Análisis de Datos y Aprendizaje Máquina con Tensorflow 2.0: Perceptrón Multicapa

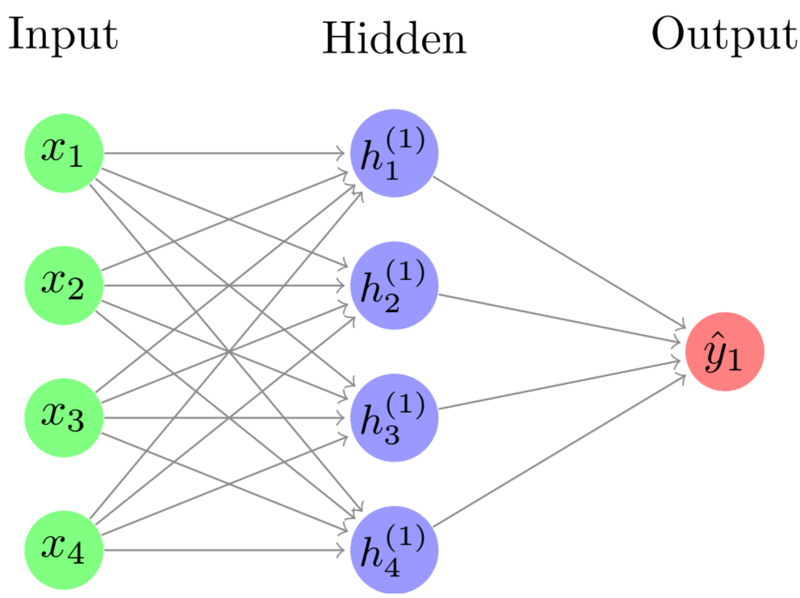
2019/09/30

1 Perceptrón Multicapa (MLP)

- Objetivo: Conocer el Perceptrón Multicapa y su implementación en Keras Tensorflow 2

1.1 Características de un MLP

- Un perceptrón multicapa aprende los valores de los parámetros que mejor minimizan alguna función de error. Esto se consigue derivando respecto a los pesos, y actualizando los valores tomando en cuenta el parámetro 'learning rate'
- Cada MLP consiste en una capa de entrada, capas ocultas, y una capa de salida, en la capa de salida cambia la función de activación dependiendo de la tarea
- Un perceptrón aproxima una función
- La red aprende de forma iterativa los parámetros. Para problemas de clasificación, el número de neuronas de salida es igual a el número de clases



- Las redes a construir para el dataset cuentan en la capa de entrada y oculta con varias neuronas, y una sola neurona en la capa de salida

```
In [1]: import matplotlib.pyplot as plt
        from sklearn.datasets import load_breast_cancer
        from sklearn.model_selection import train_test_split

        data = load_breast_cancer()

        x_data = data.data
        y_data = data.target
        x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size = 0.33, ran
```

1.2 Conjunto de datos Breast Cáncer

- Comparar el desempeño de MLP con los clasificadores tradicionales

```
In [2]: print(x_train.shape)
        print(x_test.shape)
        print(y_train.shape)
        print(y_test.shape)
```

```
(381, 30)
(188, 30)
(381,)
(188,)
```

```
In [3]: epoch = 50
        verbose = 0
        batch = 50
```

1.3 Importar tensorflow y keras

```
In [4]: import tensorflow as tf
        from tensorflow import keras
```

1.4 Agregar capas y número de neuronas

- Las capas se añaden con 'keras.layers' indicando el número de neuronas
- La función sigmoide utiliza 'binary_crossentropy' y una neurona de salida
- Se prueba con varios modelos de diferentes capas y neuronas

```
In [5]: model1 = keras.Sequential([
            keras.layers.Dense(30, input_shape=(30, ), activation = 'sigmoid'),
            keras.layers.Dense(1, activation = 'sigmoid')
        ])
```

```
In [6]: model2 = keras.Sequential([
            keras.layers.Dense(50, input_shape=(30, ), activation = 'sigmoid'),
            keras.layers.Dense(1, activation = 'sigmoid')
        ])
```

```
In [7]: model3 = keras.Sequential([
            keras.layers.Dense(30, input_shape=(30, ), activation = 'sigmoid'),
            keras.layers.Dense(30, activation = 'sigmoid'),
            keras.layers.Dense(1, activation = 'sigmoid')
        ])
```

1.5 Compilar modelo

- La función de costo y el optimizador se asignan en 'compile' antes del entrenamiento. Aquí el optimizador es 'rmsprop'

```
In [8]: model1.compile(optimizer='rmsprop',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])

        model2.compile(optimizer='rmsprop',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])

        model3.compile(optimizer='rmsprop',
                      loss='binary_crossentropy',
                      metrics=['accuracy'])
```

