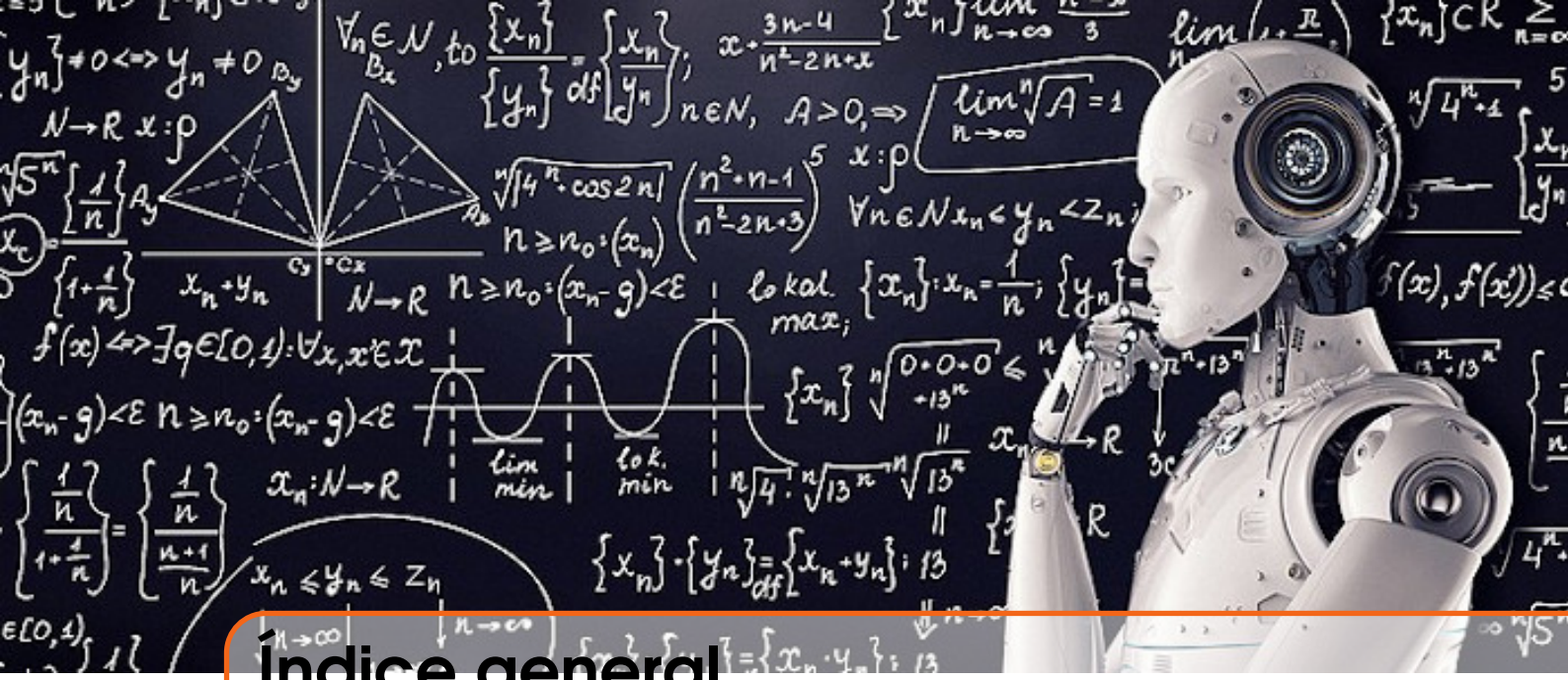
Ejercicios Aprendizaje Automático I

Grado en Ciencia e Ingeniería de Datos

Escuela de Ingeniería en Informática

Universidad de Las Palmas de Gran Canaria

Javier Lorenzo Navarro y José Carlos Rodríguez Rodríguez



Índice general

Introducción

Este documento recoge ejercicios propuestos correspondientes a las prácticas de la asignatura Aprendizaje Automático I del Grado en Ciencia e Ingeniería de Datos de la Escuela Universitaria de Informática de la ULPGC. Ellenguaje de programación utilizado será Python y las prácticas se desarrollarán como cuadernos Jupyter ya que permite integrar código, gráficos y texto usando el lenguaje de marcado Markdown. El entorno de trabajo puede ser cualquiera que permita crear y ejecutar cuadernos Jupyter entre los cuales se encuentra Anaconda que incluye los paquetes básicos preinstalados, Visual Studio Code que es un editor open source, o PyCharm.

