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 implementación más sencilla del funcionamiento de una neurona sencilla y terminará con la construcción de un **MLP** capaz de hacer predicciones para un caso concreto planteado.*

2. Elementos a utilizar:

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

3. Práctica 1 (Perceptrón para funciones lógicas)

Objetivo

Resolver funciones lógicas usando un modelo neuronal simple. Deberás codificar un modelo neuronal desde cero según lo visto en teoría y las instrucciones del apartado "**Implementación**". Responde a las preguntas que se plantean en "**Cuestiones**". Debes de usar comentarios con profusión, incluyendo celdas específicas donde expliques los algoritmos que usas y el código que has programado.

Implementación

Crea el notebook *L2P1-Perceptron.ipynb*. Programa un PERCEPTRÓN cuyos pesos iniciales tendrán un valor al azar entre -1.0 y 1.0 ambos incluidos y que se entrenará usando la Ley de Hebb.

1. Resuelve la función **AND**. Prueba diferentes *learning rates* y umbrales para saber cuál es el óptimo. Sacar por pantalla las salidas de la red neuronal y el valor del error para cada iteración.
2. Repetir el ejercicio anterior para la función **XOR**.

Entregar los resultados de ambas pruebas en el archivo *L2P1-Perceptron.csv*

Cuestiones

1. Elabora una memoria de la práctica con una tabla para cada caso. Discútelas según lo visto en teoría. Escribe de forma explícita la ecuación del hiperplano que se ha generado en cada uno. Las tablas tendrán la estructura

Iteración	Entradas		Pesos iniciales		Yr	Yd	Error	Pesos finales	
	X1	X2	W1	W2				W1	W2

4. Práctica 2 (MLP con Keras para funciones lógicas)

Objetivo

Utiliza la librería Keras/TensorFlow2 para construir y entrenar un MLP que resuelva problemas no lineales. 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 *L2P2-MLP.ipynb*. Usando la API de Keras, crea un modelo neuronal *'fully connected'* y *secuencial*. El modelo tendrá que resolver la función **XOR**. Prueba con distinto número de neuronas en la capa oculta (estima el rango de variación según lo visto en teoría) y varios valores de *epoch*. Haz al menos 3

modelos intentando optimizar los resultados. Usar como optimizer '*Adam'*, funciones de activación '*ReLU'* y '*sigmoide'*, y función de error '*Mean Squared Error'*.

Notas:

- El '*learning rate*' es un atributo de la clase 'optimizer'.
- Usa arrays de *numpy* para los valores de entrenamiento y sus salidas de predicción.
- API de Keras: https://keras.io/
- Adam: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Cuestiones

1. Continúa en la memoria creando varias tablas (como la siguiente) con los resultados del MLP que resuelve la función. Cada fila de la tabla será una de las variaciones que hayas probado para ese modelo. Haz una tabla distinta para los experimentos con cada una de las dos funciones de activación propuestas

Neuronas capa oculta	Epochs	Salida	Error

5. Práctica 3 (Predicción de sufrir un infarto con Keras)

Objetivo

Utiliza lo practicado en este laboratorio para resolver un caso práctico "real" usando la librería Keras/TensorFlow2 y una arquitectura MLP para predecir si una persona va a sufrir un infarto o no. La variable objetivo del dataset (la que indica si sufre el infarto o no) es la última columna (infarto).

Implementación

Crea el notebook *L2P3-MLP_Infarto.ipynb*. Usa como dataset el fichero *infarto.csv* que contiene 12 columnas. La última columna indica si el individuo ha sufrido un infarto o no. El resto son variables categóricas y numéricas que puedes emplear para hacer la predicción.