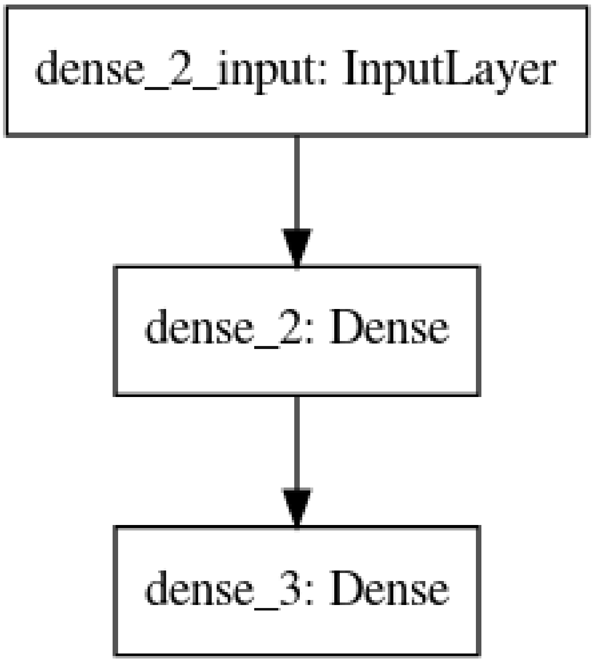
1.6 Visualizar modelos

- Se importa 'plot_model'

```
In [9]: from tensorflow.keras.utils import plot_model
```

```
In [11]: plot_model(model2)
```

```
Out[11]:
```

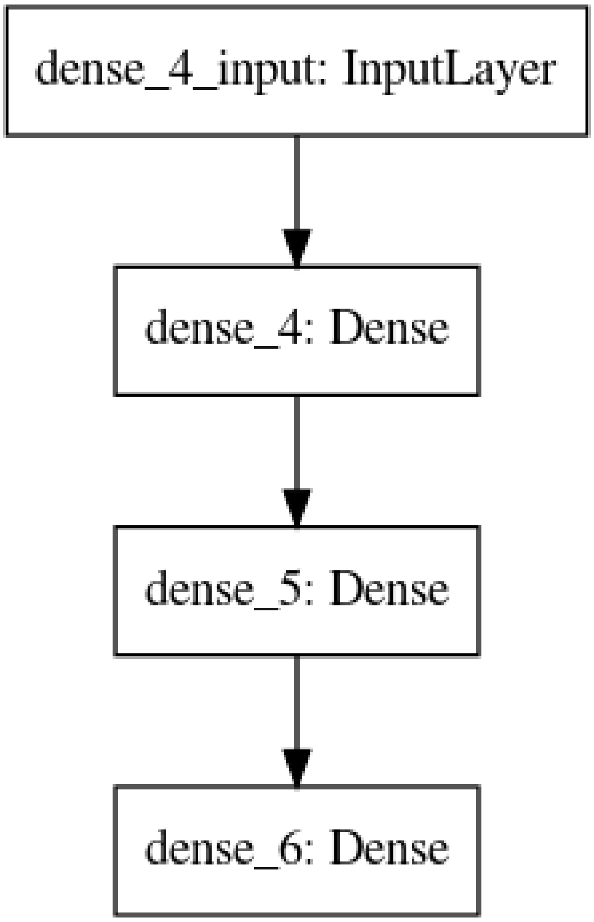



```
In [12]: model3.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 30)                930
_________________________________________________________________
dense_5 (Dense)              (None, 30)                930
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 31
=================================================================
Total params: 1,891
Trainable params: 1,891
Non-trainable params: 0
_________________________________________________________________
```

```
In [13]: plot_model(model3)

Out[13]:
```



1.7 Visualizar información del modelo

- Comparar los modelos por número de parámetros

```
In [14]: model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 30)                930
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 31
=================================================================
Total params: 961
Trainable params: 961
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [15]: `model2.summary()`

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 50)                1550
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 1,601
Trainable params: 1,601
Non-trainable params: 0
_________________________________________________________________
```

In [16]: `model3.summary()`

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 30)                930
_________________________________________________________________
dense_5 (Dense)              (None, 30)                930
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 31
=================================================================
Total params: 1,891
Trainable params: 1,891
Non-trainable params: 0
_________________________________________________________________
```

1.8 Entrenamiento

- El modelo se entrena con el método 'fit' asignando número de épocas y tamaño de batch
- El número de épocas indica la cantidad de ciclos de entrenamiento sobre todos los elementos del dataset

In [17]:
```
history1 = model1.fit(x_train, y_train, validation_split=0.33, batch_size = batch,
                             epochs = epoch, verbose = 1)
```

```
Train on 255 samples, validate on 126 samples
Epoch 1/50
255/255 [==============================] - 1s 4ms/sample - loss: 0.8334 - accuracy: 0.3608 - val_
Epoch 2/50
```

