DERECHO A LA CIUDAD Y REDES NEURONALES URBANAS

Fernando **Visedo Manzanares**
Arquitecto

Derecho a la Ciudad y Redes Neuronales Urbanas

Resumen:
Diferentes disciplinas formulan funciones que relacionan variables económicas y sociales. Sectorialmente, la Nueva Ciencia de las Ciudades profundiza en el desarrollo de la sintaxis espacial urbana.

En reciente trabajo sobre Mecánica y Termodinámica Urbana se analiza la relación entre el Derecho a la Ciudad y el Sistema Urbano, explorando la relación integral entre todas estas variables urbanas formulando la matriz urbana que modela la ciudad identificando las funciones que explican el funcionamiento del Sistema Urbano.

Complementariamente, el presente trabajo desarrolla en el marco de la IA (Inteligencia Artificial) y big data urbanos analizados por RNU (Redes Neuronales Urbanas) la metodología para identificar una función oculta entre las distintas variables del Sistema Urbano Complejo para predecir el resultado que contribuya a subsanar la vulnerabilidad urbana del Derecho a la Ciudad.

Palabras clave: Derecho a la Ciudad; Red Neuronal; Ciencia Urbana; Inteligencia Artificial

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. Redes Neuronales Urbanas

La Mecánica Urbana[1] permite deducir resultados a partir de las ecuaciones que relacionan las diferentes variables sociales, económicas, sociales y espaciales. Las funciones que relacionan las diferentes variables urbanas se deducen a partir de diferentes estudios teóricos y prácticos de las diferentes disciplinas que analizan los vectores económicos, ambientales y espaciales orientados a la subsanación de la vulnerabilidad del conjunto de derechos que construyen el Derecho a la Ciudad. Las funciones que relacionan las variables urbanas se deducen tras el análisis pormenorizado de los sistemas. A medida que el sistema es más complejo, la dificultad para encontrar las funciones de relación también resulta más compleja.

El momento actual está caracterizado por la disposición de datos, obtenidos directamente de cualquier campo de actividad, de cualquier disciplina de investigación, de la vida cotidiana, en cualquier momento y en cualquier lugar. Innumerables dispositivos fijos y móviles, repartidos de forma imprevisible y no controlada, informan de los hábitos y de las conductas humanas y animales, del comportamiento de los grupos sociales, del estado del medioambiente o de las carreteras, y así se disponen de datos que penetran hasta el fondo impensable más íntimo de cualquier naturaleza. Este conjunto de datos o macrodatos (bigdata) que requieren el procesamiento de aplicaciones informáticas para su tratamiento y aplicación, abren un nuevo campo de investigación y análisis que permite estudiar las relaciones entre las causas y efectos de un fenómeno, descrito a través de sus variables, sin necesidad de conocer la relación profunda que existen entre ellas, a través de la comprobación del resultado que genera un fenómeno que forma parte de un sistema complejo.

Alternativamente a los métodos descritos en la mecánica en el que las variables se relacionan mediante funciones contrastadas, se pueden obtener predicciones de las variables objetivo a partir de las variables de entrada a través del análisis de las Redes Neuronales Urbanas en el marco del Deep Learning y la Inteligencia Artificial. La capacidad de cálculo de los procesadores unido a la capacidad metodológica de las redes neuronales permite incorporar un número de variables de entrada y de salida muy superiores al método estructurado de la Mecánica Urbana.

En efecto, la experiencia de la Mecánica Urbana permite identificar aquellas variables que configuran las componentes principales de cada uno de los Campos Urbanos. El recorrido realizado por el conjunto de submatrices de la Matriz Urbana descrito anteriormente relaciona el conjunto de las inversiones, públicas y privadas, con la subsanación de la vulnerabilidad del Derecho a la Ciudad.

Este recorrido aporta la experiencia suficiente para elaborar las RNU (Redes Neuronales Urbanas), analógicas o booleanas, que permiten adoptar decisiones a partir de las predicciones obtenidas, a partir de un número ilimitado de las variables urbanas que definen un sistema urbano complejo.

Se han construido 2 RNU en Visual Studio Code y en Colab. La primera, denominada RNU8_2, se construye con 8 neuronas de entrada y dos neuronas de salida para predecir el valor de variación poblacional y de (n0/n1) –(1/e) que muestra la tendencia a la vida o a la muerte de la ciudad. La segunda, denominada RNU1_15, se ha construido con 1 neurona de entrada y 15 neuronas de salida para predecir el impacto del aumento de inversión pública en la subsanación de vulnerabilidad de derechos.

Las RNU calculada se han probado con 1, 2 y 3 capas de neuronas ocultas para comprobar el error de la predicción. Dicha predicción se ha comprobado introduciendo los valores urbanos disponibles, comparándolos con los valores de las predicciones correspondientes outputs con los valores reales inputs. Se ha considerado aceptable un error del 5% por lo que la red con 3 neuronas ocultas ofrece resultados más cercanos a la realidad.

En la primera RNU se alcanza un aprendizaje elevado a partir del ciclo 200. Se ha realizado la normalización y la optimización del entrenamiento con la función de activación Adam con aproximación 0,001 y pérdida a través del error cuadrático.

En la segunda RNU el aprendizaje es más complejo y las predicciones outputs tienen una mayor dispersión respecto al valor real del input correspondiente. No obstante, es necesario aclarar que en la

[1] La Mecánica Urbana es desarrollada en el trabajo bajo el título “Vida y Muerte de la Ciudad. Derecho a la Ciudad y Ciencia Urbana”, formulado por el autor, que también se resume en el artículo del mismo título.

realidad un mismo input presenta diferentes outputs, ya que depende de cada ciudad o sistema urbano, en el que existen valores coincidentes de algunas variables, pero de valores dispares en otras variables. Esta circunstancia provoca una incertidumbre en el pronóstico ya que de la perturbación de una sola variable se pretende pronosticar el impacto en otras 15 variables por lo que es necesario adoptar las cautelas necesarias así como las comprobaciones complementarias con diferentes modelos que se construyan con más variables de entrada inputs.

El resultado de las predicciones permite enfrentar a la realidad urbana o territorial con las mismas, con el objeto de identificar dispersiones o desajustes que colaboren en la corrección de dicha variable y/o directa o indirectamente la subsanación del Derecho urbano afectado.

2. Metodología de elaboración de las Redes neuronales Urbanas

Previamente a la elaboración de la red neuronal se prepara una tabla con un conjunto de 90 variables urbanas de 36 capitales de provincia de España que permite en este momento inicial la construcción del bigdata urbano.

La elaboración de las diferentes redes neuronales se realiza siguiendo los scripts propios de esta técnica. En este sentido, es necesario aclarar que los modelos propuestos de RNU8_2 y RNU1_15 las neuronas se agrupan en capas. Las diferentes capas pueden realizar diferentes transformaciones en sus entradas desde la primera capa o capa de entrada, hasta la última capa o la capa de salida, a través de las diferentes capas ocultas.

El Algoritmo de la RNU sigue la siguiente estructura

2.1. Importar las librerías necesarias

Se inicia el proceso incorporando las librerías necesarias para ejecutar el programa en Colab o en Visual Studio Code

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

2.2. Definir las características (features) y los objetivos (targets)

Los datos se han obtenido de las plataformas del Instituto Nacional de Estadística y de Ministerio de Hacienda. De la primera se han obtenido la mayor parte d los datos urbanos correspondientes a Economía, Sociedad y Población, Ambientales y Espaciales. Los datos del Ministerio de Hacienda están relacionados con los presupuestos municipales. Se ha completado con datos de confección propia relacionados con población activa y Población no activa para calcular n0/n1.

2.2.1. Características

Se elaboran las características inputs o features con los datos de la tabla preparada. Los datos de entrada o neuronas de entrada inputs son las siguientes.
Es posible elaborar la tabla de características fuera del entorno del algoritmo Colab o Visual Studio Code y enlazar la ejecución con dicha tabla. Alternativamente se ha elegido introducir la matriz de datos en el modelo. Para ello se han exportado los datos de xls a cvc y txt con la cautela de separar los decimales con puntos y cada dato de la fila con comas.

2.2.2. Targets

Se han introducido los targets o datos de salida de cada una de estas ciudades. Los targets son los resultados que se pretenden pronosticar.

2.3. Normalización de los datos de entrada

Se normalizan los datos para reducir la posibilidad de error al interpretar las cifras, en el caso de tener gran disparidad entre ellas.

