# Preprocessing Pipeline: JSON to Transformer-Ready Dataset

## 1. Ingest Raw JSON Files

*Input: data/raw/*.json*

*Output: list of JSON dicts*

- Reads all JSON race files from disk
- Filters out corrupted/malformed ones
- Prepares for flattening

## 2. Flatten Each Race JSON

*Output: one flattened pd.DataFrame per race*

*Module: flatten_day.py*

- Extracts runner-level data
- Injects country, race_id, etc.
- Normalizes inconsistent structures

## 3. Concatenate All Flattened DataFrames

*Output: df_all_runners*

- Merges all races into one large runner-level DataFrame
- Enables batch processing and feature engineering

## 4. Clean and Optimize Data Types

*Output: type-corrected, memory-optimized DataFrame*

*Module: clean_flattened_df.py*

- Converts object to int, float, category
- Parses race_datetime into datetime64
- Drops unused/empty columns (e.g. GoingStick)

## 5. Add Embedding Indices for Categorical Fields

*Output: new columns like country_idx, going_idx, etc.*

*Module: add_embedding_indices.py*

- Applies LabelEncoder to key categorical fields
- Stores encoded values in _idx columns
- Makes categorical inputs transformer-compatible

## 6. Embed Text Fields

*Output: new columns like comment_vector, spotlight_vector*

*Module: process_text_fields.py*

- Uses pretrained SentenceTransformer model
- Converts comment and spotlight to vector representations
- Stores embeddings per row as np.array

### 7. Extract Domain-Specific NLP Features

*Output: binary flag columns (e.g. mentions_layoff_comment)*

*Included in: process_text_fields.py via extract_race_phrases()*

- Applies racing-focused regex rules to comment/spotlight
- Flags useful conditions like: trip change, ground preference, trainer form, etc.
- Adds explainable NLP insights

### 8. Save Final Model-Ready Dataset

*Output: model_ready_flat.pkl*

*Module: save_model_ready_df.py*

- Stores full cleaned, enriched DataFrame
- Can be reloaded immediately for training or inspection

### 9. Save Text Embeddings as .npz (Optional)

*Output: text_embeddings.npz*

*Module: embedding_io.py*

- Saves comment / spotlight embeddings as standalone npz
- Enables separate loading for memory efficiency or multi-modal models

### Optional: Schema Validation (on load)

*Function: load_embeddings_npz(..., expected_schema=...)*

- Ensures the .npz matches expectations before model training

### Final Output

- model_ready_flat.pkl  full tabular DataFrame
- text_embeddings.npz  optional fast access text input
- Full reproducibility from raw to ready