```
255/255 [==============================] - 0s 154us/sample - loss: 0.6917 - accuracy: 0.3608 - va
Epoch 3/50
255/255 [==============================] - 0s 156us/sample - loss: 0.6655 - accuracy: 0.3725 - va
Epoch 4/50
255/255 [==============================] - 0s 133us/sample - loss: 0.6448 - accuracy: 0.8745 - va
Epoch 5/50
255/255 [==============================] - 0s 131us/sample - loss: 0.6328 - accuracy: 0.8902 - va
Epoch 6/50
255/255 [==============================] - 0s 158us/sample - loss: 0.6110 - accuracy: 0.8667 - va
Epoch 7/50
255/255 [==============================] - 0s 159us/sample - loss: 0.5839 - accuracy: 0.8235 - va
Epoch 8/50
255/255 [==============================] - 0s 154us/sample - loss: 0.5744 - accuracy: 0.8039 - va
Epoch 9/50
255/255 [==============================] - 0s 173us/sample - loss: 0.5688 - accuracy: 0.8039 - va
Epoch 10/50
255/255 [==============================] - 0s 157us/sample - loss: 0.5529 - accuracy: 0.8314 - va
Epoch 11/50
255/255 [==============================] - 0s 151us/sample - loss: 0.5306 - accuracy: 0.8039 - va
Epoch 12/50
255/255 [==============================] - 0s 164us/sample - loss: 0.5265 - accuracy: 0.8078 - va
Epoch 13/50
255/255 [==============================] - 0s 178us/sample - loss: 0.5178 - accuracy: 0.8000 - va
Epoch 14/50
255/255 [==============================] - 0s 139us/sample - loss: 0.5114 - accuracy: 0.8000 - va
Epoch 15/50
255/255 [==============================] - 0s 144us/sample - loss: 0.5065 - accuracy: 0.8196 - va
Epoch 16/50
255/255 [==============================] - 0s 155us/sample - loss: 0.5027 - accuracy: 0.8039 - va
Epoch 17/50
255/255 [==============================] - 0s 155us/sample - loss: 0.4999 - accuracy: 0.8275 - va
Epoch 18/50
255/255 [==============================] - 0s 149us/sample - loss: 0.4959 - accuracy: 0.7961 - va
Epoch 19/50
255/255 [==============================] - 0s 163us/sample - loss: 0.4924 - accuracy: 0.8196 - va
Epoch 20/50
255/255 [==============================] - 0s 160us/sample - loss: 0.4882 - accuracy: 0.8196 - va
Epoch 21/50
255/255 [==============================] - 0s 163us/sample - loss: 0.4860 - accuracy: 0.8039 - va
Epoch 22/50
255/255 [==============================] - 0s 157us/sample - loss: 0.4809 - accuracy: 0.8118 - va
Epoch 23/50
255/255 [==============================] - 0s 159us/sample - loss: 0.4786 - accuracy: 0.8235 - va
Epoch 24/50
255/255 [==============================] - 0s 200us/sample - loss: 0.4729 - accuracy: 0.8314 - va
Epoch 25/50
255/255 [==============================] - 0s 198us/sample - loss: 0.4726 - accuracy: 0.8275 - va
Epoch 26/50
255/255 [==============================] - 0s 205us/sample - loss: 0.4685 - accuracy: 0.8196 - va
```

