1. Cargar el fichero de datos.

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from keras.layers.advanced_activations import LeakyReLU
from sklearn.preprocessing import MinMaxScaler
```

```
diabetes = pd.read_table('/content/drive/MyDrive/Colab Notebooks/diabetesnn', delimiter="
print(diabetes)
```

```
     pregnant   glucose  pressure   triceps   insulin      mass  pedigree  \
1    0.352941  0.670968  0.489796  0.304348  0.170130  0.314928  0.234415
2    0.058824  0.264516  0.428571  0.239130  0.170130  0.171779  0.116567
3    0.470588  0.896774  0.408163  0.240798  0.170130  0.104294  0.253629
4    0.058824  0.290323  0.428571  0.173913  0.096154  0.202454  0.038002
5    0.000000  0.600000  0.163265  0.304348  0.185096  0.509202  0.943638
..        ...       ...       ...       ...       ...       ...       ...
764  0.588235  0.367742  0.530612  0.445652  0.199519  0.300613  0.039710
765  0.117647  0.503226  0.469388  0.217391  0.170130  0.380368  0.111870
766  0.294118  0.496774  0.489796  0.173913  0.117788  0.163599  0.071307
767  0.058824  0.529032  0.367347  0.240798  0.170130  0.243354  0.115713
768  0.058824  0.316129  0.469388  0.260870  0.170130  0.249489  0.101196

          age  class
1    0.483333      1
2    0.166667      0
3    0.183333      1
4    0.000000      0
5    0.200000      1
..        ...    ...
764  0.700000      0
765  0.100000      0
766  0.150000      0
767  0.433333      1
768  0.033333      0
```

```
[768 rows x 9 columns]
```

```
display(diabetes.describe().transpose())
```

	count	mean	std	min	25%	50%	75%	max
pregnant	768.0	0.226180	0.198210	0.0	0.058824	0.176471	0.352941	1.0
glucose	768.0	0.501205	0.196361	0.0	0.359677	0.470968	0.620968	1.0
pressure	768.0	0.493930	0.123432	0.0	0.408163	0.491863	0.571429	1.0
triceps	768.0	0.240798	0.095554	0.0	0.195652	0.240798	0.271739	1.0
insulin	768.0	0.170130	0.102189	0.0	0.129207	0.170130	0.170130	1.0
mass	768.0	0.291564	0.140596	0.0	0.190184	0.290389	0.376278	1.0
pedigree	768.0	0.168179	0.141473	0.0	0.070773	0.125747	0.234095	1.0
age	768.0	0.204015	0.196004	0.0	0.050000	0.133333	0.333333	1.0
class	768.0	0.348958	0.476951	0.0	0.000000	0.000000	1.000000	1.0

5. Separar train y test.

```
# Reparto de datos en train y test de los datos normalizados (67/33)
from sklearn.model_selection import train_test_split

labels = diabetes['class']
labels = labels.astype('int64')

diabetes_train, diabetes_test, labels_train, labels_test = train_test_split(diabetes.loc[:
```

Primera red neuronal profunda 2 capas de 12 y 6 nodos:

```
model1 = Sequential()
# Primera capa oculta densa. Incluye la capa de lectura de dimensión␣
len(diabetes_train.columns)
model1.add(Dense(12, input_dim=len(diabetes_train.columns),activation='relu'))
# Segunda capa oculta
model1.add(Dense(6,activation='relu'))
model1.add(Dense(1,activation='sigmoid'))
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 12)                108

 dense_1 (Dense)             (None, 6)                 78
 dense_2 (Dense)             (None, 1)                 7

=================================================================
Total params: 193
Trainable params: 193
Non-trainable params: 0
_________________________________________________________________
```

```
# ajustes del modelo
model1.compile(optimizer="adam", loss=keras.losses.binary_crossentropy, metrics=["accuracy
```

