

Memoria final de Prácticas - Machine Learning y Azure

Ignacio Alcalde Torrecusa
Laboratorio L1

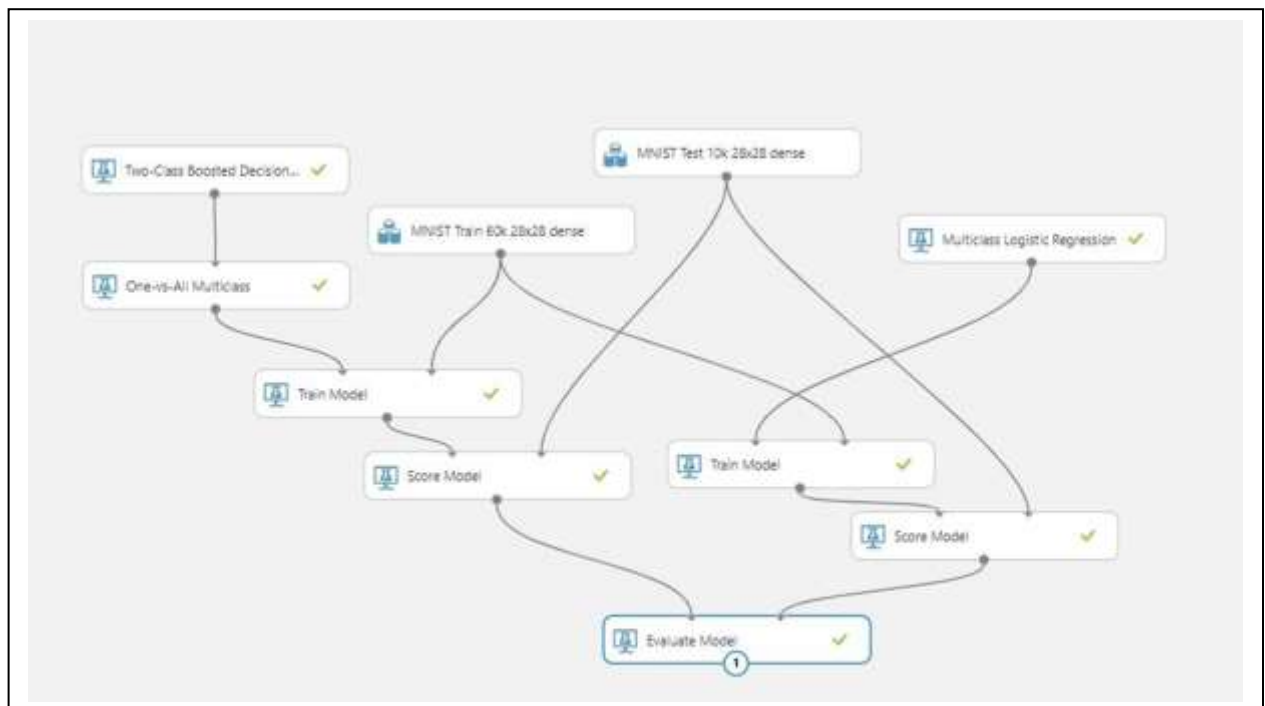
Sesión 01: introducción a Machine Learning y Azure

Pregunta planteada en la sesión: ¿Para qué sirve el split data y porque se le asigna al entrenamiento la mayor parte de él?

Nuestro algoritmo se divide en conjunto de entrenamiento y conjunto de test, esta división es llevada a cabo en el split data. Este componente es útil cuando necesita separar los datos en conjuntos de entrenamiento y prueba. También puede personalizar la forma en que se dividen los datos.