```
Epoch 27/50
255/255 [==============================] - 0s 171us/sample - loss: 0.4646 - accuracy: 0.8549 - va
Epoch 28/50
255/255 [==============================] - 0s 181us/sample - loss: 0.4647 - accuracy: 0.8431 - va
Epoch 29/50
255/255 [==============================] - 0s 178us/sample - loss: 0.4590 - accuracy: 0.8471 - va
Epoch 30/50
255/255 [==============================] - 0s 164us/sample - loss: 0.4594 - accuracy: 0.8235 - va
Epoch 31/50
255/255 [==============================] - 0s 152us/sample - loss: 0.4512 - accuracy: 0.8706 - va
Epoch 32/50
255/255 [==============================] - 0s 165us/sample - loss: 0.4524 - accuracy: 0.8706 - va
Epoch 33/50
255/255 [==============================] - 0s 144us/sample - loss: 0.4497 - accuracy: 0.8863 - va
Epoch 34/50
255/255 [==============================] - 0s 170us/sample - loss: 0.4436 - accuracy: 0.8784 - va
Epoch 35/50
255/255 [==============================] - 0s 154us/sample - loss: 0.4431 - accuracy: 0.8784 - va
Epoch 36/50
255/255 [==============================] - 0s 156us/sample - loss: 0.4379 - accuracy: 0.8863 - va
Epoch 37/50
255/255 [==============================] - 0s 154us/sample - loss: 0.4433 - accuracy: 0.8627 - va
Epoch 38/50
255/255 [==============================] - 0s 173us/sample - loss: 0.4406 - accuracy: 0.8510 - va
Epoch 39/50
255/255 [==============================] - 0s 165us/sample - loss: 0.4307 - accuracy: 0.8824 - va
Epoch 40/50
255/255 [==============================] - 0s 156us/sample - loss: 0.4295 - accuracy: 0.8784 - va
Epoch 41/50
255/255 [==============================] - 0s 174us/sample - loss: 0.4243 - accuracy: 0.9059 - va
Epoch 42/50
255/255 [==============================] - 0s 140us/sample - loss: 0.4224 - accuracy: 0.8941 - va
Epoch 43/50
255/255 [==============================] - 0s 167us/sample - loss: 0.4181 - accuracy: 0.8784 - va
Epoch 44/50
255/255 [==============================] - 0s 142us/sample - loss: 0.4127 - accuracy: 0.8863 - va
Epoch 45/50
255/255 [==============================] - 0s 146us/sample - loss: 0.4146 - accuracy: 0.8745 - va
Epoch 46/50
255/255 [==============================] - 0s 223us/sample - loss: 0.4084 - accuracy: 0.8902 - va
Epoch 47/50
255/255 [==============================] - 0s 148us/sample - loss: 0.4090 - accuracy: 0.8902 - va
Epoch 48/50
255/255 [==============================] - 0s 159us/sample - loss: 0.4066 - accuracy: 0.8784 - va
Epoch 49/50
255/255 [==============================] - 0s 159us/sample - loss: 0.4002 - accuracy: 0.8980 - va
Epoch 50/50
255/255 [==============================] - 0s 198us/sample - loss: 0.3964 - accuracy: 0.9020 - va
```

```
In [18]: history2 = model2.fit(x_train, y_train, validation_split=0.33, batch_size = batch,
                               epochs = epoch, verbose = 1)
```

```
Train on 255 samples, validate on 126 samples
Epoch 1/50
255/255 [==============================] - 1s 2ms/sample - loss: 0.6487 - accuracy: 0.6392 - val_
Epoch 2/50
255/255 [==============================] - 0s 175us/sample - loss: 0.5902 - accuracy: 0.6392 - va
Epoch 3/50
255/255 [==============================] - 0s 183us/sample - loss: 0.5687 - accuracy: 0.6588 - va
Epoch 4/50
255/255 [==============================] - 0s 184us/sample - loss: 0.5500 - accuracy: 0.6431 - va
Epoch 5/50
255/255 [==============================] - 0s 166us/sample - loss: 0.5442 - accuracy: 0.6863 - va
Epoch 6/50
255/255 [==============================] - 0s 157us/sample - loss: 0.5397 - accuracy: 0.6549 - va
Epoch 7/50
255/255 [==============================] - 0s 134us/sample - loss: 0.5308 - accuracy: 0.6863 - va
Epoch 8/50
255/255 [==============================] - 0s 149us/sample - loss: 0.5278 - accuracy: 0.6863 - va
Epoch 9/50
255/255 [==============================] - 0s 162us/sample - loss: 0.5230 - accuracy: 0.6745 - va
Epoch 10/50
255/255 [==============================] - 0s 174us/sample - loss: 0.5185 - accuracy: 0.6863 - va
Epoch 11/50
255/255 [==============================] - 0s 185us/sample - loss: 0.5161 - accuracy: 0.8157 - va
Epoch 12/50
255/255 [==============================] - 0s 172us/sample - loss: 0.5144 - accuracy: 0.8588 - va
Epoch 13/50
255/255 [==============================] - 0s 144us/sample - loss: 0.5064 - accuracy: 0.8980 - va
Epoch 14/50
255/255 [==============================] - 0s 162us/sample - loss: 0.5048 - accuracy: 0.8745 - va
Epoch 15/50
255/255 [==============================] - 0s 161us/sample - loss: 0.4955 - accuracy: 0.7843 - va
Epoch 16/50
255/255 [==============================] - 0s 143us/sample - loss: 0.4893 - accuracy: 0.8941 - va
Epoch 17/50
255/255 [==============================] - 0s 140us/sample - loss: 0.4941 - accuracy: 0.7451 - va
Epoch 18/50
255/255 [==============================] - 0s 135us/sample - loss: 0.4852 - accuracy: 0.8706 - va
Epoch 19/50
255/255 [==============================] - 0s 146us/sample - loss: 0.4804 - accuracy: 0.8902 - va
Epoch 20/50
255/255 [==============================] - 0s 130us/sample - loss: 0.4805 - accuracy: 0.8706 - va
Epoch 21/50
255/255 [==============================] - 0s 129us/sample - loss: 0.4730 - accuracy: 0.8863 - va
Epoch 22/50
255/255 [==============================] - 0s 134us/sample - loss: 0.4684 - accuracy: 0.8863 - va
Epoch 23/50
255/255 [==============================] - 0s 165us/sample - loss: 0.4629 - accuracy: 0.8980 - va
```

