/n1)-(1/e)): 0.13728784024715424
```

*FIG. 4. Dispersion between actual input value and target prediction on the left and prediction interface widgets on the right*
*Source: Author's own elaboration obtained from Algorithm RNU8_2 with 16 neurons in 3 hidden layers in Colab*

The scatter plot shows the relationship between the actual values and the model's predictions for the Output. An upward trend is observed in the points scattered along the red dashed reference line, indicating that the model achieves a positive correlation between both variables.
The points are more concentrated around values close to zero and dispersed at the more extreme values because the distribution of the actual data is mostly close to zero. Dispersion is also observed at the most extreme values with respect to the reference line, although several points are close to it, which means that the model exhibits inaccuracy in the actual and predicted values furthest from the origin (approximately ±100).

The model may generate a prediction error that affects the MSE because some points deviate from the reference line. The MSE of 112 is moderate, so fit tests were performed using L2 regularization and dropout, but the results raise concerns about learning and validation.

Therefore, it can be concluded that the model offers accurate results, although it is possible to improve the prediction with additional adjustments, but above all with more training inputs that allow for better capture of variations. Unfortunately, the database is limited to 36 cities, which considerably reduces the learning capacity. The present model has established relationships between different inputs and outputs. Undoubtedly, it is possible to find other relationships that could improve the prediction, which represents a diverse and multidisciplinary avenue of research aimed at strengthening the tool for predicting the behavior of a city, an urban system, or a territorial system in order to evaluate different scenarios due to disturbances in each of the different urban variables. Nevertheless, the initial stages of this study will allow for the systematization of urban data collection by grouping by city range and location.

## 4. Urban Neural Networks RNU1_15

Now, a set of urban characteristics will be calculated from the value of a variable. The X and Y matrices represent the input data and the target (or output) values of the machine learning problem.

X is the one-dimensional matrix containing the model's input values for the vulnerability of the right to housing, using the number of households/number of dwellings ratio as a reference.

Y is the two-dimensional matrix where each row represents the attributes of each urban input, which correspond to the urban objectives or targets that the model aims to predict. In this case, these objectives or targets comprise 15 variables that model the city, the urban system, or the territorial system. The following variables obtained from the INE (National Institute of Statistics) have been selected:

n0/n1 City vitality n0=inactive population n1=active population
INVMUN M Municipal investment in
% AGR Percentage of agricultural land
% IND_COM_PUB_MIL_PRIV Percentage of industrial, commercial, public, military, and private land
SALARIO m€ Salary in thousands of €
HOG/VIV Ratio between number of households and number of dwellings
DPOB Population density in thousands of inhabitants per km² mhab/km²
% RESD Percentage of residential land
INVMA Municipal investment in the Environment in millions of € M€
% V_D_O Percentage of land
∆ POB14/21 Population increase between 2014 and 2021
% NAT Percentage of natural land
INVVU Municipal investment in urban green spaces in millions of € M€
DPOBTRESS Density Population density (thousands of inhabitants per km²)
COMPRES Residential density (thousands of dwellings per km²) m dwellings/km²res %INFT Percentage of land dedicated to communication infrastructure

Each of these variables corresponds to a value in the Urban Matrix formulated in Right to the City through urban entropy and enthalpy. In this matrix, the vulnerability input occupies position (2,2). The remaining cells are occupied by the variables listed above, following the order by rows i

$$(i,j) \quad \forall i, j \quad 1<i,j<4 \ \neq (2,2) = \text{input}$$

It is important to clarify that the variables chosen to represent the magnitude of the urban matrix are limited by the municipal data available from the INE (National Institute of Statistics), and therefore their representativeness will be limited in some cases, as will be discussed later.

1. Installation and Import of Libraries
As in the previous model, the libraries necessary for data analysis and visualization, as well as for building the neural network, are installed and imported.

```
# Importar las bibliotecas necesarias
!pip install ipywidgets tensorflow matplotlib numpy tensorflowjs
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import Input
import matplotlib.pyplot as plt
import seaborn as sns
from ipywidgets import widgets, Button
from IPython.display import display
```

This set of libraries allows you to:
Create and train neural network models (`tensorflow`, `keras`).
Work with arrays and perform mathematical calculations (`numpy`).
Visualize data and graphs (`matplotlib`, `seaborn`).

Build an interactive user interface (`ipywidgets`, `IPython.display`).

2. Defining Input Data and Target Values
The matrices of real values X and Y are defined, representing the input data and the values that the model must learn to predict.