2.4. Definir el modelo de la red neuronal
Se define el número de neuronas de entrada y el número de neuronas de salida, así como el número de neuronas ocultas. Por otra parte, se define la función de activación del rectificador ReLU.

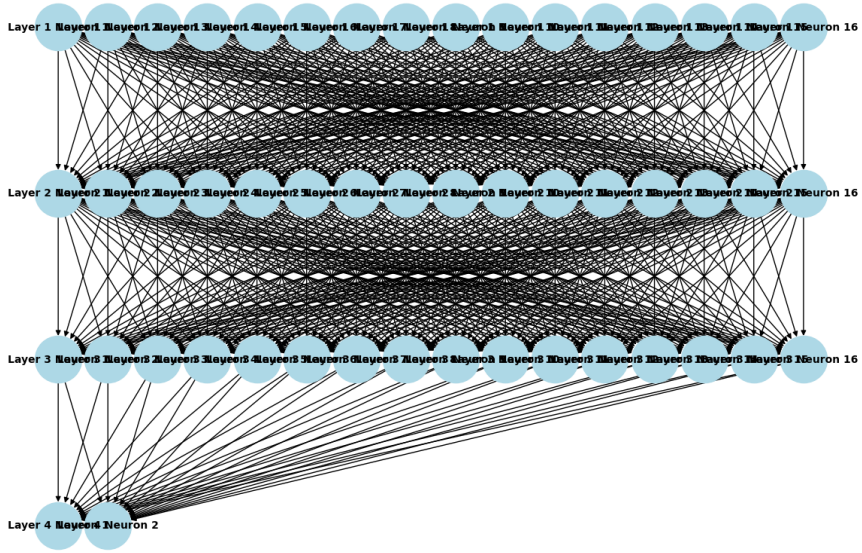

FIG. 1. Red Neuronal Urbana ensayada
Fuente: Elaboración propia obtenida de Algoritmo RNU8_2 con 16 neuronas en 3 capas ocultas en Colab

2.5. Compilar el modelo
Se ha usado el Algoritmo de Optimización del Gradiente ADAM con un grado de precisión adecuado con el objeto de reducir la probabilidad de error, sin alcanzar el sobreaprendizaje. Alternativamente se ha probado RMSprop pero no ha aportado mayor precisión en las predicciones.

2.6. Entrenar el modelo
El entrenamiento del modelo se ha ajustado a 1000 epochs o ciclos de aprendizaje de la red neuronal. En ambos casos, el aumento de los ciclos de aprendizaje no ha aportado mayor precisión en la predicción y tienen como efecto negativo el consumo de recursos del equipo además de aumentar el tiempo de cálculo.

2.7. Graficar la magnitud de la pérdida durante el entrenamiento
Se dibuja la gráfica del entrenamiento y de validación para percibir el proceso de acercamiento a los valores de salida que realiza la red neuronal en su aprendizaje.

2.8. Mostrar los pesos y sesgos de cada capa
Los pesos representan la contribución a la predicción final de cada una de las relaciones entre las características o neuronas de cada capa de la red neuronal desde la entrada input y las predicciones de salida output. Los sesgos son neuronas de polarización que aportan compensaciones colaboradoras en la predicción con un valor de entrada fijo de 1 y un peso asociado.
Se ha solicitado que muestre la secuencia de sesgos y pesos de cada ciclo de aprendizaje epoch.

2.9. Guardar el modelo completo en un archivo .h5
El archivo .h5 es responsable de almacenar los datos numéricos, gráficos y de texto de forma jerárquica. El entrenamiento se conserva en un archivo .h5 con objeto de utilizarlo en un modelo script de predicción .html con tensorflow para uso interno o para su disposición en el navegador previa conversión del archivo .h5 en .json y .bin mediante tensorflowjs.

2.10. Convertir y guardar el modelo en formato TensorFlow.js
Se realiza el guardado y serialización para modelos secuenciales con TensorFlow Keras en archivos .json

2.11. Función para hacer predicciones con el modelo entrenado
Se solicita realizar la predicción de los outputs a partir de los inputs.

La función que relaciona las entradas con la salida siendo X el vector de características de entrada, $W^{(i)}$ los pesos de las sucesivas capas, b(i) los sesgos de las sucesivas capas

$$y=W^{(3)}\cdot\text{ReLU}\left(W^{(2)}\right)\cdot\text{ReLU}\left(W^{(1)}X+b^{(1)}\right)+b^{(2)}\ )+b^{(3)}$$

Cada elemento de la función representa lo siguiente

1. Primera capa oculta:
   $\left(W^{(1)}X+b^{(1)}\right)$ Calcula la combinación lineal de las características de entrada con los pesos $W^{(1)}$ y sesgos $b^{(1)}$
   $\text{ReLU}\left(W^{(1)}X+b^{(1)}\right)$ Aplica la función de activación ReLU a la salida de la combinación lineal, lo que introduce no linealidad en el modelo. ReLU transforma los valores negativos en cero y deja los valores positivos sin cambios.

2. Segunda capa oculta:
   $\left(W^{(2)}\right)\cdot\text{ReLU}\left(W^{(1)}X+b^{(1)}\right)+b^{(2)}\ )$ Toma la salida de la primera capa (ya transformada por ReLU) y aplica otra combinación lineal con los con los pesos $W^{(2)}$ y sesgos $b^{(2)}$
   $\left(W^{(2)}\right)\cdot\text{ReLU}\left(W^{(1)}X+b^{(1)}\right)+b^{(2)}\ )$ Aplica nuevamente ReLU para introducir más no linealidad.
3. Tercera capa y capa de salida:
   $W^{(3)}\cdot\text{ReLU}\left(W^{(2)}\right)\cdot\text{ReLU}\left(W^{(1)}X+b^{(1)}\right)+b^{(2)}\ )+b^{(3)}$ La tercera capa toma la salida de la segunda capa transformada y aplica una última combinación lineal con los pesos pesos $W^{(3)}$ y sesgos $b^{(3)}$ para obtener la salida final y.

Se observa que las características de entrada X se transforman sucesivamente a través de tres capas ocultas utilizando combinaciones lineales, sesgos y la función de activación ReLU, para finalmente producir una salida y.

Durante el entrenamiento, los pesos $W^{(1)}$de primera capa oculta , $W^{(2)}$ de segunda capa oculta , $W^{(3)}$de la capa de salida y los sesgos $b^{(1)}$, $b^{(2)}$, $b^{(3)}$ de las capas respectivas, se ajustan utilizando un algoritmo de optimización (descenso de gradiente) para minimizar la diferencia entre los valores reales y los valores de predicción.

2.12. Normalizar los datos de entrada antes de predecir
Con objeto de interpretar los datos correctamente es necesario “desnormalizar” para recuperar las unidades iniciales.

2.13. Mostrar los cálculos internos de la red
Se solicita mostrar los cálculos para la comprobación, aunque esta operación no es necesaria para la predicción.

2.14. Calcular el error de predicción (MSE)
Se solicita mostrar el error de la predicción calculado mediante el Error cuadrático medio MSE. Esta comprobación es imprescindible para conocer la probabilidad de certeza y de verificación de la predicción. El MSE debe encontrarse por debe ser mayor que la cifra que se considere adecuada al cálculo.

2.15. Crear los widgets de entrada para los datos_botón de predicción_salida de datos
Con carácter operativo se elabora una tabla para introducir los inputs urbanos de nuevos municipios con objeto de realizar las predicciones. En este caso todos los valores son numéricos. Activando el botón se reiniciará para realizar nuevas predicciones al incluir los datos urbanos de nuevas ciudades.

3. Redes Neuronales Urbanas RNU8_2

Para desarrollar el presente trabajo se han seleccionado los siguientes inputs que describen el sistema urbano municipal desde el campo económico, social y espacial seleccionados entre las 90 variables.

Renta neta/Persona
Renta/Hogar
% población activa entre 15-64 años

% población 25-64 años máximo nivel educación
Precio medio vivienda 2023
Número de Viviendas
Compacidad
Número de viviendas/Número de hogares
La selección de indicadores se ha realizado de forma estimada para la muestra de resultados del presente ejercicio. En el trabajo de Mecánica Urbana ("Vida y Muerte de la Ciudad. Derecho a la Ciudad y Ciencia Urbana") se formula la propuesta de Análisis de Componentes Principales con objeto de identificar el menor número de variables urbanas que mejor representan al sistema urbano perdiendo la menor cantidad de información posible. Se escapa de este ejercicio dicho análisis y se han seleccionado variables de referencia utilizadas frecuentemente en los análisis urbanos.

