```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout
from tensorflow.keras.callbacks import EarlyStopping
import json
import os
import numpy as np
from sklearn.model_selection import train_test_split
from PIL import UnidentifiedImageError  # Importar para capturar errores
de imagen no reconocida

# Ruta de la carpeta de imágenes de entrenamiento
ruta_imagenes = r"C:\Users\ismae\Desktop\TFG\Python Windows\frames"
# Ruta del archivo JSON de metadatos
ruta_metadatos = r"C:\Users\ismae\Desktop\DATA-Tendon-
Long\file_info.json"

# Cargar los metadatos
with open(ruta_metadatos, 'r') as f:
    metadatos = json.load(f)

# Filtrar y etiquetar las imágenes
imagenes = []
etiquetas = []

for item in metadatos:
    calidad = item['Echography_quality']
    # Obtener solo el nombre de archivo de la ruta en "Original_image"
    nombre_imagen = os.path.basename(item['Original_image'])  # Extrae el
nombre de archivo
    ruta_imagen = os.path.join(ruta_imagenes, nombre_imagen)

    # Solo procesar archivos PNG o JPG
    if not ruta_imagen.lower().endswith(('.png', '.jpg', '.jpeg')):
        print(f"Archivo omitido (no es PNG o JPG): {ruta_imagen}")
        continue

    # Etiquetar: 1 para 'buenas' (good, fair) y 0 para 'malas' (null)
    etiqueta = 1 if calidad in ["good", "fair"] else 0
    try:
        imagen = load_img(ruta_imagen, target_size=(128, 128))  # Cargar
y redimensionar
        imagen_array = img_to_array(imagen) / 255.0  # Normalizar
        imagenes.append(imagen_array)
        etiquetas.append(etiqueta)
    except (FileNotFoundError, UnidentifiedImageError, OSError):
        print(f"Archivo no válido o no encontrado: {ruta_imagen}")

# Verificar si se cargaron imágenes
if not imagenes:
    print("Error: No se cargaron imágenes válidas. Revisa las rutas y el
archivo de metadatos.")
else:
    # Convertir listas a arrays numpy
```

```
    imagenes = np.array(imagenes)
    etiquetas = np.array(etiquetas)

    # Dividir en conjunto de entrenamiento y validación
    x_train, x_val, y_train, y_val = train_test_split(imagenes,
etiquetas, test_size=0.2, random_state=42)

    # Definir el modelo de clasificación de imágenes
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])

    # Compilar el modelo
    model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

    # Entrenar el modelo con EarlyStopping
    early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
    history = model.fit(
        x_train, y_train,
        epochs=20,
        validation_data=(x_val, y_val),
        callbacks=[early_stopping],
        batch_size=32
    )

    # Guardar el modelo
    model.save('modelo_ecografias.h5')
    print("Modelo guardado como 'modelo_ecografias.h5'")
```