```
# Definir los datos de entrada (2.2) Num viv/Num hogares
X = np.array([131, 139, 131, 137, 127, 126, 137, 110, 124, 134, 124, 121, 140, 160, 140, 122,
135, 135, 120, 141,
              115, 101, 127, 132, 114, 120, 142, 115, 120, 122, 121, 129, 123, 117])
# Definir los valores target
Y = np.array([  (21,13.04,75.60,1.39,9.48,0.15,1.37,0.05,0.26,-2.73,18.56,8.12,9.49,4.67,1.00),
(24,15.74,13.00,6.95,7.99,1.68,11.88,2.31,2.11,26.14,54.45,3.43,9.76,5.32,3.03),
…
(13,41.30,30.47,3.71,10.22,0.70,2.60,3.80,0.99,11.63,53.49,15.22,16.90,8.04,3.61)
])
```

The values in the 1x34 matrix X represent the ratio of dwellings to households in the urban system, serving as a coefficient to represent the vulnerability of the right to housing in this case. It is important to clarify that this easily obtainable coefficient has been adopted, but the number of housing applicants could be used as a reference, offering a better measure of the vulnerability of this urban right. This data is published by some regional administrations with municipal status, but it could not be included because many of the cities in the database do not have this data available on open platforms.

| | n0/n1 | Total gastos de inversión M€ | Uso del suelo (%): Zonas agrícolas | Uso del suelo (%): Unidades industriales, comerciales, públicas militares y priva | Salario m€ | Num viv/Num hogares | Densidad población (hab/km2) | Uso del suelo (%): Tejido urbano residencial discontinuo | Medio ambiente M€ | Uso del suelo (%): Zonas verdes urbanas, instalaciones deportivas y de ocio | Δ Poblacional 2014/2021 | Uso del suelo (%): Zonas naturales | Vivienda y urbanismo M€ | Densidad población tejido residencial (m hab/km2) | Compacidad (m viv/Km2 residencial) | Uso del suelo (%): Infraestructuras de transporte |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Albacete | 0,21 | 13,04 | 75,60 | 1,39 | 9,48 | 1,31 | 152,7 | 1,37 | 0,05 | 0,26 | -2,73 | 18,56 | 8,12 | 9,49 | 4,67 | 1,00 |
| Alicante | 0,24 | 15,74 | 13,00 | 6,95 | 7,99 | 1,39 | 1.676,5 | 11,88 | 2,31 | 2,11 | 26,14 | 54,45 | 3,43 | 9,76 | 5,32 | 3,03 |
| Almería | 0,34 | 11,53 | 11,05 | 1,37 | 8,03 | 1,31 | 679,5 | 2,48 | 0,45 | 0,80 | 26,81 | 77,15 | 1,54 | 13,35 | 6,46 | 2,38 |
| Ávila | 0,16 | 11,68 | 9,46 | 1,92 | 9,57 | 1,37 | 253,0 | 1,24 | 0,11 | 0,77 | -11,92 | 81,02 | 4,13 | 9,77 | 5,72 | 1,85 |
| Badajoz | 0,27 | 7,83 | 53,95 | 1,15 | 8,49 | 1,27 | 104,8 | 0,86 | 1,26 | 0,28 | -0,23 | 41,12 | 3,76 | 8,25 | 4,02 | 0,83 |
| Barcelona | 0,13 | 488,90 | 0,20 | 15,64 | 12,24 | 1,02 | 16.256,5 | 6,47 | 38,06 | 9,95 | 2,80 | 20,48 | 382,78 | 37,91 | 15,62 | 9,69 |
| Burgos | 0,14 | 18,34 | 44,91 | 13,56 | 10,15 | 1,26 | 1.644,6 | 3,50 | 0,02 | 2,57 | -25,55 | 19,55 | 6,56 | 18,60 | 9,71 | 5,01 |
| Cáceres | 0,24 | 3,48 | 13,15 | 0,29 | 9,40 | 1,37 | 55,0 | 0,42 | 1,13 | 0,16 | -4,22 | 83,85 | 0,34 | 7,85 | 4,34 | 0,78 |
| Cádiz | 0,26 | 13,42 | 0,00 | 14,59 | 7,92 | 1,10 | 9.408,2 | 0,45 | 0,30 | 2,69 | -52,57 | 40,69 | 5,85 | 34,22 | 14,99 | 14,17 |
| Castelló | 0,20 | 16,64 | 36,76 | 10,14 | 9,24 | 1,24 | 1.599,9 | 10,55 | 0,20 | 1,37 | -13,41 | 29,67 | 6,21 | 10,08 | 4,96 | 4,60 |
| Ciudad Real | 0,19 | 0,94 | 64,50 | 1,32 | 10,43 | 1,34 | 264,9 | 2,50 | 0,11 | 0,52 | -5,33 | 24,54 | 0,35 | 9,23 | 4,85 | 2,39 |
| Córdoba | 0,38 | 13,66 | 67,11 | 1,22 | 8,12 | 1,24 | 258,2 | 2,24 | 0,63 | 0,31 | -17,08 | 25,77 | 2,50 | 7,97 | 3,69 | 1,24 |
| Coruña, A | 0,15 | 24,37 | 6,32 | 16,46 | 10,06 | 1,21 | 6.541,7 | 6,35 | 1,48 | 5,86 | -2,93 | 33,55 | 8,36 | 25,87 | 13,16 | 6,69 |
| Cuenca | 0,19 | 4,83 | 9,93 | 0,48 | 9,57 | 1,40 | 59,3 | 0,50 | 0,82 | 0,11 | -31,76 | 87,15 | 1,69 | 10,98 | 6,18 | 0,58 |
| Girona | 0,15 | 5,50 | 10,65 | 7,30 | 11,20 | 1,60 | 2.648,4 | 8,38 | 0,15 | 4,72 | 51,87 | 55,07 | 3,10 | 15,95 | 9,41 | 3,57 |