```
Epoch 24/50
255/255 [==============================] - 0s 120us/sample - loss: 0.4564 - accuracy: 0.8863 - va
Epoch 25/50
255/255 [==============================] - 0s 151us/sample - loss: 0.4545 - accuracy: 0.9020 - va
Epoch 26/50
255/255 [==============================] - 0s 130us/sample - loss: 0.4581 - accuracy: 0.8902 - va
Epoch 27/50
255/255 [==============================] - 0s 130us/sample - loss: 0.4458 - accuracy: 0.8980 - va
Epoch 28/50
255/255 [==============================] - 0s 140us/sample - loss: 0.4486 - accuracy: 0.8902 - va
Epoch 29/50
255/255 [==============================] - 0s 191us/sample - loss: 0.4408 - accuracy: 0.8980 - va
Epoch 30/50
255/255 [==============================] - 0s 197us/sample - loss: 0.4367 - accuracy: 0.9020 - va
Epoch 31/50
255/255 [==============================] - 0s 159us/sample - loss: 0.4356 - accuracy: 0.8902 - va
Epoch 32/50
255/255 [==============================] - 0s 166us/sample - loss: 0.4247 - accuracy: 0.9020 - va
Epoch 33/50
255/255 [==============================] - 0s 133us/sample - loss: 0.4334 - accuracy: 0.8941 - va
Epoch 34/50
255/255 [==============================] - 0s 140us/sample - loss: 0.4185 - accuracy: 0.9098 - va
Epoch 35/50
255/255 [==============================] - 0s 138us/sample - loss: 0.4235 - accuracy: 0.8980 - va
Epoch 36/50
255/255 [==============================] - 0s 139us/sample - loss: 0.4162 - accuracy: 0.9216 - va
Epoch 37/50
255/255 [==============================] - 0s 141us/sample - loss: 0.4081 - accuracy: 0.9059 - va
Epoch 38/50
255/255 [==============================] - 0s 153us/sample - loss: 0.4095 - accuracy: 0.9020 - va
Epoch 39/50
255/255 [==============================] - 0s 154us/sample - loss: 0.4008 - accuracy: 0.9098 - va
Epoch 40/50
255/255 [==============================] - 0s 124us/sample - loss: 0.4102 - accuracy: 0.8784 - va
Epoch 41/50
255/255 [==============================] - 0s 153us/sample - loss: 0.4009 - accuracy: 0.9059 - va
Epoch 42/50
255/255 [==============================] - 0s 154us/sample - loss: 0.4008 - accuracy: 0.9176 - va
Epoch 43/50
255/255 [==============================] - 0s 174us/sample - loss: 0.3885 - accuracy: 0.9098 - va
Epoch 44/50
255/255 [==============================] - 0s 191us/sample - loss: 0.3896 - accuracy: 0.9176 - va
Epoch 45/50
255/255 [==============================] - 0s 159us/sample - loss: 0.3928 - accuracy: 0.9059 - va
Epoch 46/50
255/255 [==============================] - 0s 154us/sample - loss: 0.3875 - accuracy: 0.9216 - va
Epoch 47/50
255/255 [==============================] - 0s 136us/sample - loss: 0.3902 - accuracy: 0.9216 - va
Epoch 48/50
```

```
255/255 [==============================] - 0s 138us/sample - loss: 0.3785 - accuracy: 0.9176 - va
Epoch 49/50
255/255 [==============================] - 0s 141us/sample - loss: 0.3778 - accuracy: 0.9098 - va
Epoch 50/50
255/255 [==============================] - 0s 159us/sample - loss: 0.3695 - accuracy: 0.9137 - va
```

In [19]:
```python
history3 = model3.fit(x_train, y_train, validation_split=0.33, batch_size = batch,
                          epochs = epoch, verbose = 1)
```

