Dynamic Padding for Race-Level Transformer Batching

Date: 2025-03-31

1. Group by race_id

- Use DataFrame.groupby('race_id') to segment each race.
- Each group represents a single transformer sequence.

2. Filter races with fewer than 5 runners

- Transformer logic requires at least 5 entities to learn contextual relationships.
- Use a filter like: len(group) >= 5

3. Create batches of races

- Accumulate a fixed number of races (e.g., 32) into a batch.
- Avoid mixing races in the same batch - race is your sequence unit.

4. Find max runners in the batch

- Determine max sequence length using max(len(race) for race in batch).

5. Pad each race to max length

- Use zero-padding (or -1 for categorical) to expand each race matrix.
- Apply this across float features, embeddings, indices, targets, etc.

6. Create a mask

- Construct a binary mask: 1 = real runner, 0 = padded slot.
- Shape: (batch_size, max_runners_per_batch).

7. Package into a batch dictionary

- Use keys like: float_features, embedding_indices, comment_vector, mask, etc.
- Ensure shapes are consistent and tensor-ready.

8. (Optional) Truncate races with > max_len

- If you cap absolute max length (e.g., 24), truncate based on quality (rpr, or).
- Avoid dropping winners during truncation where possible.