Las librerías necesarias para realizar las prácticas de esta asignatura son:

- Scikit-learn: Esta librería implementa modelos de aprendizaje automático tanto supervisado como no supervisado, además de métodos para preprocesamiento de datos, selección y evaluación de modelos entre otros. [5]
- Pandas: Es una librería orientada al análisis de datos que facilita las tareas en conjuntos de datos estructurados organizado como filas y columnas, de forma similar a una hoja de cálculo o tabla de una base de datos relacional. Las dos estructuras de datos básicas de Pandas son las `Series` y `Dataframe` para el manejo de datos unidimensionales y bidimensionales, respectivamente. [3]
- Numpy: Es una librería que proporciona la capacidad de computación numérica a Python así como estructuras de datos multidimensionales (matrices). El tipo de datos principal en Numpy es el `ndarray` que a diferencia de los dataframe de Pandas, solo pueden contener datos del mismo tipo, y la librería proporciona métodos para operar de forma eficiente con los ndarray. [2]
- Matplotlib: Esta librería permite crear visualizaciones en python tanto estáticas como animadas en Python. Además es la base de otras librerías como es Seaborn. [1]

En las prácticas donde se utilicen datos estructurados se considera que estarán almacenados en una estructura de datos organizada como una tabla de filas y columnas. Cada fila corresponderá a una muestra del problema mientras que cada columna se corresponderá con una característica o atributo. En los problemas de clasificación o regresión, una de las características (columna) será la clase a la que pertenece la muestra o el valor numérico que se desea estimar.

En Python esta estructura de datos se podrá almacenar como una lista de listas, cada elemento (muestra) con el mismo número de elementos (características), una matriz de Numpy o un Dataframe de Pandas. Todos estos tipos de datos son iterables por lo que se pueden recorrer muestra a muestra. Dependiendo del problema, las características (columnas) serán numéricas o simbólicas. En el caso de las matrices de Numpy, normalmente todos los elementos suelen ser del mismo tipo mientras que los Dataframe de Pandas pueden almacenar diferentes tipos de datos de forma similar a como lo hace una hoja de cálculo.

1. Entorno de desarrollo y librerías básicas

1.1 Librería Pandas

A continuación se enumeran algunos ejercicios para realizar utilizando la librería Pandas. Para conocer los argumentos de las funciones así como a ejemplos de utilización, se recomienda consultar la documentación del paquete [3]. Todos los ejercicios se basan en el archivo csv que se crea en el primer ejercicio.

1. Crear un archivo csv con el siguiente contenido.

```
marca,modelo,cilindrada,kms,precio,año
Seat,Ibiza,1400,59979,7728,2021
Seat,Ibiza,1400,29707,16691,2011
VW,Polo,1600,84461,6638,2010
Opel,Corsa,1400,98292,11695,2018
VW,Golf,1600,93720,10642,2022
Seat,Arona,1600,90083,14982,2021
VW,Polo,2000,57076,14271,2014
Opel,Corsa,2000,42932,18689,2014
Seat,Leon,2000,70544,7874,2013
Seat,Arona,1800,52457,8003,2013
Opel,Corsa,1800,75277,7230,2022
Opel,Corsa,1800,63751,17087,2018
Seat,Ibiza,1600,24698,11870,2022
Opel,Astra,1800,47881,6114,2013
Opel,Corsa,1800,77320,6239,2022
```

2. Leer el archivo anterior y mostrar el número de elementos que posee, el número de

columnas y el nombre de las mismas.

3. Mostrar de las 5 primeras filas solo la marca y el modelo.
4. Obtener cuantas marcas diferentes y cuales son.
5. Mostrar solo las filas de la marca Opel.
6. Calcular los kms promedio de cada una de las marcas.
7. Sumar 1000 euros a los vehículos de año posterior al 2015.

1.2 Librería Numpy

Lista de ejercicios basados en la librería Numpy. La descripción de todas las funciones disponibles en la librería y sus argumentos, así como ejemplos de utilización se pueden encontrar en la documentación [2].

