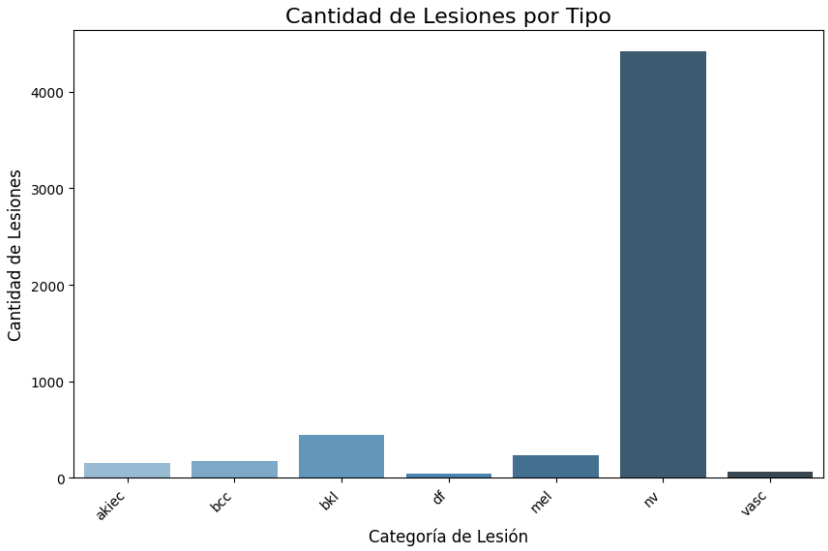
Pruebas 27-9

La baja performance del modelo de las pruebas anteriores pueden ser debido a que los datos están desbalanceados.

Encontré otro dataset que puede utilizarse para incorporar imágenes de alguna de las clases más bajas. Asimismo, elimine del dataset original las filas que tenían algún nulo en la variable age, y el valor unknown en las variables sex o localization.

Eso deja el dataset todavía con una distribución desbalanceada

akiec	151
1	bcc	175
2	bkl	440
3	df	39
4	mel	230
5	nv	4415
6	vasc	64



Estos metadatos filtrados están en el archivo: HAM10000_metadata_cleaned.csv

Del otro dataset https://www.kaggle.com/datasets/mahdavi1202/skin-cancer/data

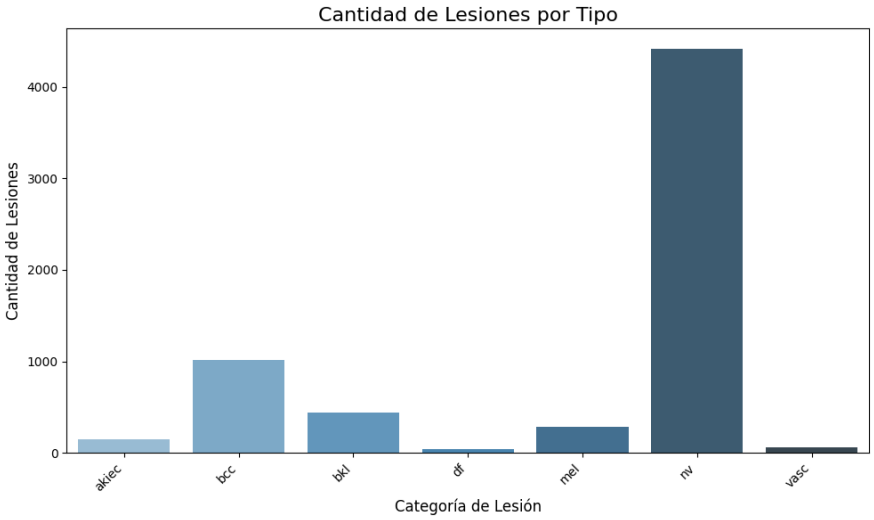
tomé el archivo de metadatos (raw.githubusercontent.com/mvegetti/IAcourse/refs/heads/main/metadataPAD-UFES-20.csv) y me quedé sólo con las columnas que coinciden con el dataset original (le tuve que cambiar los nombres) y con las imágenes cuyas etiquetas son bcc, mel y nev.

De los directorios con las imágenes copie al directorio de imágenes originales las que corresponden a los metadatos que quedaron en los archivos.

