1. Objetivo del laboratorio

*Desarrollar de forma autónoma distintas implementaciones de **redes neuronales de aprendizaje supervisado** que permitan resolver distintos casos de uso. La práctica comenzará con la construcción de un **MLP** capaz de hacer predicciones para un caso concreto planteado y continúa con el desarrollo de un modelo de DNN para clasificación de imágenes usando redes de convolución.*

Adicionalmente se plantea la práctica P3 (totalmente voluntaria) cuya calificación se sumará a las dos anteriores (la nota de este laboratorio es, por tanto, sobre 12 puntos) y que solo se corregirá si se han resuelto las prácticas P1 y P2.

2. Elementos a utilizar:

- Lenguaje Python
- Librerías: numérica *NumPy*, estructuras de datos *pandas*, gráfica *Matplotlib* (opcional si se quieren implementar gráficas) y redes neuronales *Keras* y *Tensorflow*.
- Entorno Anaconda
- Editor Jupyter
- Archivos .csv proporcionados

3. Práctica 1 (MLP multicapa con Keras: supervivientes del Titanic)

Objetivo

Utiliza la librería Keras para construir y entrenar un MLP para predecir si los pasajeros del Titanic, en base a varias características, sobreviven o por el contrario perecen. En vez de usar la Regla Delta Generalizada, usaremos **Adam** como función de modificación de matriz de pesos (*optimizer*) de la forma que se indica en el apartado de "**Implementación**". Responde a las preguntas que se plantean en "**Cuestiones**".

Implementación

Crea el notebook *L3P1-Titanic.ipynb*. El programa en Python deberá responder a los siguientes puntos:

1. Crea un MLP que has de **entrenar** y **validar** con la información del archivo *''titanic.csv'*. Usar como optimizer *'Adam'*, funciones de activación *'ReLU'* y *'sigmoide* y función de error *'Mean Squared Error'*. Prueba distintas arquitecturas (en número de capas y número de neuronas por capa).
   Los valores que se usarán para entrenar serán:
   - Clase en la que viajaba el pasajero (PClass)
   - Sexo
   - Edad
   - Tarifa
   - Cubierta en la que se encontraba el camarote (Embarked)

   El valor que nos indica la posibilidad de sobrevivir de un personaje es '**survived**', con valores 0 o 1.
   Notas:
   - Categorizar la variable *Embarked* y *Sexo* usando el método de keras *to_categorical*.
   - Tener en cuenta que algunas edades no vienen en el dataset y se verán como NaN. Sustituir dichos valores por la media de edad del resto de pasajeros usando la función de pandas *mean()*.
2. Una vez creada y entrenada la red neuronal coge la mejor arquitectura y con ella usa los valores del archivo *'predict_titanic.csv'* para predecir que les pasa a los pasajeros cuyos datos están recogidos en ese dataset.

Cuestiones

Elabora una memoria de la práctica en la que respondas a las siguientes cuestiones

1. Explica cómo has llevado a cabo la normalización de los datos de entrada

2. ¿Cuál es la mejor arquitectura? Justifícalo con una tabla que recoja los valores de *loss* y *accuracy* para el conjunto de entrenamiento y el de validación para las distintas pruebas que has llevado a cabo.
3. Con esa arquitectura, determina el error que se obtiene para el dataset *predict_titanic* usando los valores de *survive* de ese dataset.
4. ¿Es verdad aquello de las mujeres y los niños primero? ¿Hay más posibilidades siendo pasajero de primera clase (se consideran de primera clase aquellos billetes de más de 500 libras) debido a que los camarotes están más cerca de la cubierta? Justifícalo en base a los resultados de la predicción

4. Práctica 2 (DNN para clasificar imágenes)

Objetivo

Utiliza lo aprendido en clase respecto a Deep Learning y redes convolucionales para construir un clasificador de imágenes. El dataset usado será CIFAR-10, un dataset de imágenes a color de tamaño 32x32 que están clasificadas en 10 categorías. Crea una red convolucional de la forma que se indica en el apartado de "**Implementación**". Responde a las preguntas que se plantean en "**Cuestiones**".