1. Crear un vector de tamaño 10 de ceros excepto el quinto elemento que será 1.
2. Crear un vector con valores entre 10 y 49.
3. Crear una matriz de 3x3 con valores entre 0 y 8.
4. Crear una matriz identidad de dimensiones 4x4.
5. Crear un array de dimensiones 3x3x3 con valores aleatorios.
6. Crear una matriz de 5x5 con 1 en los bordes y 0 en el interior.
7. Crear un vector de tamaño 10 con valores aleatorios y modificar el valor máximo por 0.
8. ¿Cómo encontrar el valor más próximo a un escalar dado, en un vector?

1.3 Librería Matplotlib

Lista de ejercicios basados en la librería Matplotlib. Ejemplos de gráficas y la descripción de todos los argumentos de las funciones, se pueden encontrar en la documentación [1]. En algunos se utilizan los datos del archivo csv creado en el primer ejercicio del apartado 1.1

1. Dibujar la función $f(x) = \frac{1}{x^2+1}$ en el intervalo $[-10, 10]$.
2. Añadir una rejilla a la gráfica anterior y poner nombre a los ejes.
3. Dibujar la función $f(x) = \frac{1}{e^{-x}+1}$ en color rojo en la misma gráfica que la antior función. Debe aparecer una en azul y la otra en rojo.
4. Dibujar como un diagrama de puntos los kms y el precio de los vehículos. En el eje x debe ir los kms y en el eje y el precio. Los puntos serán estrellas en color negro y los ejes estarán etiquetados.
5. Dibujar un diagrama de barras con el número de vehículos de cada marca.

2. Preprocesado básico de datos

2.1 Introducción

En muchos problemas de Aprendizaje Automático es necesario realizar un procesado de los datos antes de aplicar los métodos de aprendizaje. Esta fase se denomina como Ingeniería de Datos y es en si misma una disciplina. Aunque será cubierta con profundidad en la asignatura Análisis Exploratorio de Datos y Visualización, hay procesos muy sencillos como dividir los datos en un conjunto de entrenamiento y otro para prueba, convertir valores simbólicos en numéricos, encontrar los valores perdidos en un conjunto de datos o escalar los valores numéricos de un conjunto de datos a un determinado rango. Estas serán las tareas que se realizarán en esta práctica.

2.2 Ejercicios propuestos

Los ejercicios propuestos de esta práctica son los siguientes:

1. Descargar del campus virtual de la asignatura el archivo *datos-practica-2.zip* y descomprimirlo.
2. Utilizando los datos del archivo *iris-perdidos.csv*, sustituir para cada variable los valores perdidos por el promedio de los valores de dicha variable.
3. Implementar una función que acepte un array unidimensional y dos valores reales, `minimo` y `maximo`, y realice el escalado del array entre los valores `minimo` y `maximo`.
4. Implementar una clase en Python que permita convertir valores simbólicos a numéricos. La clase además del constructor debe poseer los métodos `convertir`,

`convertir_inversa`. El primer método admite un array con valores simbólicos y devuelve otro array con los valores numéricos. El segundo método debe tomar como entrada un array que previamente fue convertido a numérico y devolver el correspondiente con valores simbólicos.

3. Regresión Lineal I

3.1 Introducción

Dentro de los modelos predictivos regresivos, quizás uno de los más básicos sea la regresión lineal. Este modelo se basa en considerar que el valor de la variable dependiente a predecir (y) se puede obtener como una combinación lineal $f(\mathbf{x})$ de las variables independientes o características ($\mathbf{x} = (x_1, x_2, \ldots, x_n)$).

$$y = f(\mathbf{x}) = w_0 + w_1 \cdot x^1 + w_2 \cdot x^2 + \cdots + w_n \cdot x^n = w_0 + \mathbf{w}^T \cdot \mathbf{x} \tag{3.1.1}$$

Por tanto, dado un conjunto de m muestras $x_i^1, x_i^2, \ldots, x_i^n, y_i$ el objetivo es encontrar el conjunto de pesos $w_0, w_1, \ldots, w_n$ que minimice una función de pérdida J_w que depende de los valores reales (y_i) y los estimados ($\hat{y}_i = w_0 + \mathbf{w}^T \cdot \mathbf{x_i}$) por (3.1.1). Una función de pérdida muy utilizada en los modelos de regresión lineal es el error cuadrático medio, cuya expresión es la siguiente:

$$J_w = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{3.1.2}$$

La minimización de la función de pérdida (3.1.2) se puede realizar por diferentes métodos cuyo estudio queda fuera del objetivo de esta práctica y para lo que se remite al estudiante a la bibliografía de la asignatura.

