fastai_2

October 16, 2022

```
clases:  [2 3 4 1]
```

1 Experimentos de entrenamiento:

Juntando los 4 subsets el modelo entrena regular, así que vamos a probar lo siguiente: 1. entrenamiento de los 4 subsets juntos

```
Learning Rate:  0.0005
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Better model found at epoch 0 with balanced_accuracy_score value:
0.4445641491735003.
No improvement since epoch 0: early stopping
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
Better model found at epoch 0 with accuracy value: 0.42539963126182556.
Better model found at epoch 6 with accuracy value: 0.44049733877182007.
No improvement since epoch 6: early stopping
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

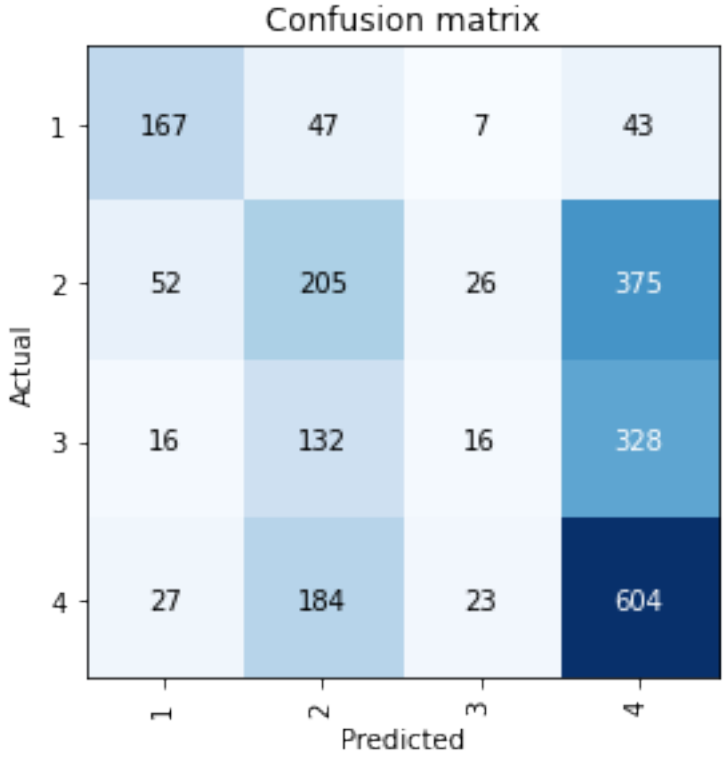
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

<IPython.core.display.HTML object>

	precision	recall	f1-score	support
1	0.64	0.63	0.63	264
2	0.36	0.31	0.33	658
3	0.22	0.03	0.06	492
4	0.45	0.72	0.55	838
accuracy			0.44	2252
macro avg	0.42	0.42	0.39	2252
weighted avg	0.40	0.44	0.39	2252



2. Entrenamiento de los 4 datasets juntos, obligando a entrenar con más épocas (datasets más complejos suelen necesitar entrenamientos más largos)

Learning Rate: 0.0005

<IPython.core.display.HTML object>

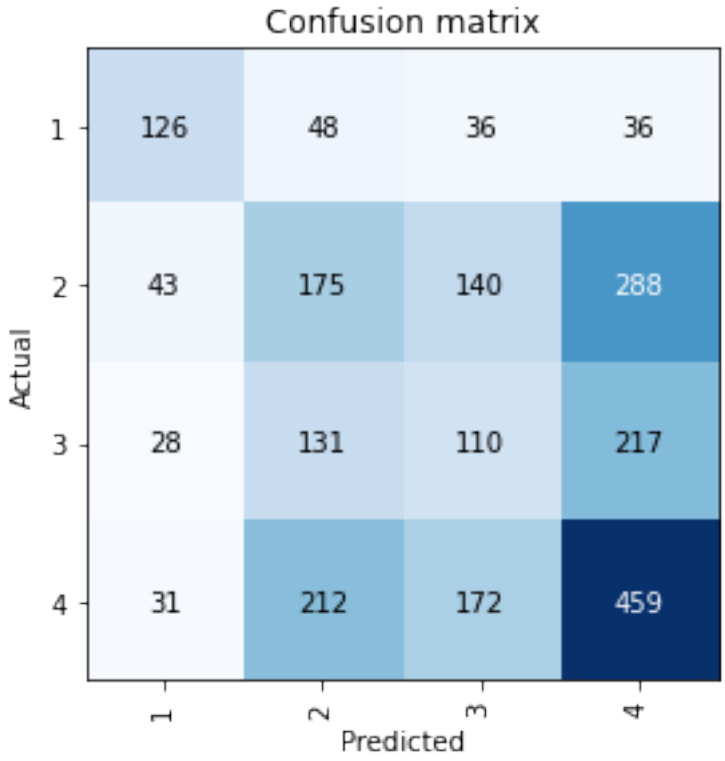
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Better model found at epoch 0 with balanced_accuracy_score value:
0.43026879829578624.
No improvement since epoch 1: early stopping

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Better model found at epoch 0 with train_loss value: 1.191886305809021.
Better model found at epoch 1 with train_loss value: 1.1812348365783691.
Better model found at epoch 3 with train_loss value: 1.1687073707580566.
Better model found at epoch 7 with train_loss value: 1.1554505825042725.
Better model found at epoch 10 with train_loss value: 1.1420049667358398.
Better model found at epoch 12 with train_loss value: 1.1313071250915527.
Better model found at epoch 14 with train_loss value: 1.119753122329712.
Better model found at epoch 16 with train_loss value: 1.108251690864563.
Better model found at epoch 18 with train_loss value: 1.0969239473342896.
Better model found at epoch 20 with train_loss value: 1.0861973762512207.
Better model found at epoch 23 with train_loss value: 1.074162483215332.
Better model found at epoch 27 with train_loss value: 1.0632930994033813.
Better model found at epoch 31 with train_loss value: 1.0495795011520386.
Better model found at epoch 34 with train_loss value: 1.0385147333145142.
Better model found at epoch 37 with train_loss value: 1.0281896591186523.
Better model found at epoch 41 with train_loss value: 1.0159963369369507.
Better model found at epoch 44 with train_loss value: 1.00502347946167.
Better model found at epoch 47 with train_loss value: 0.9946674704551697.
Better model found at epoch 50 with train_loss value: 0.982183575630188.
Better model found at epoch 52 with train_loss value: 0.9708428978919983.
Better model found at epoch 55 with train_loss value: 0.9573861956596375.
Better model found at epoch 58 with train_loss value: 0.9442201852798462.
Better model found at epoch 61 with train_loss value: 0.9332265853881836.
Better model found at epoch 63 with train_loss value: 0.9229888319969177.
Better model found at epoch 65 with train_loss value: 0.9095810651779175.
Better model found at epoch 67 with train_loss value: 0.8962709903717041.