```
x1 = model1.fit(diabetes_train, labels_train,
batch_size= 16, epochs= 60,
verbose=1,
validation_split= 0.33)
```

```
Epoch 33/60
22/22 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0
Epoch 34/60
22/22 [==============================] - 0s 4ms/step - loss: 0.5001 - accuracy: 0
Epoch 35/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4942 - accuracy: 0
Epoch 36/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4910 - accuracy: 0
Epoch 37/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4878 - accuracy: 0
Epoch 38/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4865 - accuracy: 0
Epoch 39/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4821 - accuracy: 0
Epoch 40/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4837 - accuracy: 0
Epoch 41/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4768 - accuracy: 0
Epoch 42/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4764 - accuracy: 0
Epoch 43/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4736 - accuracy: 0
Epoch 44/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4733 - accuracy: 0
Epoch 45/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4701 - accuracy: 0
Epoch 46/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4682 - accuracy: 0
Epoch 47/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4673 - accuracy: 0
Epoch 48/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4662 - accuracy: 0
Epoch 49/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4657 - accuracy: 0
Epoch 50/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4650 - accuracy: 0
Epoch 51/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4614 - accuracy: 0
Epoch 52/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4610 - accuracy: 0
Epoch 53/60
```

```
22/22 [==============================] - 0s 3ms/step - loss: 0.4673 - accuracy: 0
Epoch 54/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4641 - accuracy: 0
Epoch 55/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4611 - accuracy: 0
Epoch 56/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4615 - accuracy: 0
Epoch 57/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4597 - accuracy: 0
Epoch 58/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4571 - accuracy: 0
Epoch 59/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4564 - accuracy: 0
Epoch 60/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4557 - accuracy: 0
```

```
rend1 = model1.evaluate(diabetes_test, labels_test,verbose=0)

res1 = [["Loss" ,"Accuracy"], [round(rend1[0],3),round(rend1[1],3)]
]
df=pd.DataFrame(res1)
print('\n'.join(df.to_string(index = False).split('\n')[1:]))
```

```
Loss Accuracy
0.44    0.795
```

```
prediction_m1 = model1.predict(diabetes_test)

prediction_m1_binary = (prediction_m1 > 0.5).astype("int32")
```

```
from sklearn.metrics import confusion_matrix, precision_score, cohen_kappa_score, classifi
test_predicted_labels_1 = np.round(model1.predict(diabetes_test)).astype("int32")
conf_matrix_1 = confusion_matrix(labels_test, test_predicted_labels_1)
```

```
Accuracy1=(conf_matrix_1[0,0]+conf_matrix_1[1,1])/(conf_matrix_1[0,0]+conf_matrix_1[1,1]+c
print ('Accuracy:',round(Accuracy1,4))
Sensibilidad1= conf_matrix_1[1,1]/(conf_matrix_1[1,1]+conf_matrix_1[1,0])
print ('Sensibilidad:',round(Sensibilidad1,4))
Especificidad1= conf_matrix_1[0,0]/(conf_matrix_1[0,0]+conf_matrix_1[0,1])
print ('Especificidad:',round(Especificidad1,4))
```

```
Accuracy: 0.7953
Sensibilidad: 0.506
Especificidad: 0.9357
```

Segunda red neuronal profunda 2 capas de 20 y 10 nodos:

```
model2 = Sequential()
# Primera capa oculta densa. Incluye la capa de lectura de dimensión␣
len(diabetes_train.columns)
```

```
model2.add(Dense(20, input_dim=len(diabetes_train.columns),activation='relu'))
# Segunda capa oculta
model2.add(Dense(10,activation='relu'))
model2.add(Dense(1,activation='sigmoid'))
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                180

 dense_4 (Dense)             (None, 10)                210

 dense_5 (Dense)             (None, 1)                 11

=================================================================
Total params: 401
Trainable params: 401
Non-trainable params: 0
_________________________________________________________________
```