3.2 Ejercicios propuestos

1. Descargar del campus virtual de la asignatura el archivo *datos-practica-3.zip* y descomprimirlo.

2. Mostrar mediante gráficos de dispersión el valor en cada punto de cronometraje con el tiempo final en meta.
3. Realizar la partición del conjunto de datos contenido en el archivo *tiempos.csv* en dos: conjunto de entrenamiento (70 %) y conjunto de test (30 %).
4. Ajustar un modelo de regresión lineal y obtener la raíz cuadrada del error cuadrático medio entre el valor estimado y el real para el conjunto de entrenamiento tanto centrando las muestras como sin centrarlas.
5. Mostrar mediante un gráfico de barras el valor de los coeficientes de la regresión lineal para los dos casos anteriores.

4. Regresión Lineal II

4.1 Descripción del conjunto de datos

Los datos que se utilizarán para resolver estos ejercicios es el `precio-casas.csv` que se encuentra en campus virtual de la asignatura *datos-practica-4.zip*. Los datos son el conjunto de datos *California Housing* [4] que contiene el valor mediana de los precios de las viviendas de grupo de bloques censales en California de 1990. Un grupo de bloques censal es el grano de información más fina por el que se muestra el censo de Estados Unidos se considera y tienen una población entre 600 y 3000 habitantes y entre 250 y 550 unidades de vivienda. El conjunto de datos está formado por 20640 observaciones (muestras) y cada una de ellas dispone del valor de ocho características y una variable respuesta. Las características son las siguientes:

- Ingresos promedio: Valor mediana de los ingresos de los habitantes del grupo.
- Mediana antiguedad: Valor mediana de la antigüedad de las viviendas que componen el bloque.
- Num. hab. promedio: Número de habitaciones promedio de las viviendas.
- Num. dorm. promedio: Número de dormitorios promedio de las viviendas.
- Poblacion: Número de pesonas que viven en el bloque.
- Ocup. promedio: Número promedio de habitantes de cada vivienda.
- Latitud: Latitud de grupo.
- Longitud Longitud del grupo.

El valor mediana del precio de las viviendas del grupo (Mediana precio) expresado en cientos de miles de dolares (100.000$).

4.2 Ejercicios propuestos

1. Descargar del campus virtual de la asignatura el archivo *datos-practica-4.zip* y descomprimirlo.
2. Calcular el coeficiente de dispersión R^2 para cada variable con la variable respuesta, y mostrar gráficamente los valores de la variable y la recta de regresión correspondiente.
3. Realizar la partición del conjunto de datos contenido del archivo *precio-casas.csv* en dos: conjunto de entrenamiento (70 %) y conjunto de test (30 %).
4. Calcular el coeficiente de dispersión para cada variable en el conjunto de entrenamiento y ordenar por valor decreciente del mismo.
5. Entrenar 8 modelos de regresión lineal sin regularización. El primero solo con la variable con mayor R^2, el segundo con las dos variables con mayores R^2, el tercero con las tres variables con mayores valores de R^2, y así sucesivamente.
6. Para cada uno de los modelos anteriores calcular el RMSE.

5. Regresión Logística

5.1 Ejercicio propuesto 1

1. Generar un conjunto de datos sintéticos utilizando la función `make_blobs` con dos características y tres clases. Las clases estarán centradas en el espacio de características en (-7,-7), (-2,10) y (5,2), y el conjunto de datos tendrá 20 muestras por clase.
2. Dividir el conjunto de datos en entrenamiento (70 %) y test (30 %) y obtener un clasificador lineal mediante Regresión Logística.
3. Mostrar gráficamente las muestras del conjunto de entrenamiento, y las superficies de decisión obtenidas mediante Regresión Logística (líneas rojas).