Por otra parte, en el presente estudio interesa conectar con el trabajo citado para conocer el grado de vitalidad de la ciudad, es decir, el valor que refleja la distancia a la que se encuentra la ciudad del escenario de muerte en el que existe gran probabilidad de abandono de la ciudad por la población debido a la ausencia de estímulos para el desarrollo ciudadano con calidad de vida. Se demostró que dicho escenario está relacionado con la productividad de la población activa n1 o tasa de ocupados entre 20 y 64 años y de su relación con la población no activa n0 o tasa de desempleo, en términos de entropía urbana, cifras obtenidas de INE cuya suma es 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	1386	85.838	4,67	1,31	-2,73	0,16
Alicante	11.676	29.745	66,4	23,45	1826	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. Base de datos RNU 8_2 36 ciudades España
Fuente: Elaboración propia consultando INE

El código es una implementación completa de una red neuronal en Python utilizando TensorFlow, para analizar un conjunto de datos socioeconómicos. A continuación, se explica detalladamente cada parte del código, que incluye el proceso descrito en el apartado anterior de configuración, normalización de los datos, definición del modelo de red neuronal, entrenamiento, visualización de la estructura de la red y del rendimiento, y la predicción de nuevos datos ingresados manualmente. Aquí está la interpretación con sus partes fundamentales y explicaciones detalladas.

1. Importación de Librerías y Configuración del Entorno
El código se inicia con la importación de las librerías necesarias

```
!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
```

Las primeras líneas instalan TensorFlow y TensorFlow.js para manejar redes neuronales y realizar predicciones en aplicaciones web. Las otras librerías son usadas para manipulación de datos, visualización, escalado de características y visualización gráfica de la estructura de la red neuronal.

2. Definición de Datos de Entrada y Salida (Features y Targets)
El bloque siguiente contiene dos matrices features y targets, que representan las variables independientes (datos de entrada) y las variables objetivo (datos de salida) respectivamente

```
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],
    ...])
```

Estas matrices contienen datos socioeconómicos y demográficos, en donde cada fila de features representa un conjunto de valores independientes, y cada fila de targets representa dos valores de salida objetivo para esa fila de entrada en particular.

3. Normalización de los Datos de Entrada
Para mejorar el rendimiento del modelo, los datos de entrada (features) se normalizan usando MinMaxScaler, lo que lleva todos los valores a un rango que oscila entre 0 y 1

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

La normalización ayuda a que el proceso de aprendizaje sea más eficiente para evitar que la RNU tenga problemas al tener datos con escalas muy amplias.
En el caso que se analiza, la propia introducción de valores se ha realizado con una aproximación de valores, una especie de prenormalización, para facilitar el trabajo de normalización.

4. Definición del Modelo de RNU (Red Neuronal Urbana)

La red neuronal diseñada contiene tres capas ocultas, cada una con 64 neuronas y función de activación ReLU, y una capa de salida con dos neuronas para producir las dos salidas esperadas en targets. Se han probado otras opciones con 1 y con 2 capas ocultas que han ofrecido resultados con peores predicciones. La comprobación de la predicción se realiza calculando la diferencia entre el valor de salida output y el valor real que está relacionado con los 8 inputs de una ciudad disponible. Es necesario aclarar que la coincidencia exacta tiene una probabilidad prácticamente nula ya que dos ciudades con inputs iguales pueden presentar valores reales de referencia para outputs desiguales.

```
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. Visualización de la Estructura de la Red Neuronal

Una función l dibujar_red_neuronal permite visualizar gráficamente la arquitectura de la red neuronal usando la librería de gráficos networkx. La función crea un gráfico de la red intuitiva, con capas y conexiones entre las neuronas de diferentes capas.

```
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. Compilación y Entrenamiento del Modelo

El modelo es compilado con el optimizador Adam y la función de pérdida adecuada para problemas de regresión mean_squared_error,

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

El modelo se entrena con 1400 ciclos de aprendizaje epochs. Se ha probado con 500 ciclos y con 1500 ciclos, con peores resultados de predicción por infraaprendizaje y por sobreaprendizaje.

Los datos de entrenamiento se normalizaron previamente y, al finalizar, se grafica la pérdida para visualizar el progreso del modelo.

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

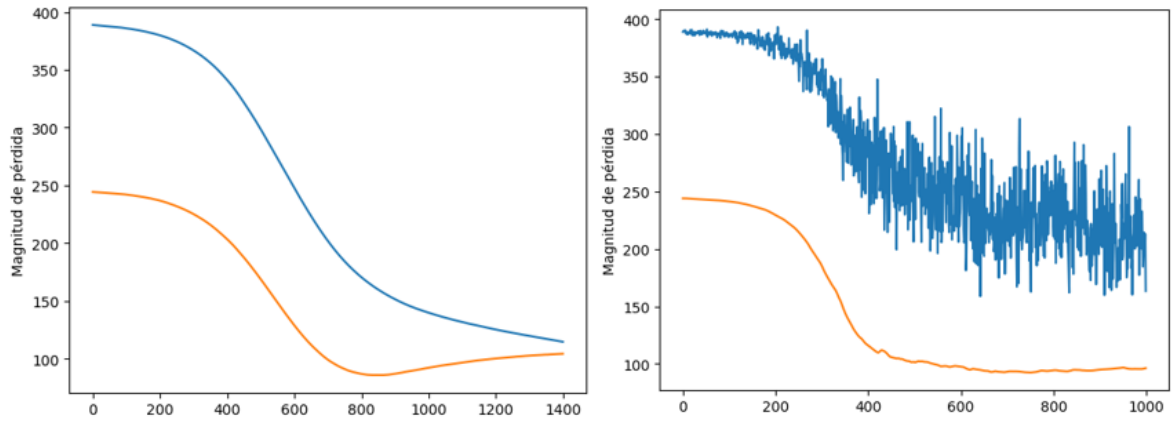

FIG. 3. Curva de aprendizaje (azul) y validación (naranja) de RNU8_2 sin dropout y con dropout 0.3
Fuente: Elaboración propia obtenida de Algoritmo RNU8_2 con 16 neuronas en 3 capas ocultas en Colab

7. Extracción de Pesos y Sesgos de la Red Neuronal
Se solicita la muestra de pesos y sesgos. El código imprime los pesos y sesgos que el modelo aprendió durante el entrenamiento para cada capa de la red neuronal.

```
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}")
```

Permite analizar el ajuste de los parámetros internos de la RNU para realizar el diagnóstico de las predicciones.

8. Guardar el Modelo en Formato Keras y TensorFlow.js
El modelo entrenado se guarda en formato .h5 para uso local y en formato TensorFlow.js para poder ser usado en la aplicación web. Su obtención es necesaria en las pruebas realizadas con Visual Studio Code.

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

9. Predicción en Nuevos Datos Ingresados Manualmente
La función make_prediction permite realizar predicciones en base a nuevos datos de actualización para predicción previa normalización de los mismos.

```
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. Cálculo del Error de Predicción (MSE)
El código incluye la función calcular_error que calcula el Error Cuadrático Medio (MSE) entre las predicciones del modelo y los valores reales incluidos en los targets

```
def calcular_error():
```

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

11. Gráfica de las Predicciones y Valores Reales

La función graficar_predicciones permite comparar visualmente las predicciones del modelo en relación a los valores reales, creando un gráfico de dispersión para cada variable objetivo

```
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. Interfaz_Widgets para Entrada de Datos del Usuario