| Granada | 0,37 | 1,96 | 32,45 | 7,02 | 8,62 | 1,40 | 2.653,3 | 7,05 | 0,01 | 2,31 | -26,54 | 36,58 | 0,17 | 19,50 | 11,49 | 3,45 |
| Guadalajara | 0,18 | 2,50 | 57,48 | 1,69 | 10,46 | 1,22 | 371,9 | 2,81 | 0,25 | 0,56 | 46,11 | 33,16 | 0,97 | 12,11 | 5,65 | 1,72 |
| Jaén | 0,31 | 4,65 | 66,76 | 1,95 | 6,52 | 1,35 | 265,9 | 1,08 | 0,69 | 0,39 | -31,62 | 22,67 | 1,22 | 8,31 | 4,26 | 1,73 |
| León | 0,18 | 11,13 | 9,21 | 11,71 | 9,11 | 1,35 | 3.178,6 | 9,02 | 0,69 | 4,71 | -49,11 | 39,29 | 4,09 | 14,80 | 8,91 | 5,92 |
| Lleida | 0,14 | 3,65 | 73,20 | 4,20 | 9,70 | 1,20 | 662,2 | 3,91 | 0,15 | 1,55 | -0,94 | 10,19 | 0,14 | 12,15 | 5,58 | 3,33 |
| Lugo | 0,11 | 10,43 | 30,56 | 1,43 | 9,09 | 1,41 | 298,6 | 4,16 | 1,04 | 0,53 | -13,74 | 55,84 | 0,90 | 4,52 | 2,69 | 3,70 |
| Madrid | 0,15 | 310,10 | 4,62 | 8,32 | 13,22 | 1,14 | 5.512,9 | 12,30 | 28,64 | 10,87 | 29,97 | 36,88 | 99,36 | 25,91 | 11,56 | 9,57 |
| Málaga | 0,22 | 15,06 | 22,17 | 4,26 | 7,49 | 1,15 | 1.463,5 | 4,66 | 1,41 | 1,31 | 22,66 | 54,55 | 2,71 | 16,21 | 6,94 | 5,08 |
| Melilla | 0,32 | 28,86 | 1,74 | 16,40 | 8,40 | 1,01 | 6.571,8 | 7,58 | 10,44 | 8,30 | 1,73 | 33,26 | 8,28 | 22,66 | 6,53 | 5,34 |
| Murcia | 0,19 | 24,38 | 47,44 | 2,29 | 8,80 | 1,27 | 518,7 | 3,58 | 1,24 | 0,72 | 55,07 | 38,81 | 6,20 | 10,29 | 4,56 | 2,07 |
| Palencia | 0,15 | 10,70 | 51,71 | 4,20 | 9,06 | 1,32 | 823,0 | 3,17 | 0,00 | 1,37 | -40,76 | 34,25 | 7,85 | 14,14 | 7,94 | 1,51 |
| Palmas, Las | 0,28 | 44,57 | 4,02 | 6,83 | 8,53 | 1,14 | 3.743,7 | 8,92 | 1,84 | 3,64 | -4,41 | 53,92 | 15,88 | 18,59 | 7,87 | 6,14 |
| Pontevedra | 0,16 | 10,39 | 18,59 | 2,62 | 9,26 | 1,20 | 706,2 | 6,38 | 2,19 | 1,26 | -3,44 | 62,44 | 3,22 | 7,50 | 3,69 | 4,17 |
| Salamanca | 0,18 | 15,50 | 20,34 | 11,99 | 8,07 | 1,42 | 3.666,5 | 9,93 | 3,16 | 7,02 | -33,04 | 29,38 | 4,41 | 18,73 | 11,79 | 6,49 |
| Santa Cruz de Tenerife | 0,32 | 39,27 | 1,78 | 2,88 | 8,44 | 1,15 | 1.394,3 | 5,32 | 3,97 | 1,01 | 14,32 | 82,25 | 7,46 | 15,99 | 7,04 | 2,26 |
| Sevilla | 0,24 | 36,99 | 27,87 | 13,84 | 8,61 | 1,20 | 4.892,1 | 3,70 | 0,99 | 9,41 | -21,32 | 12,06 | 3,05 | 22,25 | 10,22 | 8,88 |
| Tarragona | 0,19 | 11,10 | 12,48 | 10,96 | 10,25 | 1,22 | 2.436,1 | 10,64 | 0,28 | 4,62 | 24,64 | 44,26 | 1,37 | 14,24 | 6,65 | 8,75 |
| Toledo | 0,15 | 6,81 | 42,22 | 2,74 | 11,34 | 1,21 | 370,8 | 4,29 | 2,38 | 0,99 | 16,82 | 42,00 | 0,85 | 8,04 | 3,62 | 2,57 |
| Valencia | 0,19 | 81,33 | 28,14 | 8,15 | 9,57 | 1,29 | 5.872,3 | 5,28 | 2,62 | 4,07 | 4,39 | 30,70 | 10,49 | 26,16 | 13,42 | 5,73 |
| Valladolid | 0,14 | 32,79 | 45,17 | 8,52 | 9,55 | 1,23 | 1,6 | 5,73 | 7,35 | 3,07 | -33,37 | 26,37 | 12,09 | 14,44 | 7,53 | 3,00 |
| Zaragoza | 0,13 | 41,30 | 30,47 | 3,71 | 10,22 | 1,17 | 701,3 | 2,60 | 3,80 | 0,99 | 11,63 | 53,49 | 15,22 | 16,90 | 8,04 | 3,61 |
| | 1.1 | 1.2 | 1.3 | 1.4 | 2.1 | 2.2 | 2.3 | 2.4 | 3.1 | 3.2 | 3.3 | 3.4 | 4.1 | 4.2 | 4.3 | 4.4 |

*FIG. 5. Base de datos RNU 1_15 36 ciudades España*
*Fuente: Elaboración propia consultando INE*

The values in the 34x15 matrix Y represent the 15 actual values in the columns that form the urban matrix for each of the 34 cities in the rows. These 15 values are the targets to be predicted. It is important to clarify that, unlike the previous model which included data from 36 cities, the data for Madrid and

Barcelona have been removed here because they cause prediction dispersion problems due to their very different data sets compared to the rest of the cities. This affects the prenormalization process used to improve model learning.

3. Data Normalization

Normalization transforms the data into values between 0 and 1 to facilitate model training, as neural network algorithms perform better with normalized data.