5.2 Ejercicio propuesto 2

1. Descargar del campus virtual de la asignatura el archivo *datos-practica-6.zip* y descomprimirlo.
2. Leer el conjunto de entrenamiento y de test de los archivos `CelebA-1K-train.csv` y `CelebA-1K-test.csv` respectivamente.
3. Entrenar un clasificador lineal biclásico utilizando el método de Regresión Logística y obtener la tasa de acierto del clasificador en las muestras del conjunto de test (utilizar la función `accuracy_score`.
4. Clasificar las imágenes que están en el archivo comprimido `ImagenesParaClasificar.zip` y cuyas características están en el conjunto de datos de test `CelebA-1K-test.csv`, indicando cuales de ellas las clasifica incorrectamente el clasificador.

6. Perceptrón

6.1 Ejercicio propuesto 1

1. Generar un conjunto de datos sintéticos con 100 muestras utilizando la función `make_blobs` con dos características y tres clases. Las clases estarán centradas en el espacio de características en ((0,0), (2,2), (2,0])) y desviación estándar para los clusters igual a 0.6.
2. Dividir el conjunto de datos en 70 % de las muestras para entrenamiento y el resto para test usando como semilla el valor 567; y obtener el clasificador mediante el método Perceptrón.
3. Dibujar las muestras de entrenamiento y las superficies de decisión.
4. Dibujar las muestras de test y las superficies de decisión.
5. Obtener la exactitud del clasificador en el conjunto de test y analizar el resultado de acuerdo a la gráfica del apartado anterior.

6.2 Ejercicio propuesto 2

1. Descargar del campus virtual de la asignatura el archivo *datos-practica-7.zip* y descomprimirlo.
2. Leer el conjunto de entrenamiento y de test de los archivos `CelebA-1K-train.csv` y `CelebA-1K-test.csv` respectivamente.
3. Entrenar un clasificador biclásico utilizando el método Perceptrón y obtener la tasa de acierto del clasificador en las muestras del conjunto de test (utilizar la función `accuracy_score` y en el conjunto de train.

4. Comparar los resultados con los obtenidos utilizando Regresión Logística del ejercicio 5.2.

7. K Vecinos más cercanos

7.1 Ejercicio propuesto 1

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica8.zip*.
2. Leer el conjunto de datos *precio-casas-clasificacion.csv*, donde la clase se corresponde con la variable 'Precio' y tiene 5 valores simbólicos ('muy alto', 'alto', 'promedio', 'bajo', 'muy bajo')
3. Dividir los datos en entrenamiento y test.
4. Fijar el valor de k=5 y calcular el tiempo total de entrenamiento y test, solo el de entrenamiento y solo el de test, según los diferentes algoritmos ('brute', 'kd_tree', 'ball_tree'). Analizar los resultados.
5. Obtener la tasa de acierto para el conjunto de test para k de 1 a 10 utilizando una votación uniforme o basada en la distancia (argumento de la clase 'weights'). Analizar los resultados.

7.2 Ejercicio propuesto 2

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica8.zip*
2. Leer el conjunto de datos *cliente_telefono.csv* que se corresponde con información de clientes de una compañía telefónica que ha dividido a los clientes en cuatro categorías ('categoria_1', 'categoria_2', 'categoria_3', 'categoria_4') y se corresponde con la clase en este ejercicio.
3. Dividir los datos en entrenamiento y test.
4. Obtener la exactitud en el conjunto de test para valores de k de 1 a 10.

5. Normalizar las características con valores entre 0 y 1, repetir el apartado anterior y comparar los resultados.
6. Obtener la exactitud en el conjunto de aprendizaje mediante Regresión Logística y un Perceptrón y comparar los resultados con los obtenidos en los dos apartados anteriores.

8. Máquina de Vectores Soporte (SVM)

8.1 Ejercicio propuesto 1

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica9.zip*.
2. Leer el conjunto de datos `datos_svm.csv`.
3. Obtener un clasificador SVM lineal para cada uno de los siguientes valores de `C`, {0.01, 1, 10}, con todas las muestras del conjunto de datos.
4. Para cada uno de los tres clasificadores anteriores de debe mostrar gráficamente (un gráfico por clasificador):
   a) Mestras utilizadas para entrenar
   b) Superficies de decisión
   c) Vectores soporte de cada clase en un color diferente (utilizar el atributo *n_support_* de la clase)
5. Analizar los resultados obtenidos dando una explicación de los mismos.

