## Note: 1: Define the Priority Metrics and Their Weights

From your screenshot, I interpreted the five priorities and their corresponding **weights** as follows:

Priority Description	Data Field	Weight
Teleop Coral L1	_where_(teleop)_[coral_l1]	3.0
Defensive Capability (Driver Skill)	driver_skill	2.0
Endgame (Your Robot Specifically)	endgame_score	2.0
Auto Mobility	auto_mobility_score	2.0
Auto Score	auto_score	1.5

## New Step 2: Load and Filter the Data

- Loaded your full scouting dataset (unified\_event\_2025lake.json)
- Removed all teams that were already picked:
  - 0 8044, 5653, 4087, 16, 456, 2036, 3526, 8808

## 🚺 Step 3: Extract Metric Values Per Team

For each team still in the pool:

- Pulled every match entry for that team.
- Collected values for the 5 metrics listed above:
  - Some of these are nested or prefixed with underscores (like \_where\_(teleop)\_[coral\_l1]).
  - o Some require interpreting "driver\_skill" as a proxy for Defensive Capability.

# Step 4: Compute Averages Per Metric Per Team

For each team:

- Calculated the average value of each metric across all their matches.
- This smoothed out inconsistencies or outliers in performance.

## Example for Team 9717:

• Avg Coral L1 (Teleop): 1.22

• Avg Driver Skill: 6.22

• Avg Endgame Score: 7.56

Avg Auto Mobility Score: 2.78

• Avg Auto Score: 5.67

## Step 5: Apply Weighted Scoring Formula

Used this formula:

plaintext

CopyEdit

Each metric average is multiplied by its assigned weight, then all weighted values are summed to generate the **final score**.

# 🔋 Step 6: Rank Teams by Score

- Sorted teams in descending order by their weighted score.
- Created the final ranked list.

Let me know if you'd like help matching this process to your own code—or if you want to add a new ranking strategy (e.g., for 3rd pick, 1st pick, defensive focus, etc.).

40

#### You said:

picklist\_generator\_service.py

Python

Ok - I am going to provide the generator file that I use. I recognize that you won't have the full context but pay special attention to how I am prompting GPT and see where I may be messing up. If you need any other files to help parse through it, please let me know.

#### ChatGPT said:

Thanks for sharing the generator logic. Here's a focused **diagnosis and feedback** based on the GPT prompting sections and where misranking issues may be emerging:

## **Q** Where Issues Might Arise

## 1. Too Much Delegation to GPT Without Hard Logic Safeguards

Your current design offloads the **entire ranking process** to GPT via natural language prompts. While this works well for general insight, GPT is not optimal at **precise numeric calculations**, especially when:

- You pass in many teams with nuanced metrics.
- Weights are expected to be rigorously applied.
- Teams have missing or inconsistent data.

**Consequence**: Teams may be ranked for qualitative reasons or heuristic guesses rather than strict weighted scoring.

Fix Suggestion: Precompute weighted scores yourself from the metrics and provide them in the prompt as a score field. Then instruct GPT to sort by this field unless overridden by synergy/strategy. This ensures consistency.

### 2. Prompt Ambiguity on Weight Interpretation

Your current user\_prompt logic includes this instruction:

"Weight 3.0 (CRITICAL): These metrics are the PRIMARY deciding factors and MUST dominate your decision..."

But GPT often struggles to **quantitatively enforce weights** without an explicit formula or example of how to prioritize a 3.0 weight over 2.0.

Fix Suggestion: Include a sample calculation like:

json

CopyEdit

"If metric A = 6.0 and weight = 3.0, and metric B = 8.0 with weight = 2.0, then A contributes more to the final score. Rank accordingly."

Even better: include a computed score field per team.

### 3. The Reason Limit May Bias Output

You're forcing reasons to be:

"≤12 words and cite ≥1 metric value"

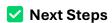
This limits the model's explanation fidelity and may lead it to **prioritize only high-value single metrics** (e.g., a flashy "Teleop Coral L1: 6") while ignoring better all-around contributors.

Fix Suggestion: Allow slightly longer reasons (e.g., 18 words) during testing. You can shorten after confidence in ordering is achieved.

### 4. Possible Token Truncation & Cut-Offs

If your model is truncating team data or context due to token limits (especially with gpt-4o), GPT might rank teams only from the first few hundred lines of data.

Fix Suggestion: Log len(user\_prompt) and len(messages) to confirm full team data is reaching GPT. Also consider chunking teams into batches of 25–30 when not using batch mode.



To help isolate the issue and fix it fast:

1. **Temporarily hard-code your weighted score calculation** into the JSON sent to GPT like this:

```
json
CopyEdit
{
  "team_number": 9717,
  "nickname": "Team Hydra",
  "weighted_score": 45.3,
  "metrics": { ... },
  ...
}
```

2. In the prompt, add:

"Unless strategy overrides it, sort teams primarily by weighted\_score, descending."

- 3. Run the generator again and compare with your manual rankings.
- 4. If needed, I can help you **write a drop-in function** to do this score calculation and JSON prep for use in \_prepare\_team\_data\_for\_gpt().