```
# Normalizar los datos de entrada y salida
X_max, Y_max = np.max(X), np.max(Y)
X_normalized, Y_normalized = X / X_max, Y / Y_max
```

Normalization involves calculating the maximum values in X and Y and dividing each value by these maximums, generating X_normalized and Y_normalized, which are normalized versions of the input and output data.

4. Architecture of the Urban Neural Network

The neural network model is defined using the Keras Sequential class, with three hidden layers and one output layer containing 15 neurons, corresponding to the 15 values in each row of Y.

```
# Red neuronal ajustada
model = Sequential([
    Input(shape=(1,)),
    Dense(64, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    Dense(15, activation='linear')
])
```

The first Input layer indicates that the model expects a single, one-dimensional input. Five Dense layers of 128 and 64 neurons are included, each with linear relu activation, allowing the generation of any real-world value. While L2 regularization can be applied, it has been shown that in this case, its contribution to reducing overfitting by penalizing neuron weights is unnecessary.
The output layer, Dense15, activation=linear, has 15 neurons, each representing an output value for the 15 elements in the Y matrix.

5. Model Compilation and Training
The model is trained using the Adam optimizer, with a low learning rate (0.00001) and the mean squared error loss function.

```
# Compilación y entrenamiento con ajuste en tasa de aprendizaje y optimizador Adam
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
loss='mean_squared_error')
historial = model.fit(X_normalized, Y_normalized, epochs=1000, verbose=1, validation_split=0.2)
```

The model is trained for 1000 cycles, reserving 20% of the data for validation. Feedback is applied in each cycle during training to adjust the network weights and minimize prediction error.

6. Visualization of Training Loss
A script is added to graph the loss over time in order to evaluate model convergence or overfitting issues. This check is essential to determine the model's accuracy, as observed in the previous model, which exhibited validation problems during training. However, in this case, the model performed better.

```
# Gráfica de pérdida ajustada
plt.xlabel('#Época')
plt.ylabel('Pérdida')
plt.plot(historial.history['loss'], label='Entrenamiento')
plt.plot(historial.history['val_loss'], label='Validación')
plt.legend()
plt.show()
```

The loss is plotted on both the training and validation sets, allowing observation of whether the model consistently improves or is overfitted.

7. Calculation of the Mean Squared Error (MSE)
The mean squared error is calculated as a benchmark value that reflects the confidence in the predictive model.

```
# Calcular el Error Cuadrático Medio (MSE)
predicciones = model.predict(X_normalized) * Y_max  # Escalar las predicciones a los valores originales
mse = np.mean(np.square(Y - predicciones))  # Calcular el MSE
print(f"Error cuadrático medio (MSE): {mse}")
```

8. Prediction and Error Calculation Functions
Functions are created to predict a given value (predict) and to calculate the absolute error between the actual and predicted values (calculate_error).

```
def predecir(value):
    value_normalized = np.array([value]) / X_max
    prediction = model.predict(value_normalized)
    return prediction * Y_max

def calcular_error(real_target, prediccion_target):
    return np.abs(real_target - prediccion_target)
```

The `predict` function takes an input value, normalizes it, and obtains the prediction by running the model. This prediction is then denormalized to obtain the predicted value on its original scale by multiplying by Y_max. The `calculate_error` function provides the absolute error, as the absolute difference between the actual and predicted values.

9. Calculation of Absolute Error and Error Graph
The absolute errors between the predictions and actual values are calculated for each output.

```
# Obtener las predicciones y los valores reales
predicciones = model.predict(X_normalized) * Y_max
valores_reales = Y

# Calcular el error absoluto entre cada valor real y predicción
errores = calcular_error(valores_reales, predicciones)

# Gráfica de error para cada output
plt.figure(figsize=(12, 8))
for i in range(15):
    plt.subplot(3, 5, i + 1)
    plt.bar(range(len(errores)), errores[:, i], color='salmon')
    plt.xlabel('Muestra')
    plt.ylabel(f'Error Output {i+1}')
    plt.title(f'Error en Output {i+1}')
plt.tight_layout()
plt.show()
```

Each subplot displays the absolute error for each of the 15 outputs.

10. Creating a 4x4 Prediction Table
A function is created that generates a 4x4 table with predictions and displays it using Seaborn. It's important to remember that the 4x4 table is the urban matrix composed of the output or target values resulting from the prediction.

```
def crear_tabla_4x4(value, prediction):
    tabla_4x4 = np.zeros((4, 4))
    tabla_4x4[1, 1] = value
    tabla_4x4[0, :] = prediction[0][:4]
    tabla_4x4[1, [0, 2, 3]] = prediction[0][4:7]
    tabla_4x4[2, :] = prediction[0][7:11]
    tabla_4x4[3, :] = prediction[0][11:]
    return tabla_4x4

names = [
    ["n0/n1", "INVMUN M", "% AGR", "% IND_COM_PUB_MIL_PRIV"],
    ["SALARIO m€", "viv/hog", "DPOB mhab/km2", "% RESD"],
    ["INVMA M€", "% V_D_O", "∆ POB14/21", "% NAT"],
    ["INVVU M€", "DPOBTRES m hab/km2", "COMPRES m viv/Km2res", "%INFT"]
]
def mostrar_tabla_4x4(tabla_4x4, titulo="Tabla de Predicción 4x4"):
    plt.figure(figsize=(10, 10))
    sns.heatmap(tabla_4x4, annot=[[f"{names[i][j]}\n{tabla_4x4[i][j]:.2f}" for j in range(4)]
for i in range(4)],
                fmt="",    cmap='coolwarm',    cbar=True,    linewidths=1,    linecolor='black',
xticklabels=[4, 3, 2, 1],
                yticklabels=[4, 3, 2, 1])
    plt.title(titulo)
    plt.show()
```

Figure 6 shows that the input is located in box 2.2 of the Urban Matrix corresponding to social vulnerability of urban rights.

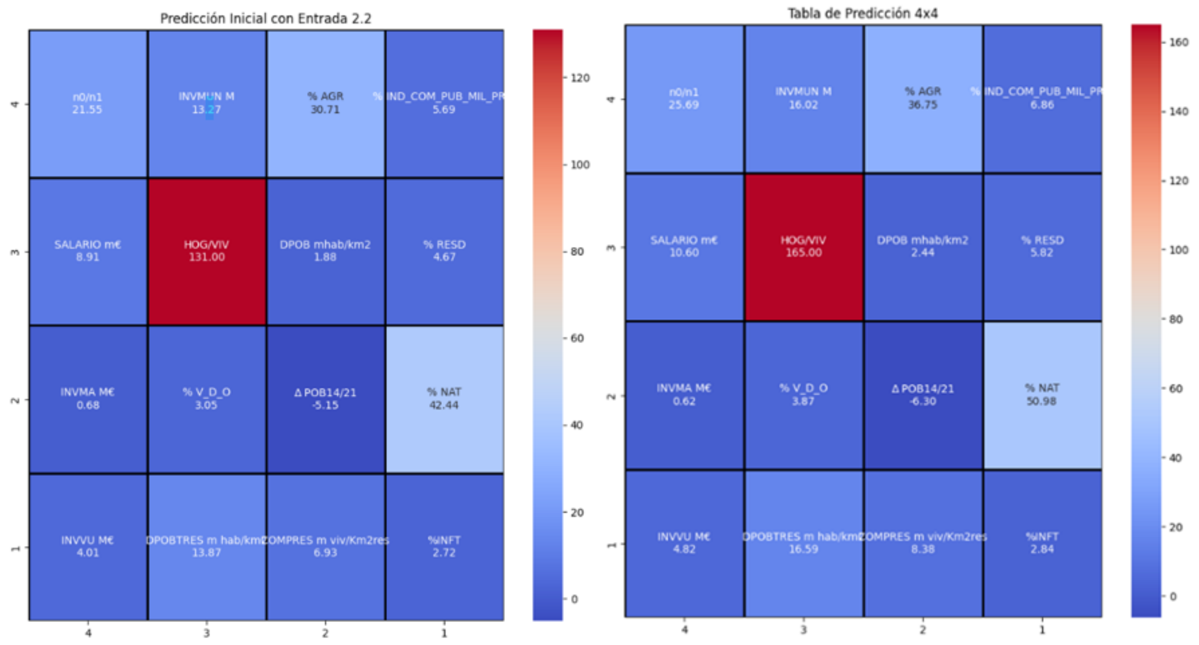


*FIG. 6. 4x4 prediction table with initial value and interactive value using widgets.*

*Source: Author's own work, obtained from Algorithm RNU1_15 in Colab*

## 11. Interactive Interface with Widgets

To make real-time predictions, interactive widgets from ipywidgets are used, allowing you to adjust input values and update the prediction and visualization of the 4x4 table in real time.

