Estos errores viene del endpointment – Train model

**1. Error de validación: "Unprocessable Content"**

**Descripción:**  
El endpoint recibía una lista de objetos, pero el parámetro no estaba correctamente tipado.  
**Solución:**  
El parámetro debe definirse como una lista de instancias del modelo new\_animal:

def post(data: List[new\_animal]):

### 2. Error de tipo: "Input should be a valid string" en el campo label

**Descripción:**  
El modelo esperaba un string en el campo label, pero se enviaba un número entero (por ejemplo, 1, 2).  
**Solución:**  
Ajustar el tipo de dato en el modelo Pydantic para aceptar int en lugar de str:

label: int

### 3. Error de concatenación de datos: "cannot concatenate object of type '<class 'str'>"

**Descripción:**  
El CSV descargado de MinIO se trataba como cadena (str), y no como un DataFrame, al intentar concatenarlo con nuevos datos.  
**Solución:**  
Convertir el contenido a DataFrame usando StringIO:

from io import StringIO

csv\_old\_df = pd.read\_csv(StringIO(csv\_old))

### 4. Error al guardar archivo: "Cannot save file into a non-existent directory"

**Descripción:**  
El script intentaba guardar un archivo en una carpeta que no existía en el sistema de archivos.  
**Solución:**  
Crear la carpeta previamente con:

import os

os.makedirs("data", exist\_ok=True)

### 5. Error del sistema operativo: "[WinError 3] El sistema no puede encontrar la ruta especificada"

**Descripción:**  
Se utilizaban rutas relativas que dependían de la ubicación desde donde se ejecutaba el script (app.py).  
**Solución:**  
Usar rutas absolutas relativas al archivo actual:

import os

BASE\_DIR = os.path.dirname(**\_\_file\_\_**)

path = os.path.join(BASE\_DIR, "data", "animalsLabeled.csv")

### 6. Error en función upload\_csv\_minio: No se encuentra el archivo a subir

**Descripción:**  
La función intentaba subir un archivo que no existía en la ruta esperada.  
**Solución:**  
Asegurarse de que la ruta source\_file sea válida en tiempo de ejecución y esté alineada con la estructura del proyecto:

source\_file = os.path.join(BASE\_DIR, "data", "animalsLabeled.csv")

### 7. Potencial error en upload\_model\_minio

**Descripción:**  
Mismo riesgo que el caso anterior: se asume una ruta relativa al directorio actual.  
**Solución:**  
Aplica la misma estrategia con os.path.join para construir rutas robustas:

source\_file = os.path.join(BASE\_DIR, "models", "decision\_tree\_model.pkl")

Ahora te comento los errores que he tenido con el **endpointment predict**:

### 1. ****AccessDenied al acceder a MinIO****

**Error:**  
S3 operation failed; code: AccessDenied, message: Access denied

**Causa:**  
El bucket "model-bucket2" existía pero el usuario de MinIO no tenía permisos para acceder o escribir.

**Solución:**

* Asegurarse de que el bucket "model-bucket2" exista.
* Verificar las credenciales (MINIO\_ACCESS\_KEY, MINIO\_SECRET\_KEY) en el .env.
* Verificar la política de permisos del bucket en MinIO: debe permitir lectura/escritura para el usuario correspondiente.

### 2. predict() ****recibe 2 argumentos pero solo acepta 1****

**Error:**  
predict() takes 1 positional argument but 2 were given

**Causa:**  
El nombre de la función para predecir (predict) colisionaba con el nombre del endpoint de FastAPI.

**Solución:**  
Renombrar la función de predicción importada para evitar el conflicto:

from DataPreprocessing\_MachineLearning import predict as predict\_model

Y usarla así en tu endpoint:

predictions = predict\_model(model, data\_predict)

### 3. ****setting an array element with a sequence****

**Error:**  
setting an array element with a sequence

**Causa:**  
El DataFrame que se le pasó al modelo tenía una estructura incorrecta, probablemente un campo tenía listas o estructuras anidadas.

**Solución:**  
Asegurarse de que el JSON que se envía al endpoint /predict sea una lista de objetos planos y convertirlos correctamente a un DataFrame:

data\_dicts = [item.dict() for item in data]

df = pd.DataFrame(data\_dicts)

**4. Feature names should match...**

**Error:**  
The feature names should match those that were passed during fit. Feature names must be in the same order as they were in fit.

**Causa:**  
Los nombres de columnas del DataFrame usado para predecir no coincidían exactamente con los usados al entrenar el modelo.