Sesión 02: árboles de decisión

ÁRBOL DE DECISIÓN:



RESULTADOS OBTENIDOS:

ÁRBOL 10:

Metrics	
Overall accuracy	0.9265
Average accuracy	0.9853
Micro-averaged precision	0.9265
Macro-averaged precision	0.926612
Micro-averaged recall	0.9265
Macro-averaged recall	0.925627

ÁRBOL 100:

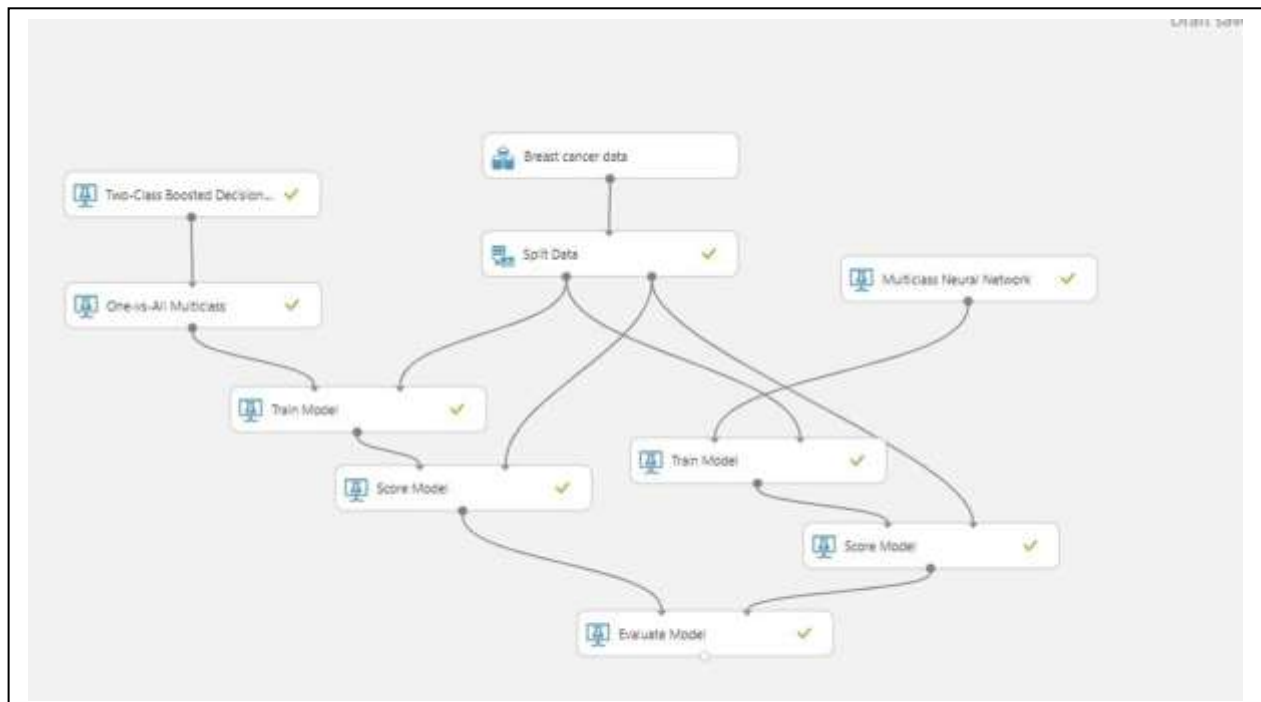
Metrics	
Overall accuracy	0.9674
Average accuracy	0.99348
Micro-averaged precision	0.9674
Macro-averaged precision	0.967335
Micro-averaged recall	0.9674
Macro-averaged recall	0.967114

Pregunta planteada en la sesión: Cambiar hiperparámetro de boosted y cambiar número de árboles generados=10. ¿Ha empeorado la ejecución con respecto a la ejecución con 100 árboles?

A mayor número de pruebas, realizará más iteraciones y obtendrás una mayor precisión.

Sesión 03: redes neuronales

ESQUEMA NEURONAL:



RESULTADOS OBTENIDOS:

ÁRBOL 100 “Hidden nodes”:

Metrics	
Overall accuracy	0.9265
Average accuracy	0.9853
Micro-averaged precision	0.9265
Macro-averaged precision	0.926612
Micro-averaged recall	0.9265
Macro-averaged recall	0.925627

Pregunta planteada en la sesión: Cambiar hiperparámetro, ¿que cambia usando una multiclass de redes neurales?