```
nuevo_valor_slider = widgets.FloatSlider(min=0, max=200, step=1, value=131, description
="Valor de entrada")
button = widgets.Button(description="Calcular predicción")

def on_button_clicked(b):
    value = nuevo_valor_slider.value
    prediction = predecir(value)
    tabla_4x4 = crear_tabla_4x4(value, prediction)
    mostrar_tabla_4x4(tabla_4x4)

button.on_click(on_button_clicked)
display(nuevo_valor_slider, button)
```

The interface in this case is presented as a bar that allows the entry of input values between 0 and 200, corresponding to actual values between 0 and 0.2, as shown in Fig. 10.

## 12. Final Scatter Plot

Finally, a scatter plot of the actual values versus the predicted values for the 15 outputs is created, allowing for a visual evaluation of the model's accuracy.

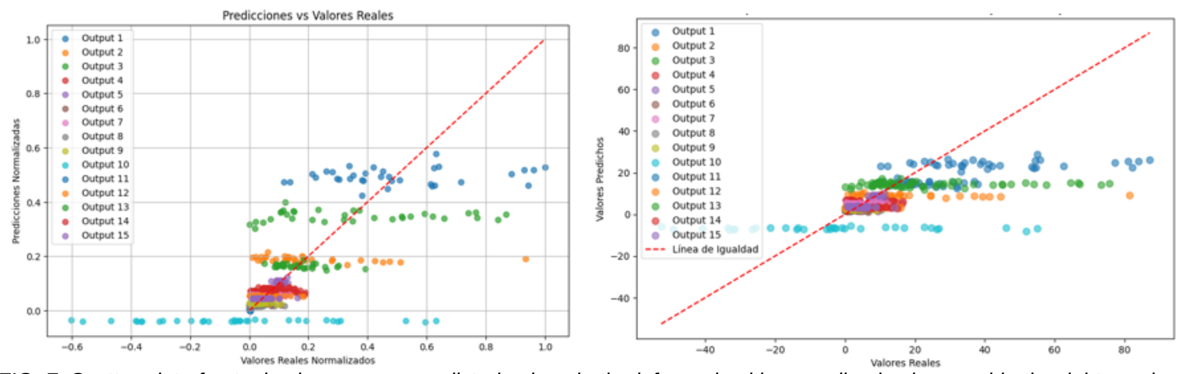


*FIG. 7. Scatter plot of actual values versus predicted values in the left graph with normalized values and in the right graph with actual values.*
*Source: Author's own elaboration obtained from Algorithm RNU1_15 with 3 hidden layers in Colab*

Figure 6 shows the error distribution in the outputs. Each output corresponds to a set of 15 real values Y and the values predicted by the neural network. The error for each output is reviewed independently, revealing that some outputs show highly accurate predictions, while others exhibit some error. The diagonal line of equality, originating at (0, 0) and extending to the maximum point of the output range, represents the equivalence between real and predicted values. Closer proximity to the line of equality indicates that the Urban Neural Network contains negligible prediction errors. Conversely, a distance from the line of equality indicates a significant prediction error for the Urban Neural Network.

The graph shows a dispersion between the actual and predicted values of the neural network for some outputs, especially in higher or lower ranges that exhibit a dispersion far from the line of equality. This is despite the model having achieved a final training loss of 0.0592 and a validation loss of 0.0616, although the model appears to have achieved good convergence in terms of loss. It is not considered necessary to check for errors related to overfitting or underfitting with L2, dropout, or early stopping, since a satisfactory decrease in loss is observed during both training and validation as training progresses.

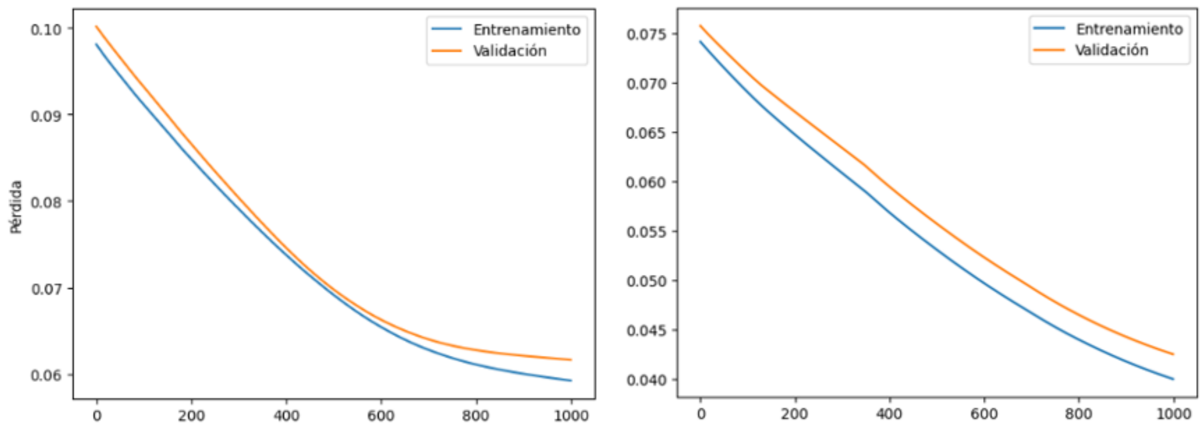


*FIG. 8. Training loss and validation loss*
*Source: Author's own elaboration obtained from Algorithm RNU1_15 with 3 hidden layers in Colab*

Error analysis for each output allows for the identification of areas where the model needs improvement.

However, tests related to hyperparameter tuning have also been performed, adjusting the learning rate to a lower level or dynamic adjustment during training (learning rate decay) to improve convergence. Different layer and neuron configurations have also been added. It is not possible, however, to increase the dataset size, which could substantially improve the model's performance. Conversely, the values for Madrid and Barcelona have been removed as they are considered outliers.