**Solución:**

* Verificar que las columnas del DataFrame estén en el **mismo orden y con los mismos nombres** que durante el entrenamiento.
* O bien guardar los nombres de las columnas durante el entrenamiento y usarlos para reordenar al predecir:

expected\_columns = [...]  # columnas usadas al entrenar

df = df[expected\_columns]

### 5. ****FastAPI espera una lista, pero se retorna un diccionario****

**Error:**  
ResponseValidationError: Input should be a valid list

**Causa:**  
El endpoint definía response\_model=List[str], pero se retornaba un diccionario como {"status": 200, "predictions": [...]}.

**Solución:**  
**Opción A:** devolver directamente una lista si se mantiene response\_model=List[str]:

return predictions

**Opción B:** cambiar el response\_model para reflejar el formato de salida:

class PredictionResponse(BaseModel):

    status: int

    predictions: List[str]

@app.post("/predict", response\_model=PredictionResponse)

Errores que he tenido con pytest:

### Resumen de errores con pytest

1. **Error: No se encontraba el módulo DataPreprocessing\_MachineLearning**
   * **Mensaje:** ModuleNotFoundError: No module named 'DataPreprocessing\_MachineLearning'
   * **Causa:** El archivo my\_api/app.py importaba ese módulo, pero no estaba en el PYTHONPATH o la estructura del proyecto no lo encontraba.
   * **Solución:**
     + Configurar PYTHONPATH correctamente para que incluya la raíz del proyecto.
     + Usar importaciones relativas si el módulo está dentro del paquete my\_api.
     + Separar o aislar los tests que no dependan de ese módulo para que no fallen por esa causa.
2. **Error: ModuleNotFoundError: No module named 'my\_api' al ejecutar pytest sin PYTHONPATH**
   * **Causa:** Ejecutar pytest tests/test\_models.py sin establecer PYTHONPATH o sin estar en la raíz del proyecto provocaba que no encontrara el paquete my\_api.
   * **Solución:** Ejecutar con PYTHONPATH=. o asegurarse de estar en la raíz y que el intérprete pueda resolver el paquete.
3. **Error por importaciones en my\_api/app.py afectando todos los tests**
   * Aunque el test no necesitaba funcionalidad de app.py, la importación de app en los tests hacía que se cargaran importaciones problemáticas (como la de DataPreprocessing\_MachineLearning), causando errores en tests que solo necesitaban modelos o endpoints.
   * Solución:
     + Aislar módulos o usar mocks para esas dependencias.
     + Modularizar importaciones para que app.py no cargue todo al importarse.
4. **Advertencias sobre uso de Pydantic en app.py durante los tests**
   * Warning: PydanticDeprecatedSince20: The dict method is deprecated; use model\_dump instead.
   * Esto no detiene tests, pero indica que hay que actualizar el código para Pydantic v2.

### Lecciones clave:

* Siempre configurar PYTHONPATH para que pytest pueda encontrar todos los paquetes internos.
* Modularizar imports para evitar que un fallo en una parte detenga toda la suite.
* Usar mocks en tests para evitar dependencias pesadas o no disponibles en el entorno de tests.
* Actualizar librerías y funciones deprecated para evitar warnings que luego podrían ser errores.

¿merece la pena el test unitario que he hecho sobre la predicción de mi modelo? Para casos exagerado si no devuelve esas etiqueta… me debería dar un error no?

En el docker:  
  
Me daba problema con minio

Error al cargar el modelo: HTTPConnectionPool(host='localhost', port=9000): Max retries exceeded with url: /model-bucket2?location= (Caused by NewConnectionError(...): Failed to establish a new connection: [Errno 111] Connection refused)

## ¿Por qué pasa esto?

Dentro de un contenedor Docker, localhost se refiere a **sí mismo**, no a tu máquina ni a otro contenedor.

Entonces, tu contenedor backend intenta conectar con MinIO en localhost:9000 pero MinIO no está corriendo dentro del mismo contenedor, sino en otro contenedor, o incluso en tu host.

Solcuión   
  
pasar de .env

MINIO\_ACCESS\_KEY=local

MINIO\_SECRET\_KEY=admin123

MINIO\_ENDPOINT=minio:9000

MINIO\_ACCESS\_KEY=local

MINIO\_SECRET\_KEY=admin123

Y de :

    def minio\_client():

    load\_dotenv()

    return Minio(

        "localhost:9000",

        access\_key=getenv("MINIO\_ACCESS\_KEY"),

        secret\_key=getenv("MINIO\_SECRET\_KEY"),

        secure=False

    )

def minio\_client():

    load\_dotenv()

    endpoint = getenv("MINIO\_ENDPOINT", "minio:9000").replace("http://", "").replace("https://", "")

    return Minio(

        endpoint,

        access\_key=getenv("MINIO\_ACCESS\_KEY"),

        secret\_key=getenv("MINIO\_SECRET\_KEY"),

        secure=False

    )