Finalmente uní los metadados del segundo dataset con el archivo de metadatos originales filtrado y quedó el archivo
raw.githubusercontent.com/mvegetti/IAcourse/refs/heads/main/dataset_unido3.csv

Aún quedan desbalanceadas las clases

dx	cantidad_de_lesiones
0 akiec	151
1 bcc	1020
2 bkl	440
3 df	39
4 mel	282
5 nv	4415
6 vasc	64



Luego generé un dataset unificado de imágenes y etiquetas (raw.githubusercontent.com/mvegetti/IAcourse/refs/heads/main/dataset_unido3.csv)

Al que utilicé para entrenar un modelo (ModeladoyEntrenamientoV4.py)

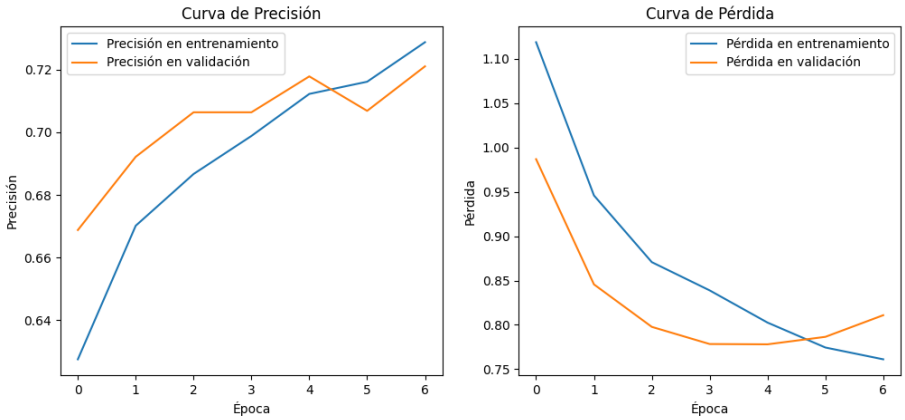
```
model_cnn = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(112, 112, 3), kernel_regularizer=l2(0.01)),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),  # Dropout después de la primera capa de MaxPooling
    Conv2D(64, (3, 3), activation='relu',
kernel_regularizer=l2(0.01)),
```

```
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),  # Dropout después de la segunda capa de MaxPooling
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),   # Dropout ya existente antes de la capa de salida
    Dense(7, activation='softmax')
])
```

Obteniendo una precisión 82%

	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.69	0.85	0.76	204
2	0.00	0.00	0.00	88
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	56
5	0.85	0.99	0.91	884
6	0.00	0.00	0.00	13
accuracy			0.82	1283
macro avg	0.22	0.26	0.24	1283
weighted avg	0.69	0.82	0.75	1283

```
Matriz de Confusión:
[[  0  20   0   0   0  10   0]
 [  0 174   0   0   0  30   0]
 [  0  30   0   0   0  58   0]
 [  0   0   0   0   0   8   0]
 [  0  17   0   0   0  39   0]
 [  0  12   0   0   0 872   0]
 [  0   1   0   0   0  12   0]]
```



Los resultados de las métricas de evaluación muestran un rendimiento desigual del modelo en las diferentes clases. Algunas clases tienen una precisión, recall y f1-score muy bajas, incluso iguales a cero. Esto indica que el modelo no está siendo capaz de predecir correctamente esas clases.

La clase 5 (nev) es la que mejor resultado tiene xq es la que más casos tiene.

Intento modificando el peso de las clases para dar mayor peso a las clases prioritarias ->

No mejoró

	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.63	0.93	0.75	204
2	0.00	0.00	0.00	88
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	56
5	0.87	0.97	0.92	884
6	0.00	0.00	0.00	13
accuracy			0.82	1283
macro avg	0.21	0.27	0.24	1283
weighted avg	0.70	0.82	0.75	1283

Matriz de Confusión:

```
[[  0  24   0   0   0   6   0]
 [  0 190   0   0   0  14   0]
 [  0  36   0   0   0  52   0]
 [  0   3   0   0   0   5   0]
 [  0  18   0   0   0  38   0]
 [  0  28   0   0   0 856   0]
 [  0   3   0   0   0  10   0]]
```

Sigue habiendo clases que performan mal.

Voy a ver como se puede implementar data augmentation para generar más figuras para las clases minoritarias