## 5. Application of neural networks to predict the urban vitality of El Puerto de Santa María.

The draft of the General Urban Development Plan (PGOM) for El Puerto de Santa María analyzes four alternatives, the most significant difference between which lies in land use.

In the analyses conducted to identify the best urban development option through Strategic Environmental Assessment, Alternative 3 was selected from an environmental perspective, based on quantitative assessment methods. It clearly achieved a higher score than the other three alternatives due to its positive performance across most key environmental aspects.

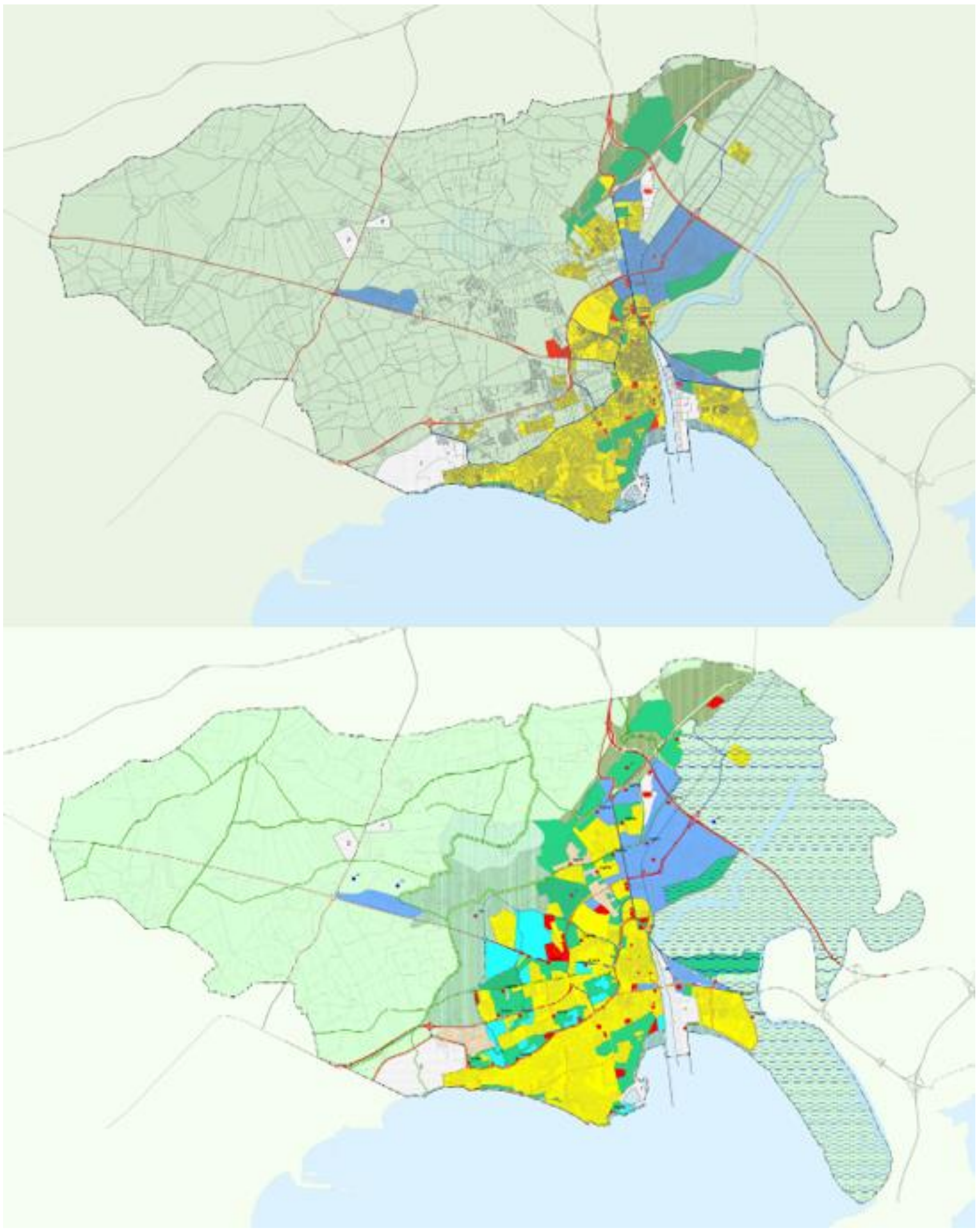


*FIG. 9. Alternatives A0 and A3 of the Preliminary General Urban Development Plan of El Puerto de Santa María (2023)*
*Source: Author's own elaboration*

One of the most noteworthy aspects considered in this assessment is the restructuring of the existing city, since the principle guiding Alternative 3 is urban development through near-zero growth, thereby minimizing the occupation of new land. For this reason, it is more environmentally favorable than Alternative 2, as it proposes a smaller area of new urban development and does not include infrastructure that has subsequently been deemed unnecessary, opting instead for the expansion and improvement of existing roads.

A similar situation exists with respect to Alternatives 0 and 1. Alternative 0 involves leaving the current situation without any urban planning intervention, maintaining the irregularly dispersed housing of more than 5,200 homes spread across the western periphery of the urban area (shown in yellow). Alternative 1 is equivalent to the annulled 2012 General Urban Development Plan (PGOU) due to the failure to carry out a Strategic Environmental Assessment. This alternative proposes expanding the urban land area to encompass all irregular settlements, creating a grid pattern over the green areas in the western part of the city. It treats all the land homogeneously, prioritizing urban sprawl over the reuse and integration of the existing city. This is crucial considering the situation of some neighborhoods in the urban center of El Puerto de Santa María, which require rehabilitation, particularly regarding housing.

Additionally, urban neural network methodology was applied to predict the urban vitality of the proposed alternatives A0 and A3 of the El Puerto de Santa María General Urban Development Plan (PGOM) Preliminary Document.

The urban vitality analysis was restricted to considering only the increase in housing and the density of thousands of dwellings per square kilometer. Alternative 3 includes irregular housing, while Alternative 0 maintains that irregular housing is illegally spread across the rural land. The remaining parameters remain constant in this exercise, as shown in the following table.

| INPUT | | A3 | A0 |
|---|---|---|---|
| Net Income/Person | m€ | 10,83 | 10,83 |
| Income/Household (€) | m€ | 30,08 | 30,08 |
| Population 15-64 years | % | 67,76 | 67,76 |
| Population 25-64 years (University Graduates) | % | 20,68 | 20,68 |
| Average Housing Price 2023 | m€/m2 | 1,614 | 1,614 |
| Housing | mviv | 49,03 | 43,83 |
| Compactness | mviv/km2 res | 2,14 | 2,64 |