Better model found at epoch 69 with train_loss value: 0.8855646252632141.
Better model found at epoch 72 with train_loss value: 0.8730986714363098.
Better model found at epoch 75 with train_loss value: 0.8586689233779907.
Better model found at epoch 77 with train_loss value: 0.8455844521522522.
Better model found at epoch 79 with train_loss value: 0.8315596580505371.
Better model found at epoch 81 with train_loss value: 0.8178314566612244.
Better model found at epoch 83 with train_loss value: 0.8055124878883362.
Better model found at epoch 86 with train_loss value: 0.7902351021766663.
```

```
Better model found at epoch 88 with train_loss value: 0.778197169303894.
Better model found at epoch 91 with train_loss value: 0.7627842426300049.
Better model found at epoch 93 with train_loss value: 0.7505537867546082.
Better model found at epoch 96 with train_loss value: 0.7357161641120911.
Better model found at epoch 98 with train_loss value: 0.7252020239830017.
Better model found at epoch 100 with train_loss value: 0.7110821008682251.
Better model found at epoch 103 with train_loss value: 0.6952039003372192.
Better model found at epoch 105 with train_loss value: 0.6828164458274841.
Better model found at epoch 107 with train_loss value: 0.6707348227500916.
Better model found at epoch 109 with train_loss value: 0.6563963890075684.
Better model found at epoch 111 with train_loss value: 0.6401558518409729.
Better model found at epoch 113 with train_loss value: 0.6233146786689758.
Better model found at epoch 115 with train_loss value: 0.6084535717964172.
Better model found at epoch 117 with train_loss value: 0.5977118611335754.
Better model found at epoch 119 with train_loss value: 0.585112452507019.
Better model found at epoch 122 with train_loss value: 0.5748468041419983.
Better model found at epoch 124 with train_loss value: 0.5606157779693604.
Better model found at epoch 126 with train_loss value: 0.5505250096321106.
Better model found at epoch 128 with train_loss value: 0.5371754765510559.
Better model found at epoch 130 with train_loss value: 0.5243620872497559.
Better model found at epoch 132 with train_loss value: 0.5130223631858826.
Better model found at epoch 135 with train_loss value: 0.49860477447509766.
Better model found at epoch 137 with train_loss value: 0.48729830980300903.
Better model found at epoch 140 with train_loss value: 0.473127543926239.
Better model found at epoch 142 with train_loss value: 0.45953652262687683.
Better model found at epoch 145 with train_loss value: 0.44446849822998047.
Better model found at epoch 147 with train_loss value: 0.434299498796463.
Better model found at epoch 150 with train_loss value: 0.42108574509620667.
Better model found at epoch 153 with train_loss value: 0.40938952565193176.
Better model found at epoch 158 with train_loss value: 0.3950420618057251.
Better model found at epoch 162 with train_loss value: 0.3834528625011444.
Better model found at epoch 165 with train_loss value: 0.36979398131370544.
Better model found at epoch 170 with train_loss value: 0.3580639660358429.
Better model found at epoch 173 with train_loss value: 0.3468324840068817.
Better model found at epoch 177 with train_loss value: 0.33677005767822266.
Better model found at epoch 185 with train_loss value: 0.32318034768104553.
Better model found at epoch 196 with train_loss value: 0.3125917315483093.

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

              precision    recall  f1-score   support
```