```
# ajustes del modelo
model2.compile(optimizer="adam", loss=keras.losses.binary_crossentropy, metrics=["accuracy
```

```
x2 = model2.fit(diabetes_train, labels_train,
batch_size= 16, epochs= 60,
verbose=1,
validation_split= 0.33)
```

```
Epoch 33/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4761 - accuracy: 0
Epoch 34/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4792 - accuracy: 0
Epoch 35/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4729 - accuracy: 0
Epoch 36/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4722 - accuracy: 0
Epoch 37/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4707 - accuracy: 0
Epoch 38/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4688 - accuracy: 0
Epoch 39/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4683 - accuracy: 0
Epoch 40/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4663 - accuracy: 0
Epoch 41/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4659 - accuracy: 0
Epoch 42/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4660 - accuracy: 0
Epoch 43/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4644 - accuracy: 0
Epoch 44/60
22/22 [==============================] - 0s 5ms/step - loss: 0.4663 - accuracy: 0
Epoch 45/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4622 - accuracy: 0
Epoch 46/60
```

```
22/22 [==============================] - 0s 3ms/step - loss: 0.4627 - accuracy: 0
Epoch 47/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4613 - accuracy: 0
Epoch 48/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4608 - accuracy: 0
Epoch 49/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4621 - accuracy: 0
Epoch 50/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4620 - accuracy: 0
Epoch 51/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4588 - accuracy: 0
Epoch 52/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4586 - accuracy: 0
Epoch 53/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4627 - accuracy: 0
Epoch 54/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4595 - accuracy: 0
Epoch 55/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4605 - accuracy: 0
Epoch 56/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4577 - accuracy: 0
Epoch 57/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4555 - accuracy: 0
Epoch 58/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4594 - accuracy: 0
Epoch 59/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4569 - accuracy: 0
Epoch 60/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4545 - accuracy: 0
```

```
rend2 = model2.evaluate(diabetes_test, labels_test,verbose=0)

res2 = [["Loss" ,"Accuracy"], [round(rend1[0],3),round(rend1[1],3)]
]
df=pd.DataFrame(res2)
print('\n'.join(df.to_string(index = False).split('\n')[1:]))
```

```
Loss Accuracy
0.44    0.795
```

```
prediction_m2 = model2.predict(diabetes_test)

prediction_m2_binary = (prediction_m2 > 0.5).astype("int32")
```

```
from sklearn.metrics import confusion_matrix, precision_score, cohen_kappa_score, classifi
test_predicted_labels_2 = np.round(model2.predict(diabetes_test)).astype("int32")
conf_matrix_2 = confusion_matrix(labels_test, test_predicted_labels_2)
```

```
Accuracy2=(conf_matrix_2[0,0]+conf_matrix_2[1,1])/(conf_matrix_2[0,0]+conf_matrix_2[1,1]+c
print ('Accuracy:',round(Accuracy2,4))
Sensibilidad2= conf_matrix_2[1,1]/(conf_matrix_2[1,1]+conf_matrix_2[1,0])
print ('Sensibilidad:',round(Sensibilidad2,4))
Especificidad2= conf_matrix_2[0,0]/(conf_matrix_2[0,0]+conf_matrix_2[0,1])
```

```
print ('Especificidad:',round(Especificidad2,4))
```

```
Accuracy: 0.7913
Sensibilidad: 0.5301
Especificidad: 0.9181
```

Tercera red neuronal profunda 3 capas de 20, 10 y 5 nodos:

```
model3 = Sequential()
# Primera capa oculta densa. Incluye la capa de lectura de dimensión␣
len(diabetes_train.columns)
model3.add(Dense(20, input_dim=len(diabetes_train.columns),activation='relu'))
# Segunda capa oculta
model3.add(Dense(10,activation='relu'))
# Tercera capa oculta
model3.add(Dense(5,activation='relu'))
model3.add(Dense(1,activation='sigmoid'))
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 20)                180

 dense_7 (Dense)             (None, 10)                210

 dense_8 (Dense)             (None, 5)                 55

 dense_9 (Dense)             (None, 1)                 6

=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
```

```
# ajustes del modelo
model3.compile(optimizer="adam", loss=keras.losses.binary_crossentropy, metrics=["accuracy
```