| Number of Housing Units/Number of Households | | 1,5 | 1,3 |

The values altered in this analysis are considered to be related to housing. The number of dwellings, 5,200, increases upon achieving regularization, with the corresponding urban or suburban amenities for each land classification, which in any case guarantee their safety, health, and habitability. On the other hand, by establishing delimited urban and suburban areas, the surface area of planned residential land decreases, thus increasing density. Finally, the increase in the number of legal and regular dwellings, which rises by 5,200 units, allows for greater availability for households. The proportion of available dwellings per household should be analyzed and compared with the number of 4,409 vacant dwellings in El Puerto de Santa María, which represents 9% of the total dwellings according to the INE (National Institute of Statistics).

Using this hidden 3-layer neural network, the value (n1/n0) - (1/e) has been calculated, which indicates urban vitality. The results show that Alternative A3 maintains a distance of 13.7 (0.23) points, and Alternative A0 maintains a distance of 12.9 (0.24) points. These figures are equivalent to unemployment rates of 18% and 19%, respectively.

The analysis yields an urban vitality value of (n0/n1) - (1/e) = 0.137, so n0/n1 equals 0.24. This ratio is equivalent to an unemployment rate of 18%, which places El Puerto de Santa María in a precarious situation regarding development in future scenarios unless there is external investment. This prediction is found to be consistent with reality, as the unemployment rate reached 18% in 2022, the year in which the calculation was made.

Therefore, it is considered advisable to implement urban land planning by regularizing and legalizing the 5,200 currently irregular dwellings.

Next, the indicators of the 4x4 Urban Matrix are evaluated, incorporating the impact value of 1.5 dwellings per household, which amounts to 1.50. This situation represents a low level of vulnerability, as there is a high proportion of dwellings per household compared to other cities. This coincides with the 578 housing applicants, representing 0.6% of the population, which differs substantially from the 4,752 in Cádiz, representing 4.2% of the population.

The execution of the model offers a predicted scenario of the Urban Matrix resulting from applying the behavior of the set of medium-sized Spanish cities in the year 2022 to El Puerto de Santa María in a scenario of moderate housing vulnerability of 1.5 homes per household.

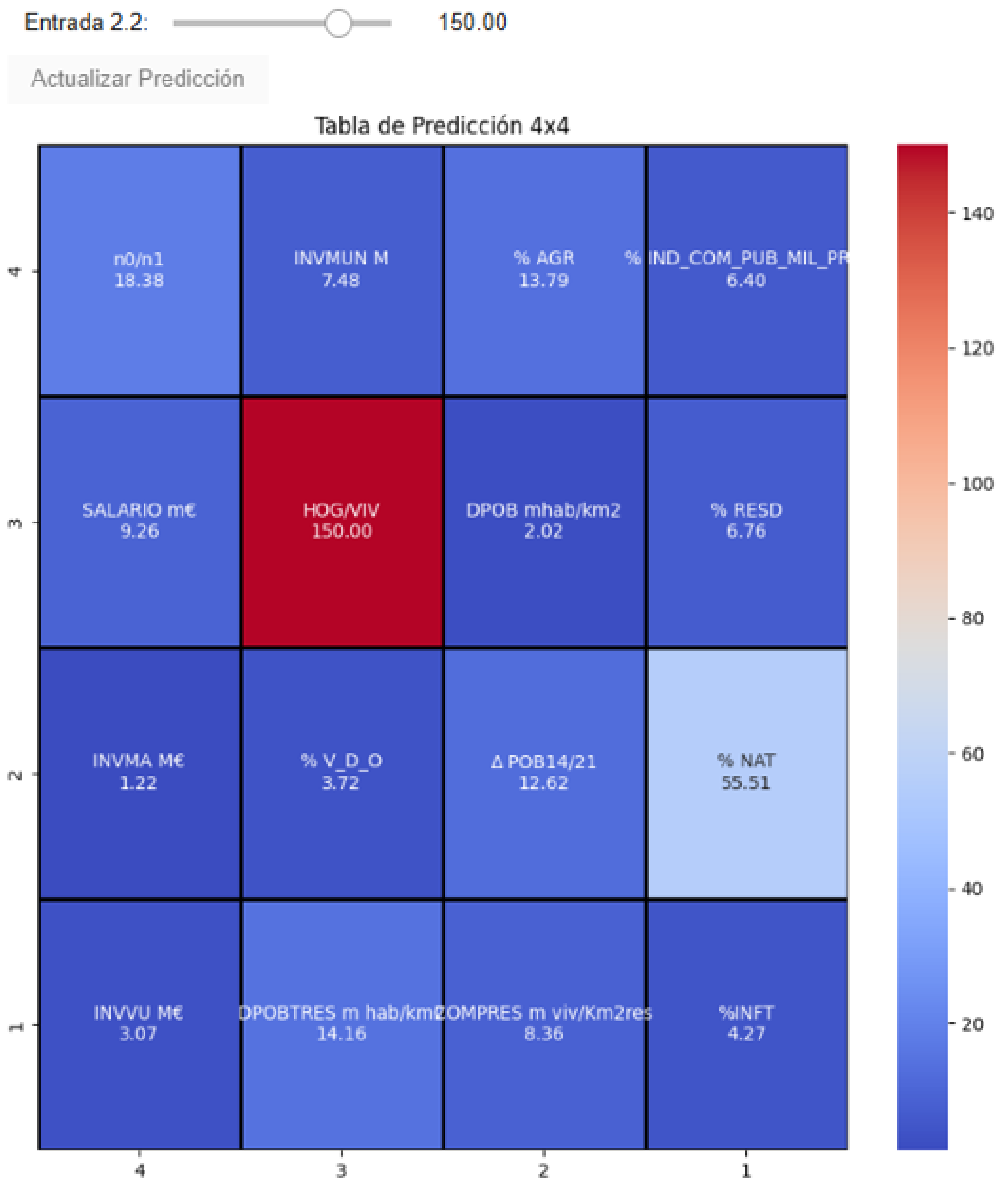


*FIG. 10. Predicted values for the Urban Matrix of El Puerto de Santa María (2023)*
*Source: Prepared by the author using Algorithm RNU1_15 with 3 hidden layers in Colab*

This data can be classified into the following sections

| | |
|---|---|
| Modelo | Datos propios calculados a partir de datos existentes del sistema urbano |
| Política | Datos derivados de decisiones políticas municipales |
| Territorio | Datos derivados de la geografía del sistema urbano |
| Cultura | Datos del sistema urbano basados en la tradición e historia municipal |
| Cultura_Política | Datos procedentes de los hábitos y costumbres, así como de decisiones políticas |