1	0.55	0.51	0.53	246
2	0.31	0.27	0.29	646
3	0.24	0.23	0.23	486
4	0.46	0.53	0.49	874
accuracy			0.39	2252
macro avg	0.39	0.38	0.39	2252
weighted avg	0.38	0.39	0.38	2252

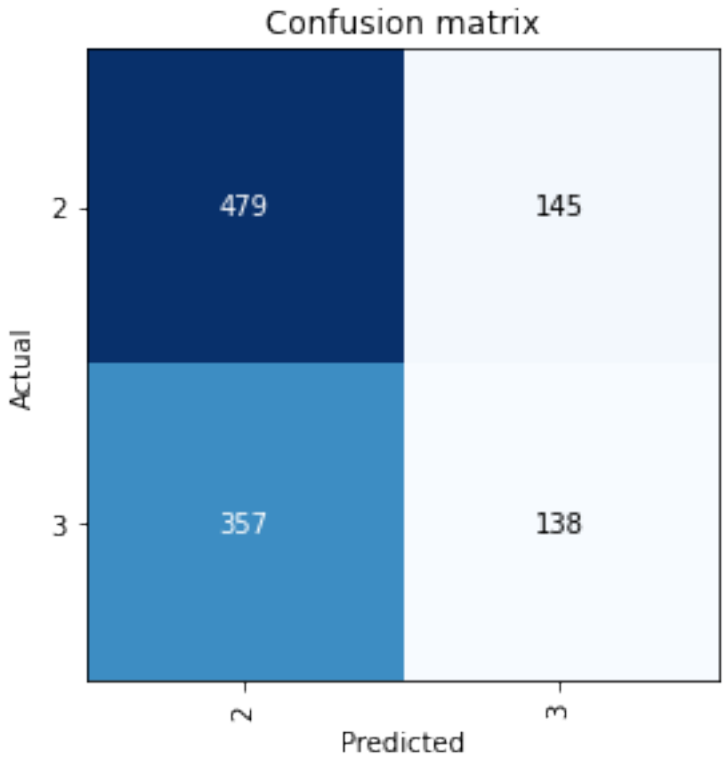


3. entrenamiento clasificación binaria –> clases 2 y 3 (las que daban peor)

```
Learning Rate:  0.0005
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Better model found at epoch 0 with balanced_accuracy_score value:
0.49047202797202794.
Better model found at epoch 1 with balanced_accuracy_score value:
0.521493783993784.

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

Better model found at epoch 0 with accuracy value: 0.5406612753868103.
Better model found at epoch 3 with accuracy value: 0.5513851642608643.
No improvement since epoch 3: early stopping

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

	precision	recall	f1-score	support
2	0.57	0.77	0.66	624
3	0.49	0.28	0.35	495
accuracy			0.55	1119
macro avg	0.53	0.52	0.51	1119
weighted avg	0.54	0.55	0.52	1119



4. Entrenamiento de clasificación binaria de los subsets 1 y 4 (Los que daban mejor)

