Used to determine pipelining for the Prediction Bot (RL FOCUSED)

Overview:

A prediction engine for Rocket League that takes in player stats, recent performances, and social signals (like tweets), then outputs Over/Under (O/U) style predictions like:

- *"Will Firstkiller score more than 4 goals in a Bo5?"*
- *"Does Chronic get more than 6 demos across 2 games?"*

Eventually, users can interact with the bot via a chat interface.

Stage 1: Data Collection (Scraping and Aggregation)

🔹 Sources

- **Liquipedia** (match history, team rosters, player matchups)
- **Ballchasing API** (if available—match replays & stats)
- **Twitter API (Tweepy)** (player moods, confidence, rumors, etc.)

Source	Data Needed
Liquidpedia	Upcoming matchups, Bo5/Bo7 format, teams, players
Ballchasing	Goals, assists, demos, saves, shot %, match speed
Twitter	Sentiment (NLP), confidence, team synergy hints

🔹 Tools

- **Selenium** for dynamic scraping (Liquipedia, if needed)
- **BeautifulSoup** for static pages
- **Tweepy** for Twitter
- Async queueing for efficient scraping

Stage 2: Prediction Engine

🔹 Core Model

Use **Neural Networks** or **XGBoost** with structured inputs:

- Game-specific context (opponent team, format, pressure)
- Player recent stats (last X games)
- Player historical data vs opponent
- Sentiment from Twitter

🔹 Targets

Predict probabilities for:

- Goals (Over/Under)
- Saves
- Demos
- Assists
- MVP chance (maybe)

🔹 Feature Engineering Ideas

- Rolling averages (e.g., last 5-game demo average)
- Opponent strength (team Elo or win rate)
- Twitter confidence (sentiment score normalized 0–1)
- Match pressure (elimination game? grand finals?)

Stage 3: Interactive Bot Layer

🔹 Concept

Let users query in plain English:

> "Think Beastmode gets 3+ goals in the next series?"

Bot parses:

- Player: *Beastmode*
- Stat: *Goals*
- O/U: *3+*
- Context: *Next series*

🔹 Output

Use rule-based templates mixed with model outputs:

“There is a 78% chance Beastmode scores 3+ goals in the next series. His current form is hot, averaging 1.3 goals/game over the last 5, and he’s playing a weaker defense in Team XYZ.”

Basic Pipelining:

[Scraper Layer] --> [Data Store] --> [Prediction Engine] --> [Bot Interface/API]

Daily scraping scheduler (Airflow or custom cron)

Data cache (SQLite, Postgres, or Firebase)

Model API (Flask/FastAPI)

Frontend/CLI/chat wrapper

Future Proofing:

Add “**locks of the day**” for automated betting recs

Track **prediction history** to show model confidence

Use **parlay builder**: predict multiple stats per player