Al tener menos nodos funcionales el programa pierde precisión, tal y como puede comprobarse en la foto adjunta:

■ Multiclass Neural Network

Create trainer mode:
Single Parameter

Hidden layer specification:
Fully-connected case

Number of hidden nodes
10

The learning rate
0.1

Number of learning iter...
100

The initial learning wel...
0.1

The momentum
0

ÁRBOL 10 “Hidden nodes”:

Metrics	
Overall accuracy	0.9674
Average accuracy	0.99348
Micro-averaged precision	0.9674
Macro-averaged precision	0.967335
Micro-averaged recall	0.9674
Macro-averaged recall	0.967114

Sesión 04: redes convolucionales

Pregunta planteada en la sesión

TAREA 1:

- Parámetro: Los datos de entrenamiento X_train (conjunto de imágenes de entrenamiento en MNIST).
- Parámetro: Las etiquetas de los datos de entrenamiento y_train (conjunto de etiquetas correspondientes a las imágenes de entrenamiento MNIST).

```
• TAREA 1-> Completa los dos primeros parámetros. El primero deben ser los datos de entrenamiento, y el segundo, las etiquetas de dichos datos.
```

```
[43] model.fit(X_train, y_train, batch_size = 128, epochs = 5)
```

```
Epoch 1/5
469/469 [=====] - 6s 12ms/step - loss: 0.2673 - categorical_accuracy: 0.9231
Epoch 2/5
469/469 [=====] - 5s 10ms/step - loss: 0.1068 - categorical_accuracy: 0.9686
Epoch 3/5
469/469 [=====] - 4s 10ms/step - loss: 0.0705 - categorical_accuracy: 0.9794
Epoch 4/5
469/469 [=====] - 6s 12ms/step - loss: 0.0507 - categorical_accuracy: 0.9850
Epoch 5/5
469/469 [=====] - 5s 10ms/step - loss: 0.0381 - categorical_accuracy: 0.9888
<keras.callbacks.History at 0x7f117f7b74c0>
```

TAREA 2:

- Parámetro: Los datos de pruebas X_test correspondientes al conjunto de imágenes de prueba MNIST.
- Parámetro: Las etiquetas de los datos de prueba y_test correspondientes al conjunto de etiquetas correspondientes a las imágenes de prueba MNIST.

```
• TAREA 2-> Completa los dos únicos parámetros de la función evaluate. Esta función se encarga de hacer el test del modelo MLP
```

```
[44] test_loss, test_acc = model.evaluate(X_test, y_test)
```

```
print("Test loss: ", test_loss)
print("Test accuracy: ", test_acc)
```

```
313/313 [=====] - 2s 4ms/step - loss: 0.0690 - categorical_accuracy: 0.9791
Test loss:      0.0689881443977356
Test accuracy:  0.9790999889373779
```

TAREA 3:

- Parámetro: `X_test[8]`, seleccionamos la muestra de prueba 8 de los datos de prueba `X_test`.
- Parámetro: `y_test[8]`, selecciona la etiqueta de la muestra de prueba 8 de los datos de prueba `y_test`.