The section on proprietary data is considered representative of the element of the urban matrix that models the urban system. The policy section represents the municipal effort to address urban law. The existing and fixed territorial data are the result of geography and territory, so it will be observed that there will be differences in the forecast. The cultural data are pre-existing data, meaning they evaluate the settlement system built over time and across generations, and are therefore highly stable. The Culture_Politics data also correspond to characteristics of the urban system whose value is flexible depending on the policies and investments adopted for their correction.

| MATRIZ URBANA | OUTPUT | | PREDICCION | REAL |
|---|---|---|---|---|
| 1,1 | n0/n1 | | 18,38 | 24 |
| 1,2 | INVMUN | M€ | 7,48 | 2,2 |
| 1,3 | AGR | % | 13,79 | 48,41 |
| 1,4 | IND_COM_PUB_MIL_PRIV | % | 6,4 | 2,8 |
| º2,1 | SALARIO | m€ | 9,26 | 10,8 |
| 2,2 | VIV/HOG | | 1,5 | 1,5 |
| 2,3 | DPOB | Mhb/km2 | 2,02 | 0,55 |
| 2,4 | RESD | % | 6,76 | 3,6 |
| 3,1 | INVMA | M€ | 1,22 | 0,26 |
| 3,2 | V_D_O | % | 3,72 | 0,6 |
| 3,3 | ∆ POB14/21 | % | 12,62 | 1,2 |
| 3,4 | NA | % | 55,51 | 1 |
| 4,1 | INVVU | M€ | 3,07 | 0,28 |
| 4,2 | DPOBTRES | mhb/km2 | 14,16 | 0,56 |
| 4,3 | COMPRES | mhb/km2 | 8,36 | 8,4 |
| 4,4 | NFT | % | 4,27 | |

It was noted that the results of the predictions allow for a comparison of urban or territorial realities with the same data, in order to identify discrepancies or imbalances that contribute to correcting the variable in question and/or directly or indirectly addressing the affected urban planning regulations. Thus, the difference between the obtained and predicted values reflects that policy decisions regarding investments (orange) for this scenario of moderate housing vulnerability are inadequate, making it likely that this vulnerability will persist over time unless this decision is corrected. It is considered relevant that the predicted population increase in a city like El Puerto de Santa María has practically stabilized. During the drafting of the Preliminary Report, it was also observed that the municipalities in the Bay and along the nearby coast had experienced population growth of around 8% since 2015, closer to the prediction, and this unique characteristic is also reflected in the present analysis.

On the other hand, it is observed that the values corresponding to territorial characteristics (green) continue to show deficiencies compared to the forecast. The reason for this difference may reveal some general deficit in the management of resources of the urban system of El Puerto de Santa María, which would need to be developed, including the excessive unemployment rate that is derived from the value 24 corresponding to n0/n1.

## 2. Bibliographies.

VISEDO MANZANARES, F. (21-24 de abril, 2024): “*Right to the City through urban entropy and enthalpy”* [Comunicación en congreso]. XIV Biennale of european towns and town planners. Napoli. (pp 110-111)
https://www.ectpceu-inubiennalenaples.com/_files/ugd/f7633c_77158ad707664f0d8087859ba0dadd8a.pdf

## 3. List of Acronyms/Abbreviations

| | |
|---|---|
| INE | Instituto Nacional de Estadística |