```
Train on 255 samples, validate on 126 samples
Epoch 1/50
255/255 [==============================] - 1s 2ms/sample - loss: 1.1683 - accuracy: 0.3608 - val_
Epoch 2/50
255/255 [==============================] - 0s 162us/sample - loss: 1.0078 - accuracy: 0.3608 - va
Epoch 3/50
255/255 [==============================] - 0s 138us/sample - loss: 0.9162 - accuracy: 0.3608 - va
Epoch 4/50
255/255 [==============================] - 0s 147us/sample - loss: 0.8459 - accuracy: 0.3608 - va
Epoch 5/50
255/255 [==============================] - 0s 161us/sample - loss: 0.7918 - accuracy: 0.3608 - va
Epoch 6/50
255/255 [==============================] - 0s 138us/sample - loss: 0.7496 - accuracy: 0.3608 - va
Epoch 7/50
255/255 [==============================] - 0s 132us/sample - loss: 0.7154 - accuracy: 0.3608 - va
Epoch 8/50
255/255 [==============================] - 0s 154us/sample - loss: 0.6943 - accuracy: 0.4824 - va
Epoch 9/50
255/255 [==============================] - 0s 159us/sample - loss: 0.6768 - accuracy: 0.7255 - va
Epoch 10/50
255/255 [==============================] - 0s 168us/sample - loss: 0.6665 - accuracy: 0.6392 - va
Epoch 11/50
255/255 [==============================] - 0s 176us/sample - loss: 0.6581 - accuracy: 0.6392 - va
Epoch 12/50
255/255 [==============================] - 0s 153us/sample - loss: 0.6497 - accuracy: 0.6392 - va
Epoch 13/50
255/255 [==============================] - 0s 132us/sample - loss: 0.6434 - accuracy: 0.6392 - va
Epoch 14/50
255/255 [==============================] - 0s 156us/sample - loss: 0.6423 - accuracy: 0.6392 - va
Epoch 15/50
255/255 [==============================] - 0s 169us/sample - loss: 0.6385 - accuracy: 0.6392 - va
Epoch 16/50
255/255 [==============================] - 0s 159us/sample - loss: 0.6367 - accuracy: 0.6392 - va
Epoch 17/50
255/255 [==============================] - 0s 140us/sample - loss: 0.6358 - accuracy: 0.6392 - va
Epoch 18/50
255/255 [==============================] - 0s 163us/sample - loss: 0.6340 - accuracy: 0.6392 - va
Epoch 19/50
255/255 [==============================] - 0s 161us/sample - loss: 0.6324 - accuracy: 0.6392 - va
Epoch 20/50
```

```
255/255 [==============================] - 0s 156us/sample - loss: 0.6297 - accuracy: 0.6392 - va
Epoch 21/50
255/255 [==============================] - 0s 174us/sample - loss: 0.6292 - accuracy: 0.6392 - va
Epoch 22/50
255/255 [==============================] - 0s 157us/sample - loss: 0.6269 - accuracy: 0.6392 - va
Epoch 23/50
255/255 [==============================] - 0s 159us/sample - loss: 0.6265 - accuracy: 0.6392 - va
Epoch 24/50
255/255 [==============================] - 0s 151us/sample - loss: 0.6245 - accuracy: 0.6392 - va
Epoch 25/50
255/255 [==============================] - 0s 137us/sample - loss: 0.6226 - accuracy: 0.6392 - va
Epoch 26/50
255/255 [==============================] - 0s 143us/sample - loss: 0.6205 - accuracy: 0.6392 - va
Epoch 27/50
255/255 [==============================] - 0s 155us/sample - loss: 0.6185 - accuracy: 0.6392 - va
Epoch 28/50
255/255 [==============================] - 0s 162us/sample - loss: 0.6166 - accuracy: 0.6392 - va
Epoch 29/50
255/255 [==============================] - 0s 140us/sample - loss: 0.6148 - accuracy: 0.6392 - va
Epoch 30/50
255/255 [==============================] - 0s 155us/sample - loss: 0.6127 - accuracy: 0.6392 - va
Epoch 31/50
255/255 [==============================] - 0s 146us/sample - loss: 0.6108 - accuracy: 0.6392 - va
Epoch 32/50
255/255 [==============================] - 0s 166us/sample - loss: 0.6092 - accuracy: 0.6392 - va
Epoch 33/50
255/255 [==============================] - 0s 174us/sample - loss: 0.6075 - accuracy: 0.6392 - va
Epoch 34/50
255/255 [==============================] - 0s 161us/sample - loss: 0.6063 - accuracy: 0.6392 - va
Epoch 35/50
255/255 [==============================] - 0s 141us/sample - loss: 0.6063 - accuracy: 0.6392 - va
Epoch 36/50
255/255 [==============================] - 0s 151us/sample - loss: 0.6005 - accuracy: 0.6392 - va
Epoch 37/50
255/255 [==============================] - 0s 163us/sample - loss: 0.5999 - accuracy: 0.6392 - va
Epoch 38/50
255/255 [==============================] - 0s 152us/sample - loss: 0.5968 - accuracy: 0.6392 - va
Epoch 39/50
255/255 [==============================] - 0s 187us/sample - loss: 0.5949 - accuracy: 0.6392 - va
Epoch 40/50
255/255 [==============================] - 0s 161us/sample - loss: 0.5932 - accuracy: 0.6392 - va
Epoch 41/50
255/255 [==============================] - 0s 155us/sample - loss: 0.5915 - accuracy: 0.6392 - va
Epoch 42/50
255/255 [==============================] - 0s 148us/sample - loss: 0.5903 - accuracy: 0.6392 - va
Epoch 43/50
255/255 [==============================] - 0s 154us/sample - loss: 0.5910 - accuracy: 0.6392 - va
Epoch 44/50
255/255 [==============================] - 0s 132us/sample - loss: 0.5865 - accuracy: 0.6392 - va
```