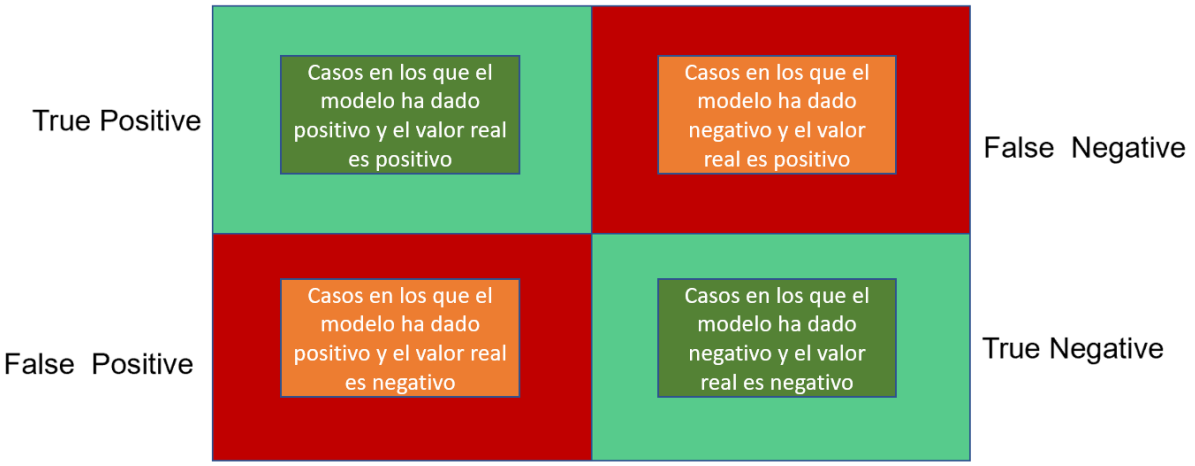
El programa en Python deberá responder a los siguientes puntos:

1. Importar el dataset y convertir las variables categóricas a numéricas usando el método de *keras to_categorical*. Eliminar aquellas columnas que no sean necesarias para hacer una predicción.
2. Normalizar todas las variables.
3. Obtener un conjunto de prueba que sea el 20% del dataset original. Para ello usa *train_test_split()* disponible en la librería *scikit-learn*.
4. Crear un MLP que has de **entrenar** y **validar** con el resto de los datos del dataset inicial. Usa como optimizer '*Adam'* en vez de usar la Regla Delta Generalizada, funciones de activación '*ReLU'* y '*sigmoide'* y función de error '*Mean Square Error'*.

Cuestiones

Continúa en la memoria. Usa tablas y gráficas según tu criterio.

1. Explica cómo has llevado a cabo la categorización y la normalización de los datos de entrada
2. Prueba distintas arquitecturas (en **número de capas** y **número de neuronas por capa**) ¿Cuál es la mejor arquitectura? Dibújala y justifícala 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. Utiliza también las métricas *precision* y *recall*. Compáralas con los valores de *accuracy* ¿Que puedes concluir de ello? ¿Porqué crees que es útil en un campo como la medicina tener en cuenta estas dos métricas? Ten en cuenta los conceptos de falso positivo y falso negativo.
4. Con esa arquitectura, determina el número de aciertos que se obtiene para el dataset de prueba y obtén la matriz de confusión e interprétala. Utiliza *confusion_matrix()* de la librería *scikit-learn*. ¿Qué conclusiones extraes mirando la matriz de confusión?



5. ¿Detectas algún problema en tus datos? En caso de existir, modifica los datos para que esto no ocurra y vuelve a entrenar tu modelo.
6. ¿Qué variables influyen más a la hora de que un paciente padezca un infarto? ¿Qué variables influyen menos? Describe el método que has llevado a cabo para llegar a esa conclusión.

6. Forma de entrega del laboratorio:

La entrega consistirá en un fichero comprimido RAR con nombre LAB02-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*
- *Bibliografía*

Calificación de las cuestiones:

PRÁCTICA	CUESTIÓN	VALORACIÓN (sobre 10)
Práctica 1	Cuestión 1	0.75
Práctica 2	Cuestión 1	0.75
Práctica 3	Cuestión 1	1

	Cuestión 2	0,5
	Cuestión 3	2
	Cuestión 4	2
	Cuestión 5	2
	Cuestión 6	1