Finalmente, el código crea widgets interactivos que permiten ingresar valores para los diferentes parámetros con objeto de predecir el resultado, que en este caso es n0/n1. Este conjunto de widgets es útil para implementar el modelo en el entorno como Visual Studio Code o Google Colab con un interfaz comprensivo, como se muestra en la 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)
```

Incluye la predicción del objetivo buscado en la RNU, pudiendo introducir nuevos valores para realizar nuevas predicciones.

El código descrito genera un flujo completo del desarrollo de una red neuronal para predicción en datos socioeconómicos, ambientales y espaciales incluyendo la base de datos, la generación y el entrenamiento del modelo hasta la integración de la interfaz gráfica interactiva para realizar predicciones de outputs previstos, en este caso (n0/n1) – (1/e), en tiempo real. Este valor refleja la aproximación que el sistema urbano presenta ante el riesgo de su “muerte” o abandono debido a la ausencia de oportunidades para la población. En este caso se han identificado 8 valores que caracterizan la “vitalidad de la ciudad”. A medida que el valor aumenta entre 0 y 1 aumenta también la

vitalidad de la ciudad. Un valor negativo representa que el sistema urbano ha superado el momento de recuperación y requiere una inyección de recursos externos extraordinarios para su supervivencia. Las variables urbanas que se relacionan con este indicador de vitalidad urbana pueden ser sustituidas por otras que se consideren más adecuadas, como resultado de los estudios de Mecánica Urbana desarrollados o por disponer de valores de otros indicadores urbanos.

Interpretación de los parámetros de las 3 capas ocultas 1,2,3
Cada una de las tres capas ocultas tiene 16 neuronas y cada una de ellas tiene un conjunto de pesos y un sesgo asociado. Los Pesos son los valores que determinan la influencia de cada entrada en cada neurona de la capa que se reajustan durante el entrenamiento del modelo para minimizar el error entre la predicción y los valores reales. Los Sesgos colaboran en ajustar la función de activación, desplazándola para activar o no activar una neurona dada.

Interpretación de los parámetros de la capa 4 (Capa de Salida)
En la capa de salida, cada una de las neuronas representa una salida esperada del modelo. En este caso se proyectan 15 neuronas en la salida que coinciden con 15 outputs correspondientes.

Análisis general del modelo
A pesar de comprobar que el modelo alcanza un aprendizaje suficiente, los valores extremos de algunos pesos sugieren que ciertas neuronas pueden estar más influenciadas que otras, que valida las relaciones previstas entre datos de entrada y los outputs. No obstante, los valores de peso excesivamente altos indican que el modelo está sobreajustado a ciertos patrones en los datos de entrenamiento.

Se ha comprobado que la curva de validación se separa de la curva de aprendizaje a partir del ciclo 600 aproximadamente, momento en que aún el error en la predicción es elevado. Es posible regularizar el modelo aplicando técnicas de L2 o de dropout para reducir la dependencia de los pesos extremos y mejorar el comportamiento de la RNU. El resultado de regularización con L2 consiste en reducir el valor de los pesos para que sean pequeños, no ha mejorado la magnitud de pérdida de entrenamiento (línea azul) aunque ha mejorado la validación (línea naranja). La regularización con dropout que consiste en eliminar neuronas aleatoriamente ofrece un descenso inicial de ambas líneas hasta los 200 ciclos en donde la pérdida de entrenamiento y la de validación disminuyen, lo que indica que el modelo está aprendiendo patrones tanto en los datos de entrenamiento como en los datos de validación. Posteriormente, la pérdida de entrenamiento comienza a oscilar y muestra un aumento general en lugar de continuar disminuyendo debido a fluctuaciones significativas en el ajuste de los datos de entrenamiento. La pérdida de validación comienza a aumentar después de alcanzar un mínimo debido a sobreajuste (overfitting) a los datos de entrenamiento, lo que supone que el modelo no generaliza bien en datos nuevos y se está adaptando a los valores de los datos de entrenamiento. Por otro lado, las oscilaciones en la pérdida de entrenamiento señalan una dificultad del modelo para estabilizar el aprendizaje en los datos de entrenamiento.

Evaluación cuantitativa del error
El Error Cuadrático Medio (MSE) y R^2 cercano a 1 relativo a la escala de los datos de salida señala que el modelo está haciendo predicciones precisas, mientras que valores altos de MSE y un R^2 bajo señala problemas en la precisión. El modelo ha indicado un Error Cuadrático Medio (MSE) de 112.

Si los valores de salida son significativamente más altos que el valor del MSE el error es aceptable. Sin embargo, si los valores de salida están en un rango inferior, es posible que la desviación entre las predicciones y valores reales sea excesiva. Si se obtienen errores altos en outputs específicos, el modelo no captura correctamente las relaciones para algunos outputs en particular. En el análisis realizado se contrasta el MSE con la evaluación visual del error para adoptar la decisión sobre la validez del modelo.

Evaluación visual del error:
Al comparar los valores reales y los valores predichos en la gráfica de dispersión se observa que se alinean bien en la línea de identidad (y=x), por lo que se deduce que el modelo está prediciendo correctamente.

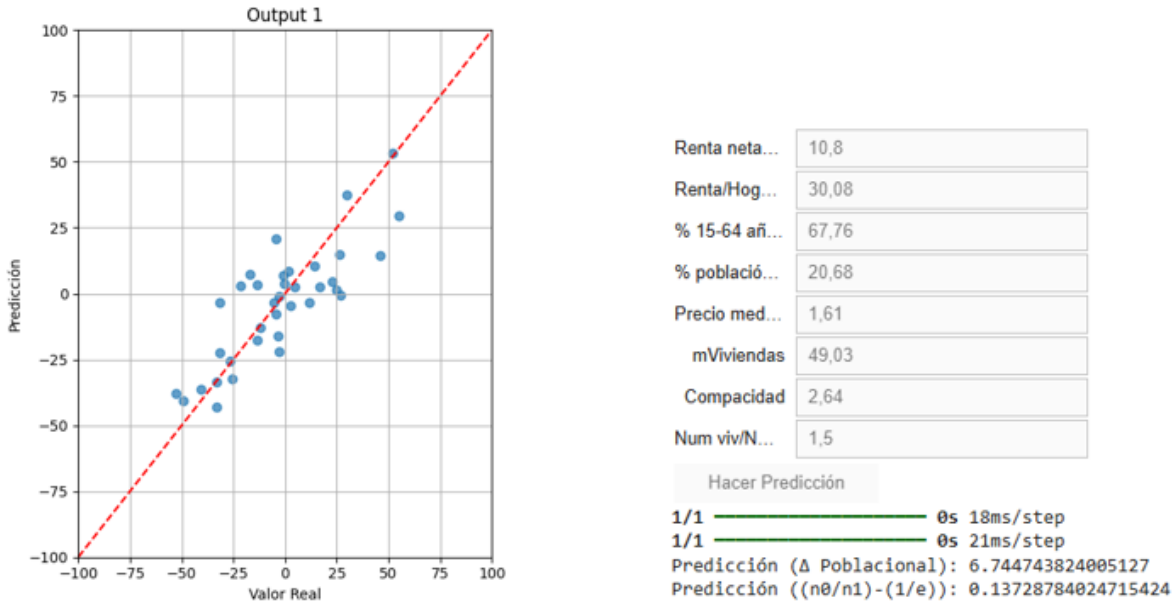

FIG. 4. Dispersión entre valor real input y predicción target a izquierda y widgets de interfaz de predicción a derecha
Fuente: Elaboración propia obtenida de Algoritmo RNU8_2 con 16 neuronas en 3 capas ocultas en Colab

La gráfica de dispersión muestra la relación entre los valores reales y las predicciones del modelo para el Output. Se observa una tendencia ascendente en los puntos dispersos a lo largo de la línea discontinua roja de referencia por lo que el modelo logra una correlación positiva entre ambas variables. Los puntos están más concentrados en torno a los valores cercanos a cero y dispersos en los valores más extremos debido a que la distribución de los datos reales es próxima a cero en su mayoría.
También se observa la dispersión en los valores más extremos respecto a la línea de referencia, aunque varios puntos están cerca de la misma, lo que quiere decir que el modelo presenta inexactitud en los valores reales y predichos más lejanos al origen cerca de ±100.

El modelo puede generar un error de predicción que afecta el MSE debido a que algunos puntos se alejan de la línea de referencia. El MSE de 112 es moderado por lo que se han realizado pruebas de ajuste con Regularización mediante L2 y dropout pero con resultados que provocan incertidumbre en el aprendizaje y la validación.

Por todo ello se puede concluir que el modelo ofrece resultados acertados, aunque es posible mejorar la predicción con ajustes adicionales, pero sobre todo con más inputs para el entrenamiento que permitan capturar mejor las variaciones. Lamentablemente la base de datos es limitada a 36 ciudades, lo que reduce considerablemente la capacidad de aprendizaje. El presente modelo ha establecido relaciones entre diferentes inputs y outputs. No cabe duda que es posible encontrar otras relaciones que puedan mejorar la predicción, lo que supone una vía de investigación diversa y pluridisciplinar con objeto de fortalecer la herramienta de predicción del comportamiento de una ciudad, de un sistema urbano o de un sistema territorial con objeto de evaluar diferentes escenarios debido a perturbaciones de cada uno de las diferentes variables urbanas. No obstante, el arranque del presente estudio permitirá sistematizar la obtención de datos urbanos agrupando por rango de ciudades y localización de las mismas.

4. Redes Neuronales Urbanas RNU1_15

Ahora se calculará un conjunto de características urbanas a partir del valor de una variable. Las matrices X e Y representan los datos de entrada y los valores objetivo (o de salida) del problema de aprendizaje automático.

X es la matriz unidimensional que contiene los valores de entrada del modelo sobre vulnerabilidad del derecho a la vivienda tomando como referencia el valor de número de hogares/número de viviendas.

Y es la matriz bidimensional donde cada fila representa los atributos de cada input urbano, que coinciden con los objetivos o targets urbanos que se pretenden predecir con el modelo que en este caso se trata de 15 variables que modelizan la ciudad, el sistema urbano o el sistema territorial. Se han seleccionado las siguientes variables obtenidas del INE

n0/n1	Vitalidad de la ciudad n0=población inactiva n1=población activa
INVMUN M	Inversión municipal en
% AGR	Porcentaje suelo agrario
% IND_COM_PUB_MIL_PRIV	Porcentaje suelo industrial_comercial_publico_militar_privado
SALARIO m€	Salario en miles €
HOG/VIV	Relación entre número de hogares y número de viviendas
DPOB	Densidad poblacional en miles de habitantes por Km2 mhab/km2
% RESD	Porcentaje de suelo residencial
INVMA	Inversión municipal en Medioambiente en Millones € M€
% V_D_O	Porcentaje de suelo
∆ POB14/21	Incremento de población entre 2014 y 2021
% NAT	Porcentaje de suelo natural
INVVU	Inversión municipal en zonas verdes urbanas en millones € M€
DPOBTRES	Densidad poblacional suelo residencial miles habitantes por Km2
COMPRES	Compacidad residencial en miles viviendas por Km2 m viv/Km2res
%INFT	Porcentaje suelo en infraestructuras de comunicación

Cada una de estas variables tiene una correspondencia en la Matriz Urbana formulada en *Right to the City through urban entropy and enthalpy*[2]. En dicha matriz el input sobre vulnerabilidad ocupa la posición (2,2). El resto d casillas están ocupadas por las variables anteriores siguiendo el orden por filas i

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

Es necesario aclarar que las variables elegidas para representar la magnitud de la matriz urbana están condicionadas por los datos municipales existentes en el INE, por lo que su representatividad será limitada en algunos casos como se analizará posteriormente.

1. Instalación e Importación de Librerías
Al igual que en el modelo anterior, se instalan e importan las bibliotecas necesarias para el análisis y visualización de datos, así como para la construcción de la red neuronal