Implementación

Crea el notebook *L3P2-Imagenes.ipynb*. El programa en Python deberá responder a los siguientes puntos:

1. Crea una red convolucional secuencial. Juega con los tamaños de los filtros y decide el tamaño de las capas de '*pooling*'. Utiliza al menos 3 capas de convolución. En el caso de las funciones de activación, lo normal en este tipo de redes es usar *ReLU* en todas las capas, menos en la de salida que se debe de usar *softmax* para clasificaciones no binarias. Para actualizar los pesos, usaremos el optimizer '*Adam*' y la función de error '*categorical_crossentropy*'. Prueba con distintos 'learning rates'.
   Notas:
   - El dataset usado será CIFAR-10 que debes cargar usando la función *load_data()* de keras
   - En caso de que fuese necesario, normalizar los datos.
2. Desarrolla las distintas fases de un predictor: entrenamiento, validación y predicción. Esta última se hará con las imágenes almacenadas en la carpeta '**imágenes'**. Para ello es necesario mostrar por pantalla, para cada imagen:
   - Las tres categorías en las que con mayor probabilidad la imagen es clasificada por la red, en orden descendente de probabilidad
   - La propia imagen pintada

Cuestiones

Continúa en la memoria respondiendo a las siguientes cuestiones

1. Entrena la red usando como criterio de parada un **loss ≤ 0,2**. Recoge en la memoria los valores de *loss* y *accuracy* tanto del conjunto de entrenamiento como del de validación de los distintos experimentos que hayas llevado a cabo indicando la arquitectura de cada una de las redes empleadas. Dibuja la arquitectura de red que mejor clasifica.
2. Clasifica el conjunto de imágenes de prueba y recoge en la memoria la salida obtenida para cada imagen tal y como se indica en el apartado **implementación**.

5. Práctica 3 (DNN para clasificar imágenes)

Objetivo

Utilizando tu propio criterio, selecciona, explora y analiza un dataset de entre los disponibles en KAGGLE.

Implementación

Crea el notebook *L3P3-kaggle.ipynb*. El dataset lo deberás de seleccionar de www.kaggle.com

Cuestiones

1. Motivación del problema seleccionado y justificación de la solución que se quiere obtener.
2. Resultados obtenidos

6. Forma de entrega del laboratorio:

La entrega consistirá en un fichero comprimido RAR con nombre LAB03-GRUPOxx.RAR subido a la tarea **LAB2** que **contenga únicamente**

1. **Por cada práctica** un notebook de Jupyter (archivos con extensión **.ipynb**).
2. Una **memoria del laboratorio** en Word.

Las entregas que no se ajusten exactamente a esta norma NO SERÁN EVALUADAS.

7. Rúbrica de la Práctica:

1. IMPLEMENTACIÓN: Multiplica la nota del trabajo por 0/1

Siendo una práctica de IA, todos los aspectos de programación se dan por supuesto. La implementación será:

- Original: Código fuente no copiado de internet. Grupos con igual código fuente serán suspendidos
- Correcta: Los algoritmos SOM están correctamente programados. El programa funciona y ejecuta correctamente todo lo planteado en el apartado "*Cuestiones*" de cada práctica.
- Comentada: Inclusión (**obligatoria**) de comentarios.

2. MEMORIA DEL LABORATORIO

Obligatorio redacción clara y correcta ortográfica/gramaticalmente con la siguiente estructura:

- *Portada con el nombre de los componentes del grupo y el número del grupo*
- *Índice*
- *Resultados de la Práctica 1*
- *Resultados de la Práctica 2*
- *Resultados de la Práctica 3 (optativa)*
- *Discusión general de la práctica*
- *Bibliografía*

Calificación de las cuestiones:

PRÁCTICA	CUESTIÓN	VALORACIÓN (sobre 12)
Práctica 1	Cuestión 1	0,5
	Cuestión 2	3
	Cuestión 3	0,5
	Cuestión 4	2
Práctica 2	Cuestión 1	2
	Cuestión 2	2
Práctica 3	Cuestión 1	1
	Cuestión 2	1