```
Epoch 45/50
255/255 [==============================] - 0s 170us/sample - loss: 0.5824 - accuracy: 0.6392 - va
Epoch 46/50
255/255 [==============================] - 0s 148us/sample - loss: 0.5862 - accuracy: 0.6471 - va
Epoch 47/50
255/255 [==============================] - 0s 175us/sample - loss: 0.5793 - accuracy: 0.6392 - va
Epoch 48/50
255/255 [==============================] - 0s 143us/sample - loss: 0.5740 - accuracy: 0.6392 - va
Epoch 49/50
255/255 [==============================] - 0s 165us/sample - loss: 0.5659 - accuracy: 0.7137 - va
Epoch 50/50
255/255 [==============================] - 0s 190us/sample - loss: 0.5616 - accuracy: 0.6392 - va
```

1.9 Evaluación

- El modelo se evalúa en un conjunto de prueba (test) no observado durante el entrenamiento con el método ‘evaluate’. De esta forma se valida si el modelo aprendió a generalizar
- El método ‘evaluate’ regresa el ‘costo’ y ‘accuracy’ por lo que se crean dos variables para guardar los resultados

```
In [20]: test_loss, test_acc = model1.evaluate(x_test,  y_test, verbose=2)

         print('\nTest accuracy:', test_acc)
```

```
188/1 - 0s - loss: 0.3904 - accuracy: 0.9096

Test accuracy: 0.90957445
```

```
In [21]: test_loss, test_acc = model2.evaluate(x_test,  y_test, verbose=2)

         print('\nTest accuracy:', test_acc)
```

```
188/1 - 0s - loss: 0.3889 - accuracy: 0.8830

Test accuracy: 0.88297874
```

```
In [22]: test_loss, test_acc = model3.evaluate(x_test,  y_test, verbose=2)

         print('\nTest accuracy:', test_acc)
```

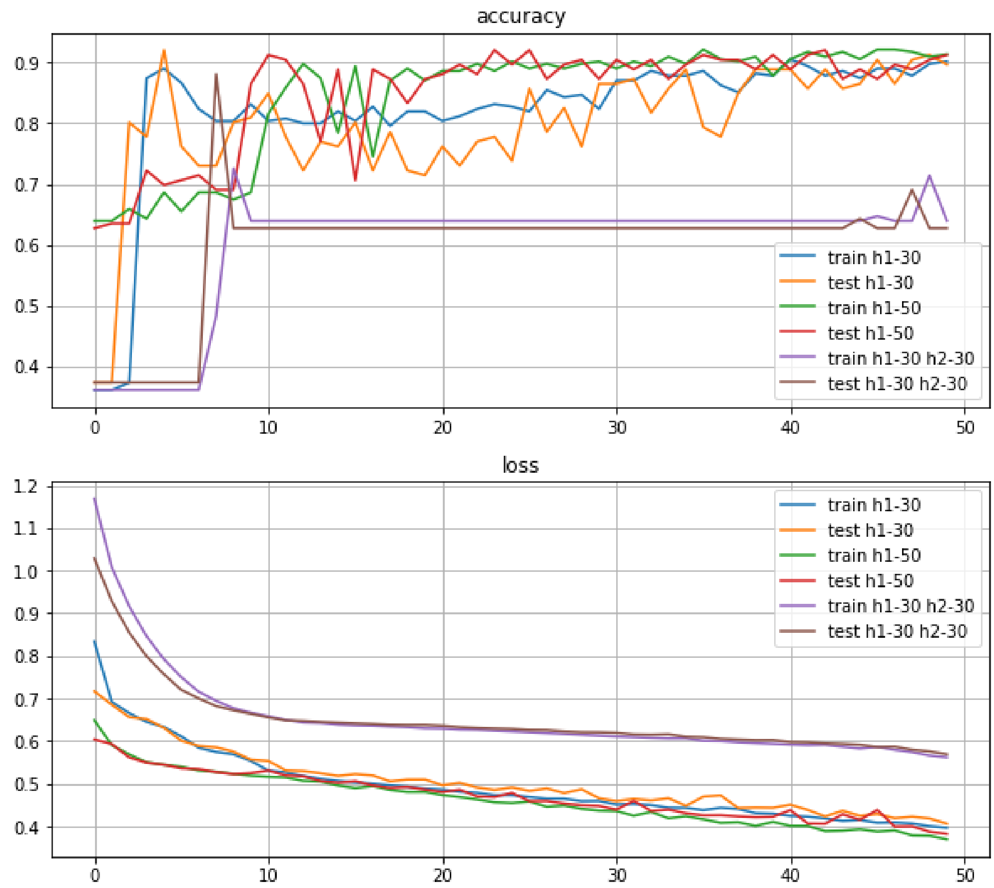
```
188/1 - 0s - loss: 0.5662 - accuracy: 0.6117