```
# 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
```

Este conjunto de bibliotecas permite
Crear y entrenar modelos de redes neuronales (`tensorflow`, `keras`).
Trabajar con arreglos y realizar cálculos matemáticos (`numpy`).
Visualizar datos y gráficos (`matplotlib`, `seaborn`).
Construir una interfaz de usuario interactiva (`ipywidgets`, `IPython.display`).

[2] La Matriz Urbana es el resultado de modelizar el sistema urbano en una serie de valores sociales, económicos, ambientales y espaciales, que tiene por objeto evaluar la vulnerabilidad del Derecho a la Ciudad. La Mecánica Urbana es un trabajo de investigación desarrollado por el autor que explora la dependencia funcional de estos valores en el trabajo “Vida y Muerte de la Ciudad. El Derecho a la Ciudad y la Ciencia Urbana”.

2. Definición de los Datos de Entrada y Valores Objetivo

Se definen las matrices de valores reales X, Y que representan los datos de entrada y los valores que el modelo debe aprender a predecir

```
# 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)
])
```

Los valores de la matriz 1x34 X representan el número de viviendas entre el número de hogares del sistema urbano como coeficiente que representa la vulnerabilidad del derecho a la vivienda en este caso. Es necesario aclarar que se ha adoptado este coeficiente sencillo de obtener, pero que es posible tomar como referencia el número de demandantes de vivienda, que ofrecería una mejor medida de la vulnerabilidad de este derecho urbano. Este dato está publicado por algunas administraciones autonómicas con rango municipal pero no ha sido posible incorporarlo debido a que muchas de las ciudades de la base datos no disponen del dato en plataformas abiertas.

	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 (mhab/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

Los valores de la matriz 34x15 Y representan los 15 valores reales de las columnas que forman la matriz urbana de cada una de las 34 ciudades que ocupan las filas. Estos 15 valores son los targets u objetivos a predecir. Es necesario aclarar que, a diferencia del modelo anterior que incluye datos de 36 ciudades, aquí se han eliminado los datos de Madrid y Barcelona ya que provocan problemas de dispersión de las predicciones al disponer de datos muy diferentes al resto, aspecto que afecta a la prenormalización que se realiza para mejorar el aprendizaje del modelo.

3. Normalización de los Datos

La normalización transforma los datos a valores entre 0 y 1 para facilitar el entrenamiento del modelo, ya que los algoritmos de redes neuronales funcionan mejor con datos normalizados

```
# 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
```

La normalización consiste en calcular el valor máximo en X e Y y se divide cada valor por estos máximos, generando X_normalized e Y_normalized, que son versiones normalizadas de los datos de entrada y salida.

4. Arquitectura de la Red Neuronal Urbana

El modelo de red neuronal se define con la clase Sequential de Keras, con tres capas ocultas y una capa de salida con 15 neuronas, que se corresponde con los 15 valores en cada fila de 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')
])
```

La primera capa Input indica que el modelo espera una sola entrada de dimensión 1.
Se incluyen cinco capas Dense de 128 y 64 neuronas, cada una con activación lineal relu que permite generar cualquier valor real. Es posible aplicar la regularización L2 pero se ha comprobado que en este caso su aportación para reducir el sobreajuste penalizando los pesos de las neuronas es innecesario.
La capa de salida Dense15, activation=linear tiene 15 neuronas, cada una representando un valor de salida para los 15 elementos en la matriz Y.

5. Compilación y Entrenamiento del Modelo
El entrenamiento del modelo se ejecuta con el optimizador Adam, con una tasa de aprendizaje baja (0.00001) y la función de pérdida mean_squared_error

```
# 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)
```

Se entrena el modelo durante 1000 ciclos, reservando el 20% de los datos para la validación, mediante una retroalimentación en cada ciclo durante el entrenamiento para ajustar los pesos de la red y minimizar el error de predicción.

6. Visualización de la Pérdida de Entrenamiento

Se añade script para graficar la pérdida en el tiempo con objeto de evaluar la convergencia del modelo o los problemas de sobreajuste. Esta comprobación es fundamental para determinar la corrección del modelo como se ha observado en el modelo anterior al ofrecer problemas en la validación durante el entrenamiento, sin embargo, en este caso el modelo ha respondido con mejor comportamiento.

```
# 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()
```

Se grafica tanto la pérdida en el conjunto de entrenamiento como en el de validación, lo que permite observar si el modelo mejora consistentemente o si está sobreajustado.

7. Cálculo del error cuadrático medio MSE
Se calcula el error cuadrático medio como un valor de referencia que muestra la confianza en el modelo predictivo.

```
# 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. Funciones de Predicción y Cálculo del Error
Se crean las funciones para predecir un valor dado predecir y para calcular el error absoluto entre los valores reales y predichos calcular_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)
```

Predecir toma un valor de entrada, lo normaliza y obtiene la predicción al ejecutar el modelo, para posteriormente desnormalizarlo para obtener el valor de predicción en su escala original multiplicando por Y_max.
calcular_error ofrece el error absoluto, como diferencia en valor absoluto entre el valor real y el predicho.

9. Cálculo del Error Absoluto y Gráfica de Error
Se calculan los errores absolutos entre las predicciones y los valores reales para cada salida

```
# 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()
```

Cada subplot muestra el error absoluto de cada uno de los 15 outputs.