• Tarea 3-> Muestra el valor predicho y el real de una muestra de test, la número 8.

```
[46] y_pred = model.predict(X_test)

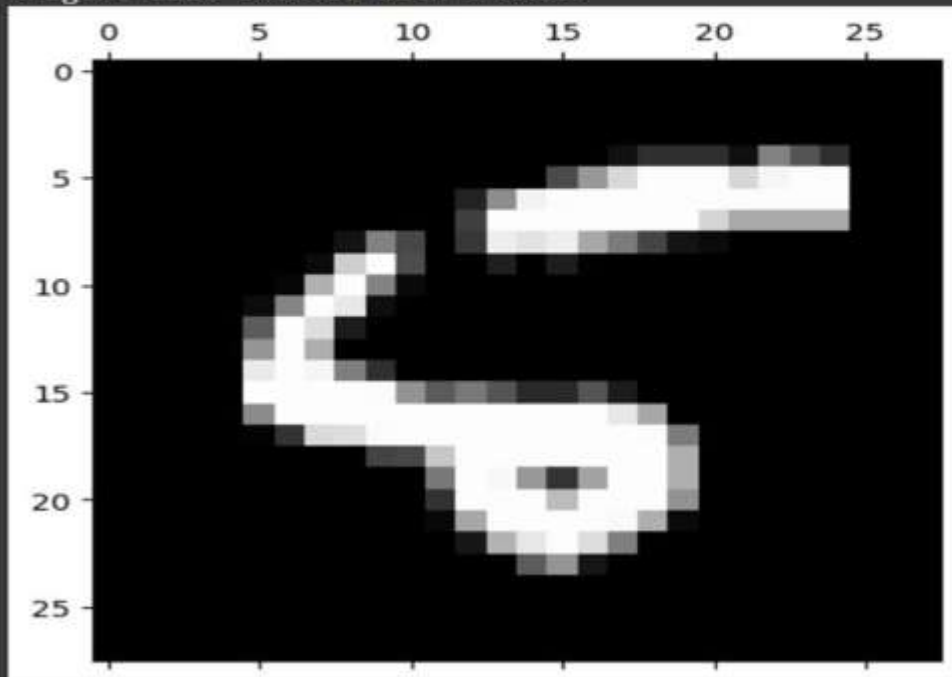
# Take a look to the predictions in the example 8
print(y_pred[8])
print("Predicted: ", np.argmax(y_pred[8]))
print("Actual:    ", np.argmax(y_test[8]))

# Plot
plt.gray()
plt.matshow(X_test[8, :].reshape(28,28))
plt.show()

313/313 [=====] - 1s 4ms/step
[7.8840077e-07 9.5234967e-09 1.4588019e-03 7.0268261e-06 4.5526060e-04
 9.1170561e-01 7.5749710e-02 6.5703667e-08 1.0214066e-02 4.0869103e-04]
Predicted: 5
Actual:    5
<Figure size 640x480 with 0 Axes>
```

Resultado:

```
313/313 [=====] - 1s 4ms/step
[7.8840077e-07 9.5234967e-09 1.4588019e-03 7.0268261e-06 4.5526060e-04
 9.1170561e-01 7.5749710e-02 6.5703667e-08 1.0214066e-02 4.0869103e-04]
Predicted: 5
Actual:    5
<Figure size 640x480 with 0 Axes>
```



Sesión 05: Python y Machine Learning

Pregunta planteada en la sesión: ¿Cuál es la diferencia que hay entre usar un kernel 3x3 y un kernel 7x7?

Un kernel es una matriz que se utiliza en el procesamiento de imágenes para aplicar operaciones como convolución, desenfoque o detección de bordes. La diferencia entre un kernel 3x3 y un kernel 7x7 es el tamaño de la matriz.

Un kernel 3x3 es más pequeño y se utiliza para operaciones más simples como el desenfoque o la detección de bordes básica. Por otro lado, un kernel 7x7 es más grande y se utiliza para operaciones más complejas como la detección de patrones, texturas o características específicas en una imagen.

En resumen, la elección del tamaño del kernel dependerá de la complejidad de la operación que se quiera realizar y del nivel de detalle que se quiera obtener en la imagen procesada.

```
# 1.5.2 Ejecución del modelo con el conjunto de pruebas. Para ello, utilizaremos la popular instrucción evaluate, obteniendo así la exactitud d
# Como siempre, se proporciona el conjunto de test, esta vez representado mediante un data_generator que carga en memoria el banco de imá
# Del mismo modo, steps refleja el número de batches que se procesarán en la predicción del conjunto de pruebas. Si eran 1000 imágenes y
# ¡Nada mal para haberlo hecho en un puñado de instrucciones!
#
test_loss, test_acc = model.evaluate(test_generator, steps=50)
print('Test accuracy:', test_acc)
```

50/50 [=====] - 17s 338ms/step - loss: 0.6383 - binary_accuracy: 0.7250
Test accuracy: 0.7250000238418579

