Apuntes / Manual de Desarrollo

Notas:

- Primero limpiar columnas, luego limpiar rutas de imagen.
- Convertir categories a lista, sacar main_category, codificar.
- Probar primero LabelEncoder antes que OneHot.
- Eliminar outliers antes del split.
- Split después de todo lo anterior.
- Imputar y escalar después del split, usando estadísticas solo de train.
- Dataset de PyTorch: probar a mano que funciona antes de meter en DataLoader.
- Modelo: diseñar primero en papel qué inputs y outputs espera.

```
# Cargar CSV
df = pd.read_csv('ruta/dataset.csv')

# Quitar columnas que no me interesan: id, nombre, etc
df = df.drop(['id', 'name', 'main_image_path', 'tags', ...], axis=1)

# Limpiar rutas de imagen y quedarme solo con las válidas
df = df[os.path.exists(df['image_path'])]

# ¿Cómo está la columna categories? ¿Son listas o strings?
# Probar a pasarlas a lista
import ast
df['categories_clean'] = df['categories'].apply(lambda x: ast.literal_eval(x) if isinstance(x,
str) else [])

# Sacar la categoría principal (la primera)
df['main_category'] = df['categories_clean'].apply(lambda x: x[0] if len(x) > 0 else
'Unknown')

# Codificar la categoría principal como número
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['cat_code'] = le.fit_transform(df['main_category'])

# Probar one-hot encoding multiclase
from sklearn.preprocessing import MultiLabelBinarizer
```

```
mlb = MultiLabelBinarizer()
onehot = mlb.fit_transform(df['categories_clean'])
# (Esto da un array, habrá que convertirlo en columnas...)

# Quitar outliers: por ejemplo, xps, visitas, likes, etc
for col in ['xps', 'Visits', 'Likes']:
   # Usar IQR
   q1 = df[col].quantile(0.25)
   q3 = df[col].quantile(0.75)
   iqr = q3 - q1
   df = df[(df[col] >= q1 - 1.5*iqr) & (df[col] <= q3 + 1.5*iqr)]

# Split train/test (cuidado, hacerlo después de todo lo anterior)
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split(df, test_size=0.2, stratify=df['cat_code'])

# Imputar nulos por la media de train
for col in ['xps', 'Visits', ...]:
   mean = train_df[col].mean()
   train_df[col] = train_df[col].fillna(mean)
   test_df[col] = test_df[col].fillna(mean)

# Escalar solo con datos de train
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(train_df[['xps', 'Visits', ...]])
train_df[['xps', 'Visits', ...]] = scaler.transform(train_df[['xps', 'Visits', ...]])
test_df[['xps', 'Visits', ...]] = scaler.transform(test_df[['xps', 'Visits', ...]])

# Dataset PyTorch que devuelva (img, metadatos, label)
# (Aquí aún no defino la clase, solo la idea)
# def __getitem__(self, idx):
#     img = cargar_imagen(df['image_path'][idx])
#     meta = df[cols_metadata].iloc[idx]
#     label = df['cat_code'][idx]
#     return img, meta, label

# Modelo: ResNet18 para imágenes + MLP para metadatos + concat + FC
```