10. Creación de Tabla de Predicción 4x4
Se construye una función que crea una tabla de 4x4 con predicciones y la visualiza con seaborn. Es necesario recordar que la tabla 4x4 es la matriz urbana compuesta por los valores outputs o targets resultantes de la predicción.

```
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", "Δ 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()
```

En la figura 6 se observa que el input se sitúa en la casilla 2,2 de la Matriz Urbana correspondiente a vulnerabilidad social de derechos urbanos.

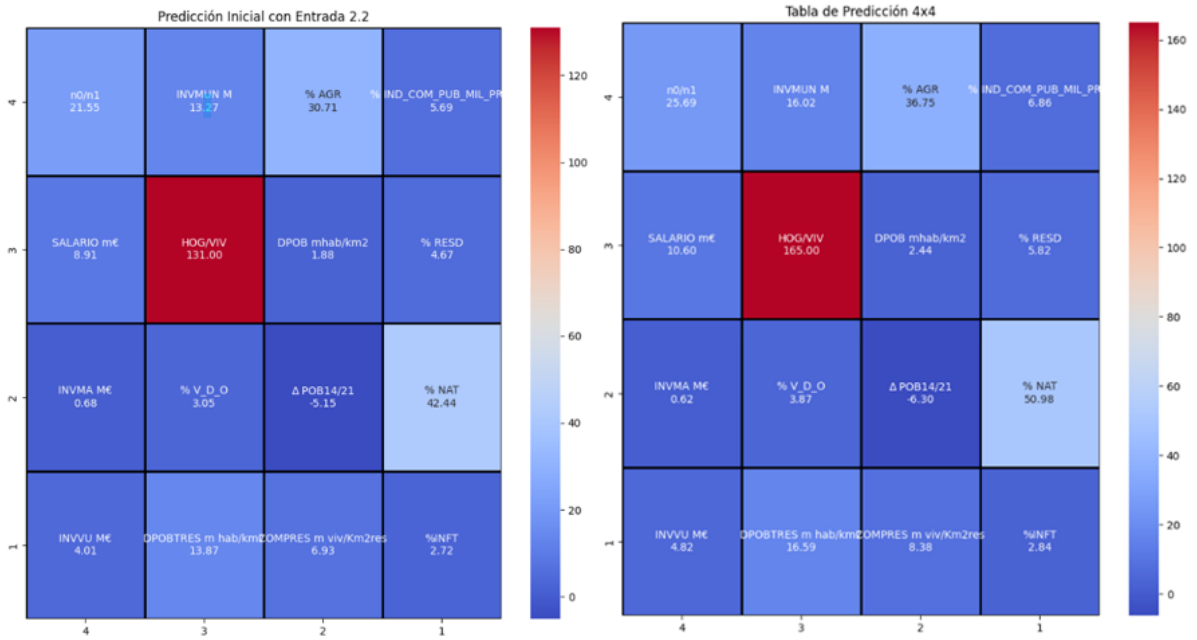

FIG. 6. Tabla 4x4 de predicción con valor inicial y con valor interactivo con widgets.
Fuente: Elaboración propia obtenida de Algoritmo RNU1_15 en Colab

11. Interfaz Interactiva con Widgets

Para realizar predicciones en tiempo real, se utilizan widgets interactivos de ipywidgets que permite ajustar los valores de entrada y actualizar la predicción y visualización de la tabla 4x4 en tiempo real

```
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)
```

El interfaz se presenta en este caso como una barra que permite introducir valores de input entre 0 y 200, correspondientes a valores reales entre 0 y 0,2 como se muestra en la Fig. 10

12. Gráfica de Dispersión Final

Por último, se crea una gráfica de dispersión de los valores reales frente a los valores predichos para los 15 outputs, que permite evaluar visualmente la precisión del modelo.

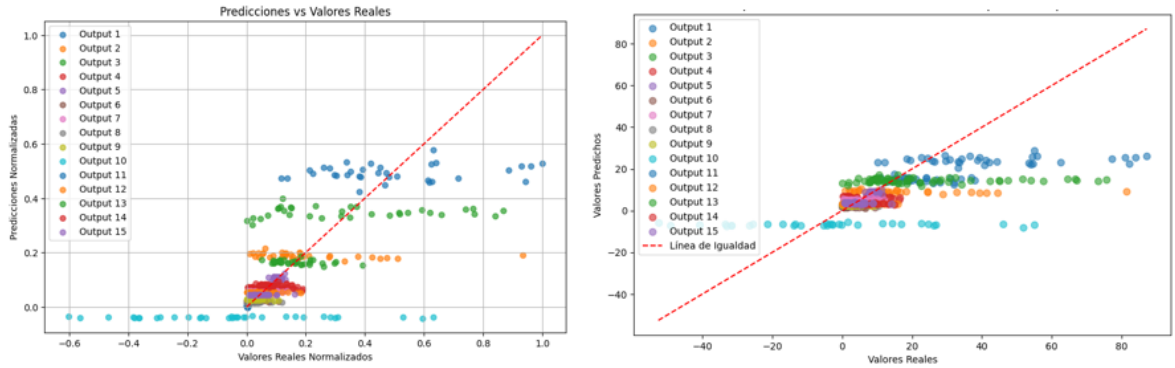

FIG. 7. Dispersión valores reales/valores predichos en gráfico izquierdo con valores normalizados y en gráfico derecho con valores reales.
Fuente: Elaboración propia obtenida de Algoritmo RNU1_15 con 3 capas ocultas en Colab

En la figura 6 se observa la distribución del error en las salidas. Cada uno de los outputs corresponde al conjunto de 15 valores reales Y con los valores predichos por la red neuronal. Se revisa el error para cada salida de forma independiente y se observa que algunos outputs presentan predicciones con mucha precisión, mientras que otros presentan algún error. La línea de igualdad diagonal con origen (0, 0) hasta el punto máximo del rango de los outputs representa la equivalencia entre valores reales y valores predichos. La mayor proximidad a la línea de igualdad supone que la Red Neuronal Urbana contiene errores no significativos en las predicciones. Por el contrario, el alejamiento de la línea de igualdad supone un error significativo de predicción de la Red Neuronal Urbana.

La gráfica muestra una dispersión entre los valores reales y los valores predichos de la red neuronal para algunos outputs, especialmente en rangos mayores o menores que presentan una dispersión alejada de la línea de igualdad, a pesar de que el modelo ha alcanzado una pérdida de entrenamiento final de 0.0592 y una pérdida de validación de 0.0616, aunque el modelo parece haber logrado una buena convergencia en términos de pérdida. No se considera necesario realizar la comprobación de relación de errores por overfitting o underfitting con L2, ni con dropout, ni con early stopping ya que se observa un correcto descenso de la pérdida en el entrenamiento y en la validación con el desarrollo del entrenamiento.

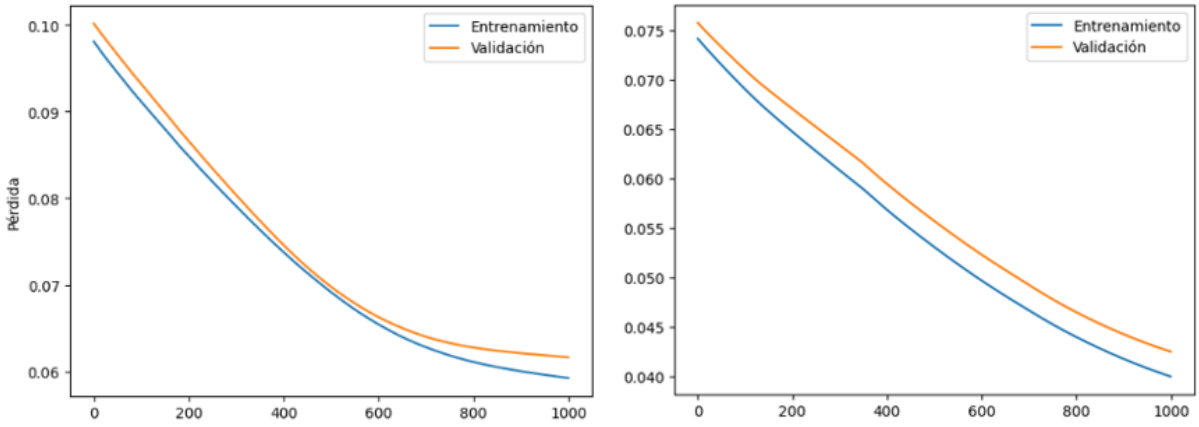

FIG. 8. Pérdida de entrenamiento y pérdida de validación
Fuente: Elaboración propia obtenida de Algoritmo RNU1_15 con 3 capas ocultas en Colab

El análisis de los errores para cada output permite a identificar áreas donde el modelo necesita mejorar. No obstante, se han realizado también pruebas relacionadas con el ajuste de de Hiperparámetros ajustando la Tasa de aprendizaje más baja o ajuste dinámico durante el entrenamiento (learning rate decay) para mejorar la convergencia. También se han añadido diferentes configuraciones de capas y neuronas. No es posible sin embargo aumentar el Tamaño del Conjunto de Datos, aspecto que puede mejorar sustancialmente el comportamiento del modelo. Por el contrario, se han eliminado los valores de Madrid y Barcelona por considerarlos Outliers.