```
x3 = model3.fit(diabetes_train, labels_train,
batch_size= 16, epochs= 60,
verbose=1,
validation_split= 0.33)
```

```
Epoch 33/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4559 - accuracy: 0
Epoch 34/60
22/22 [==============================] - 0s 6ms/step - loss: 0.4479 - accuracy: 0
Epoch 35/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4478 - accuracy: 0
```

```
Epoch 36/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4455 - accuracy: 0
Epoch 37/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4455 - accuracy: 0
Epoch 38/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4418 - accuracy: 0
Epoch 39/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4457 - accuracy: 0
Epoch 40/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4486 - accuracy: 0
Epoch 41/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4436 - accuracy: 0
Epoch 42/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4451 - accuracy: 0
Epoch 43/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4427 - accuracy: 0
Epoch 44/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4421 - accuracy: 0
Epoch 45/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4402 - accuracy: 0
Epoch 46/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4373 - accuracy: 0
Epoch 47/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4377 - accuracy: 0
Epoch 48/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4377 - accuracy: 0
Epoch 49/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4388 - accuracy: 0
Epoch 50/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4382 - accuracy: 0
Epoch 51/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4404 - accuracy: 0
Epoch 52/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4391 - accuracy: 0
Epoch 53/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4369 - accuracy: 0
Epoch 54/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4372 - accuracy: 0
Epoch 55/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4411 - accuracy: 0
Epoch 56/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4349 - accuracy: 0
Epoch 57/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4344 - accuracy: 0
Epoch 58/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4380 - accuracy: 0
Epoch 59/60
22/22 [==============================] - 0s 4ms/step - loss: 0.4392 - accuracy: 0
Epoch 60/60
22/22 [==============================] - 0s 3ms/step - loss: 0.4349 - accuracy: 0
```

```
rend3 = model3.evaluate(diabetes_test, labels_test,verbose=0)

res3 = [["Loss" ,"Accuracy"], [round(rend1[0],3),round(rend1[1],3)]
]
df=pd.DataFrame(res3)
print('\n'.join(df.to_string(index = False).split('\n')[1:]))
```

```
      Loss Accuracy
      0.44    0.795
```

```
prediction_m3 = model3.predict(diabetes_test)

prediction_m3_binary = (prediction_m3 > 0.5).astype("int32")
```

```
from sklearn.metrics import confusion_matrix, precision_score, cohen_kappa_score, classifi
test_predicted_labels_3 = np.round(model3.predict(diabetes_test)).astype("int32")
conf_matrix_3 = confusion_matrix(labels_test, test_predicted_labels_3)
```

```
Accuracy3=(conf_matrix_3[0,0]+conf_matrix_3[1,1])/(conf_matrix_3[0,0]+conf_matrix_3[1,1]+c
print ('Accuracy:',round(Accuracy3,4))
Sensibilidad3= conf_matrix_3[1,1]/(conf_matrix_3[1,1]+conf_matrix_3[1,0])
print ('Sensibilidad:',round(Sensibilidad3,4))
Especificidad3= conf_matrix_3[0,0]/(conf_matrix_3[0,0]+conf_matrix_3[0,1])
print ('Especificidad:',round(Especificidad3,4))
```

```
Accuracy: 0.7913
Sensibilidad: 0.5181
Especificidad: 0.924
```

+ Código + Texto

```
Podemos decir que de las 3 redes neuronales,
rendimiento presenta.
```

Podemos decir que de las 3 redes neuronales, la primera es la que mejor rendimiento presenta.

✓ 0 s completado a las 19:54 ● ×