Test accuracy: 0.61170214
```

```
In [26]: plt.figure(figsize=(10,9))

         plt.subplot(211)
```

```python
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.title('accuracy')
plt.legend(['train h1-30', 'test h1-30',
            'train h1-50', 'test h1-50',
            'train h1-30 h2-30', 'test h1-30 h2-30'])
plt.grid()
plt.subplot(212)
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.plot(history3.history['loss'])
plt.plot(history3.history['val_loss'])
plt.title('loss')
plt.legend(['train h1-30', 'test h1-30',
            'train h1-50', 'test h1-50',
            'train h1-30 h2-30', 'test h1-30 h2-30'])
plt.grid()
```



1.10 Visualizar la clasificación

- Visualizar la clasificación del modelo con mayor 'accuracy' en el conjunto de prueba

In [29]:
```
from sklearn.decomposition import PCA
import numpy as np
pca = PCA(n_components=2).fit(x_test)
y = y_test
X_pca = pca.transform(x_test)
```

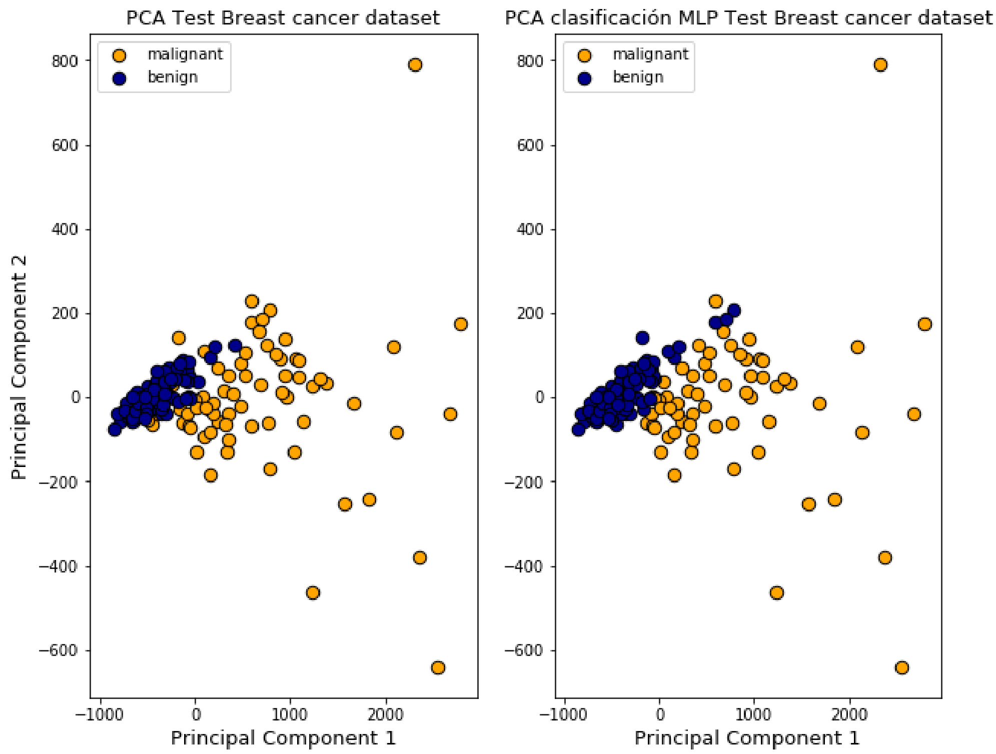
In [30]:
```
target_ids = np.unique(y)
plt.figure(figsize=(10,8))
colors = ['orange', 'darkblue']
target_names = ['malignant','benign']
plt.subplot(121)
for i, c, label in zip(target_ids, colors, target_names):
    plt.scatter(X_pca[i == y,0], X_pca[i == y,1], c = c,  edgecolors='black', s=285,labe
plt.legend()
```

```
plt.title('PCA Test Breast cancer dataset',fontsize=13)
plt.xlabel("Principal Component 1",fontsize=13)
plt.ylabel("Principal Component 2",fontsize=13)


plt.subplot(122)
y_prob = model1.predict(x_test)
y  = (y_prob > 0.5).astype(np.int)
y = y[:,0] # solo una dimensión


for i, c, label in zip(target_ids, colors, target_names):
    plt.scatter(X_pca[i == y,0], X_pca[i == y,1], c = c,  edgecolors='black', s=285,labe
plt.legend()
plt.title('PCA clasificación MLP Test Breast cancer dataset',fontsize=13)
plt.xlabel("Principal Component 1",fontsize=13)


plt.show()
```



- Experimentar con diferentes capas y número de neuronas

- Entrenar en menor tiempo y obtener mejor ‘accuracy’ modificando solo el número de capas y neuronas.
- Aplicar MLP a otro dataset