5. Aplicación de redes neuronales para pronosticar la vitalidad urbana de el Puerto de Santa María.

La redacción del Avance de PGOM de El Puerto de Santa María analiza 4 alternativas cuya diferencia sustancial más significativa está relacionada con la ocupación de suelo.

En los análisis realizados para identificar la mejor opción urbanística mediante la Evaluación Ambiental Estratégica, la Alternativa 3 resulta la elegida desde el punto de vista ambiental siguiendo métodos de valoración cuantitativa, en los cuales ha obtenido claramente un mayor valor que las otras tres alternativas debido a su buen comportamiento ante la mayoría de los aspectos ambientales clave.

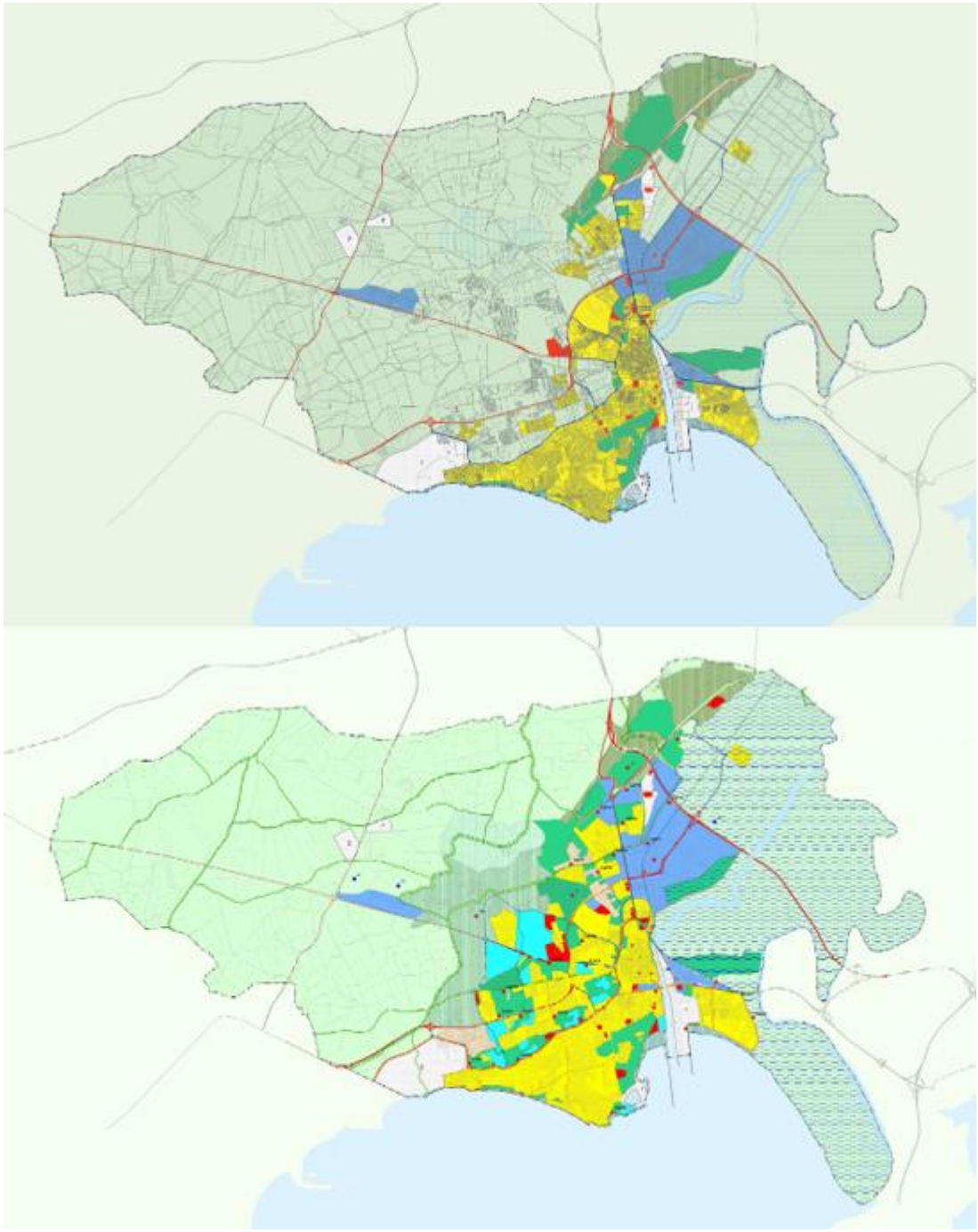

FIG. 9. Alternativa A0 y A3 del Avance PGOM El Puerto de Santa María (2023)
Fuente: Elaboración propia

Uno de los aspectos más destacables tenidos en cuenta en esta valoración es la recomposición de la ciudad existente, debido a que el principio que configura la Alternativa 3 es el desarrollo urbano a través del crecimiento casi nulo, consiguiendo de esta forma minimizar la ocupación de nuevos suelos. Para ello, resulta ambientalmente más favorable que la Alternativa 2, ya que plantea una menor cantidad de superficie de nuevo crecimiento urbano y no incluye infraestructuras que se han considerado con posterioridad como innecesarias, apostando por la ampliación y mejora de las carreteras existentes.
Algo similar ocurre con respecto a las Alternativas 0 y 1. La Alternativa 0 supone dejar la situación en la situación actual sin intervenir urbanísticamente, manteniendo el caserío diseminado irregular de más de 5.200 viviendas repartidas en la periferia occidental del área urbana (en color amarillo). La Alternativa 1, es equivalente al PGOU anulado de 2012 por no haber realizado la Evaluación Ambiental Estratégica. En dicha alternativa se opta por ampliar la superficie de suelo urbano hasta acoger todas las urbanizaciones irregulares, mallas sobre el color verde en la parte occidental, tratando todo el suelo de forma homogénea y en donde prevalece la expansión urbana en vez de la reutilización y compleción de la ciudad existente. Este hecho es clave si se tiene en cuenta la situación de algunos barrios del casco urbano de El Puerto de Santa María, que necesitan ser reacondicionados, sobre todo en cuanto a vivienda se refiere.

De forma complementaria se ha aplicado la metodología de redes neuronales urbanas para predecir la vitalidad de la propuesta urbana a las alternativas A0 y A3 del documento de Avance del PGOM de El Puerto de Santa María.

Se ha restringido el análisis de vitalidad urbana atendiendo exclusivamente al incremento de viviendas y a la compacidad de miles de viviendas /Km2. En el caso de la Alternativa 3 se incluyen las viviendas irregulares y en el caso de la Alternativa 0 las viviendas irregulares se mantienen como ilegales desplegadas en todo el suelo rústico. El resto de parámetros se mantienen constantes en este ejercicio, como se señala en el siguiente cuadro.

INPUT		A3	A0
Renta neta/Persona	m€	10,83	10,83
Renta/Hogar m€	m€	30,08	30,08
Población 15-64 años	%	67,76	67,76
población 25-64 años universitario	%	20,68	20,68
Precio medio vivienda 2023	m€/m2	1,614	1,614
Viviendas	mviv	49,03	43,83
Compacidad	mviv/km2 res	2,14	2,64
Num viv/Num hogares		1,5	1,3

Se ha considerado que los valores que se alteran en el presente análisis están relacionados con la vivienda. El número de 5200 viviendas aumenta al alcanzar la regularización, con las dotaciones urbanas o suburbanas correspondientes a cada clase de suelo, que en cualquier caso garantizan la seguridad, salubridad y habitabilidad de las mimas. Por otro lado, al establecer áreas urbanas y suburbanas delimitadas, la superficie de suelo residencial ordenado disminuye por lo que la compacidad aumenta. Por último, el aumento del número de viviendas legales y regulares que aumenta en 5200 unidades permite aumentar la disposición a los hogares. La proporción de viviendas disponibles por hogar debe ser analizado y contrastarlo con el número de 4409 viviendas vacías en El puerto de Santa María, que suponen un 9% del total de viviendas según INE.