```
Learning Rate:  0.0005
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
Better model found at epoch 0 with balanced_accuracy_score value:
0.8681745998481821.
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

```
Better model found at epoch 0 with accuracy value: 0.8719081282615662.
Better model found at epoch 2 with accuracy value: 0.8842756152153015.
Better model found at epoch 4 with accuracy value: 0.8948763012886047.
Better model found at epoch 8 with accuracy value: 0.9054770469665527.
Better model found at epoch 18 with accuracy value: 0.916961133480072.
No improvement since epoch 18: early stopping
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

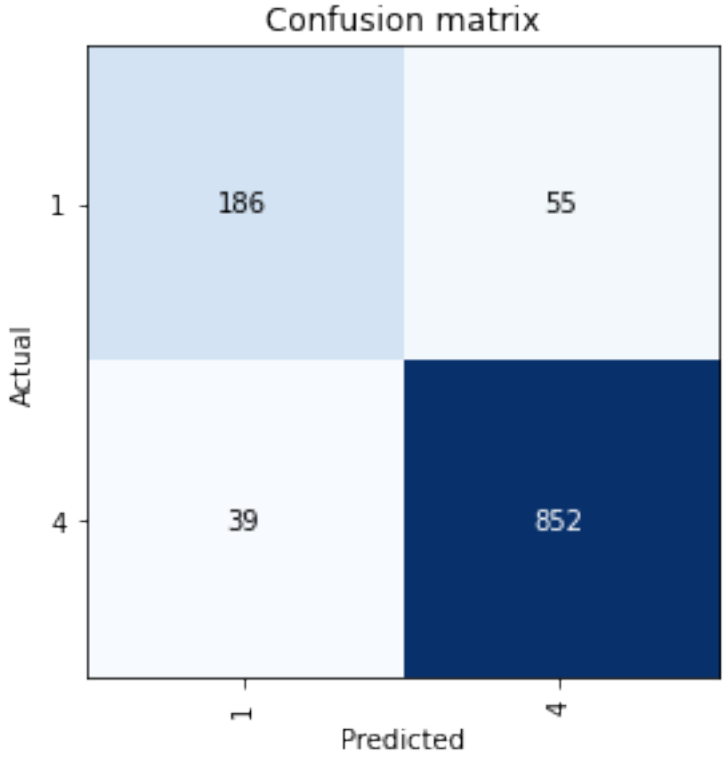
```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

	precision	recall	f1-score	support
1	0.83	0.77	0.80	241
4	0.94	0.96	0.95	891
accuracy			0.92	1132
macro avg	0.88	0.86	0.87	1132
weighted avg	0.92	0.92	0.92	1132

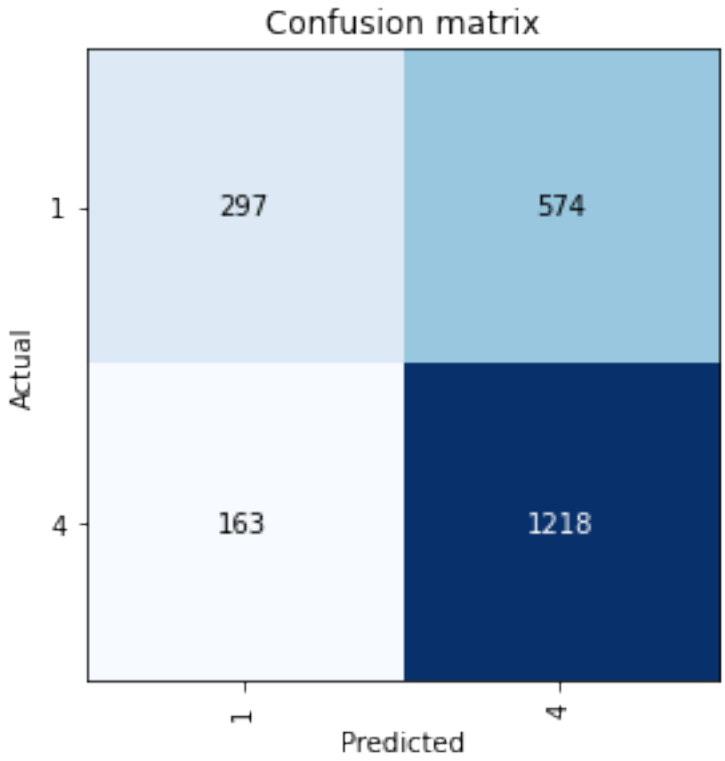


5. Entrenamiento de clasificación binaria juntando clases Efectivas (1 y 2) y las no efectivas (3 y 4)

```
Learning Rate:  0.0005
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
Better model found at epoch 0 with balanced_accuracy_score value:
0.5899479652924593.
Better model found at epoch 1 with balanced_accuracy_score value:
0.6102779978567586.
No improvement since epoch 1: early stopping
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
Better model found at epoch 0 with accuracy value: 0.672735333442688.
No improvement since epoch 1: early stopping
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>
```

	precision	recall	f1-score	support
1	0.65	0.34	0.45	871
4	0.68	0.88	0.77	1381
accuracy			0.67	2252
macro avg	0.66	0.61	0.61	2252

```
weighted avg       0.67      0.67      0.64      2252
```



1.1 Conclusiones:

1. Después de entrenar diferentes modelos para deuce y ad, vemos que entrenando los 4 modelos el resultado no es muy bueno. Sin embargo entrenando de 2 en dos sale mucho mejor.

   1.1. A parte veo que los datos no representan todas las opciones posibles, puesto que no están los que tenían efectividad 0 en el dataset BKG_Corregido. Únicamente se ha tenido en cuenta cuando gana el sacador. Faltarían datos de cuando gana el restador.

   1.2. Pensar mejor cómo proceder y si es necesario tener todos los datos para el entrenamiento.

2. Ver qué opciones da FastAI para interpretar el modelo xAI