8.2 Ejercicio propuesto 2

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica9.zip*.
2. Leer el conjunto de entrenamiento y de test de los archivos `CelebA-1K-train.csv` y `CelebA-1K-test.csv` respectivamente.
3. Dividir los datos de entrenamiento (CelebA-1K-train.csv) en entrenamiento (80 %) y validación (20 %).
4. Entrenar clasificadores basados en SVM utilizando los kernel lineal, polinomial y función de base radial con diferentes valores para el coeficiente de regularización `C`.

5. Para la mejor combinación de núcleo y coeficiente de regularización, calcular la tasa de acierto en el conjunto de test.
6. Entrenar clasificadores basados en regresión logística y otro en el método Perceptrón, y comparar los resultados con los del apartado anterior.

9. Clasificador Bayesiano Ingenuo

9.1 Ejercicio propuesto 1

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica10.zip*.
2. Dividir el conjunto de datos `spam_preprocesado_binario.csv` en entrenamiento y test.
3. Obtener la exactitud de un clasificador ingenuo gaussiano para el caso anterior.
4. Repetir el apartado anterior para los clasificadores ingenuo bernoulli y multinomial.
5. Repetir los apartados 2, 3 y 4 para los conjuntos de datos `spam_preprocesadocsv` y `spam_preprocesado_reales.csv`.

9.2 Ejercicio propuesto 2

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica10.zip*.
2. Dividir el conjunto de datos `spam_preprocesado_reales.csv` en entrenamiento y test.
3. Obtener la exactitud para los clasificadores: Perceptron, Regresión Logística, SVM y Vecinos más Cercanos.
4. Comparar con los resultados obtenidos en el ejercicio propuesto 1.

10. Estimación de rendimiento

10.1 Ejercicio propuesto 1

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica9.zip*.
2. Entrenar cinco clasificadores diferentes de los estudiados en las clases prácticas para el problema `CelebA-1K`.
3. Obtener y mostrar gráficamente las matrices de confusión para cada uno de ellos.
4. A partir de los valores de la matriz de confusión calcular: tasa de acierto (accuracy), sensibilidad (recall), precisión (precision), especifidad y medida F1, para cada uno de los clasificadores.

10.2 Ejercicio propuesto 2

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica9.zip*.
2. Entrenar cinco clasificadores diferentes de los estudiados en las clases prácticas para el problema `CelebA-1K`.
3. Mostrar en el espacio ROC los resultados de los clasificadores anteriores. ¿Cual es el mejor según su representación en el espacio ROC?

10.3 Ejercicio propuesto 3

1. Descargar del Campus Virtual de la asignatura el archivo *datos-practica9.zip*.
2. Combinar los dos conjuntos de datos `CelebA-1K-train.csv` y `CelebA-1K-test.csv` en un único conjunto de datos.

3. Dividir el conjunto resultante anterior en tres subconjuntos: entrenamiento, validación y test.
4. Seleccionar la combinación óptima del hiperparámetro de regularización C y núcleo para un clasificador SVM y obtener el rendimiento.
5. Repetir la tarea anterior para el clasificador de los vecinos más cercanos para obtener la combinación óptima de número de vecinos K y pesado (*distance*, *uniform*).

Bibliografía

[1] Matplotlib. *Librería Matplotlib*. https://matplotlib.org/stable/index.html. Accessed: 17-ene-2023 (véanse páginas 7, 10).

[2] Numpy. *Librería Numpy*. https://numpy.org/doc/stable/user/whatisnumpy.html. Accessed: 17-ene-2023 (véanse páginas 7, 10).

[3] Pandas. *Librería Pandas*. https://pandas.pydata.org/docs/getting_started/overview.html. Accessed: 17-ene-2023 (véanse páginas 7, 9).

[4] Statlib repository. *California Housing*. https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html. Accessed: 15-feb-2022 (véase página 15).

[5] Sklearn. *Librería Scikit-learn*. https://scikit-learn.org/stable/getting_started.html. Accessed: 17-ene-2023 (véase página 7).