Tomando esta red neuronal de 3 capas ocultas se ha calculado el valor (n1/n0) - (1/e) que informa sobre la vitalidad urbana. El resultado permite comprobar que la Alternativa A3 mantiene una distancia de 13,7 (0,23) puntos y en la Alternativa A0 mantiene una distancia de 12,9 (0,24) puntos. Estas cifras son equivalentes a una tasa de desempleo de 18% y de 19% respectivamente.

El análisis ofrece el valor de vitalidad urbana (n0/n1) - (1/e) = 0,137, por lo que n0/n1 asciende a 0,24. Esta proporción es equivalente a una tasa de desempleo del 18% que sitúa a El Puerto de Santa María en una situación precaria en cuanto a desarrollo en escenarios futuros salvo que existe un aporte

externo. Se comprueba que esta predicción es consistente al comparar con la realidad ya que la tasa de desempleo asciende al 18% en el momento de realizar el cálculo en 2022.

Por todo ello se considera que es recomendable realizar una planificación del Suelo urbano regularizando y legalizando 5200 viviendas irregulares en la actualidad

A continuación se evalúan los indicadores de la Matriz Urbana 4x4 introduciendo el valor de repercusión del número de 1,5 viviendas por hogar que asciende a 1,50. Esta situación representa una situación de escasa vulnerabilidad ya que existe una proporción elevada de viviendas por hogar respecto a otras ciudades, aspecto que es coincidente con los 578 demandantes de vivienda, que representa un 0,6 % de la población, que difiere sustancialmente de los 4752 de Cádiz, que representa u 4,2 % de la población.

La ejecución del modelo ofrece un escenario pronosticado de la Matriz Urbana resultante de aplicar el comportamiento del conjunto de ciudades medias españolas en el año 2022 a El Puerto de Santa María en un escenario de moderada vulnerabilidad de vivienda de 1,5 viviendas por hogar.

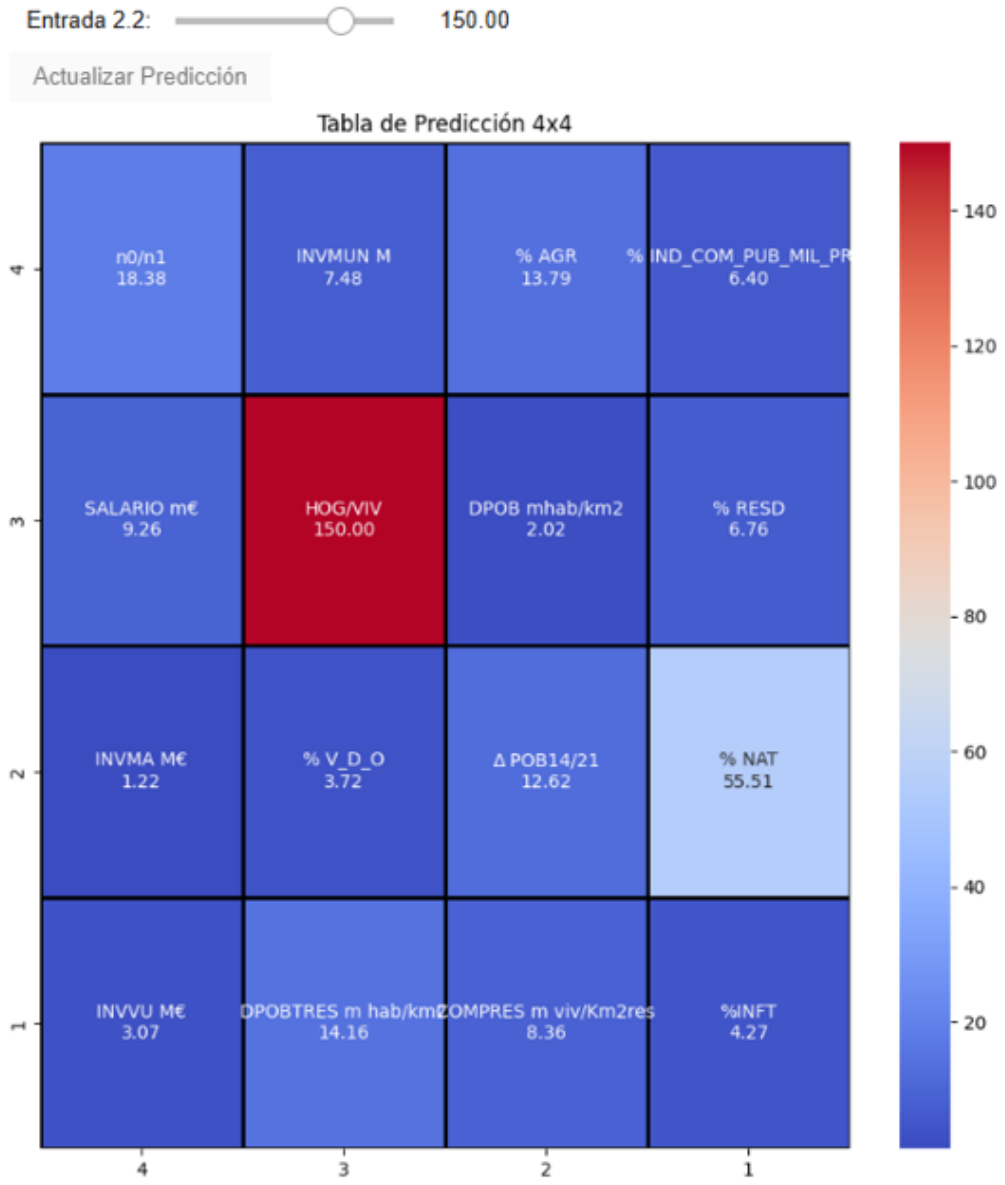

FIG.10. Predicción valores Matriz Urbana El Puerto de Santa María (2023)
Fuente: Elaboración propia obtenida de Algoritmo RNU1_15 con 3 capas ocultas en Colab

Estos datos se pueden clasificar en los siguientes apartados

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

El apartado de datos propios se considera representativo del elemento de la matriz urbana que modeliza el sistema urbano. El apartado de política representa el esfuerzo municipal en abordar el derecho urbano. Los datos territoriales existentes y fijos ya que son el resultado de la geografía y del territorio por lo que se comprobará que existirán diferencias en el pronóstico. Los datos culturales son datos preexistentes, es decir que evalúan el sistema de asentamiento construido a lo largo del tiempo y de generaciones por lo que resultan altamente estables. Los datos de Cultura_Política también corresponden a características del sistema urbano cuyo valor es flexible dependiendo de las políticas e inversiones adoptadas para su corrección.

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	

Se señaló que el resultado de las predicciones permite enfrentar a la realidad urbana o territorial con las mismas, con el objeto de identificar dispersiones o desajustes que colaboren en la corrección de dicha variable y/o directa o indirectamente la subsanación del Derecho urbano afectado. Así, la diferencia de los valores obtenidos y los pronosticados refleja que las decisiones políticas en materia de inversiones (naranja) para este escenario de vulnerabilidad de vivienda moderada son deficitarias por lo que es probable que dicha vulnerabilidad se prolongue a lo largo del tiempo salvo que se corrija dicha decisión. Se considera relevante que la predicción del incremento poblacional en una ciudad como El Puerto de Santa María se haya estabilizado prácticamente. Durante la redacción del Avance se observó también que los municipios de la Bahía y del litoral próximos habían aumentado la población en torno al 8% a partir del 2015, más cercana a la predicción, por lo que esta singularidad también se detecta en el presente análisis.

Por otro lado se observa que los valores correspondientes a características territoriales (verde) mantienen diferencias deficitarias respecto al pronóstico. El motivo de esta diferencia puede revelar algún déficit general sobre gestión de recursos del sistema urbano de El Puerto de Santa María, que sería necesario desarrollar, incluyendo la excesiva tasa de desempleo que se deriva del valor 24 correspondiente a n0/n1.

2. Bibliografía.

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. Listado de Acrónimos/Siglas

INE Instituto Nacional de Estadística