JumpShot - CS2 Performance Analytics Platform

Complete Product Requirements Document

PLATFORM OVERVIEW

JumpShot is a Counter-Strike 2 (the videogame, specific to the esports / professional side of the game) performance analytics ecosystem that transforms raw competitive match data into actionable intelligence through advanced statistical modeling, machine learning algorithms, and comprehensive visualization tools.

The platform serves professional teams, coaches, analysts, scouts, and esports organizations with deep insights into player performance, team dynamics, match prediction, and strategic optimization.

The goal of the platform is to effectively ‘moneyball’ the data. Due to CS2’s incredible innate ability to be able to pull telemetry data - we can build incredible analytic layers to draw insights. Understand players and team chemistry and make match outcome predictions.

We are managing a separate database using Supabase (connected via secrets in Replit), Replit (you) will ingest data, process the data into advanced metrics and then further process that data into our various features (match predictor, scout etc)

KEY DOCUMENTATION:

* Refer to rules.md for interaction and working parameters (between Replit / Claude and myself)
* project.md : continually update and add to it. I have attached the start of the file.

CORE TECHNOLOGY ARCHITECTURE

* Frontend Framework: React.js 18+ with TypeScript, Framer Motion animations, Recharts visualization library
* Backend Infrastructure: Node.js/Express server with TypeScript, real-time WebSocket connections
* Primary Database: Supabase (PostgreSQL) with complete relational schema and foreign key constraints
* Authentication System: Session-based authentication
* Deployment Platform: Replit
* API Architecture: RESTful endpoints with comprehensive error handling and data validation

CORE DESIGN / UI

* Visual Style: Glassmorphism design with dark theme, gradient overlays, and frosted glass effects
* Color Palette: Dark backgrounds (slate-900/800) with vibrant accent colors (blue, purple, amber gradients)
* Typography: Clean, modern fonts with bold headings and readable body text
* Animation: Smooth Framer Motion animations with staggered card reveals and hover effects

Navigation Structure

* Sidebar Navigation: Fixed left sidebar with icons and labels of each feature.

Main Routes / Core features:

* LANDING: Dashboard
  + Customizable (salesforce type style) dashboard with various pre-made templates for different users. Users will be able to add / remove from their own custom dashboard. The purpose being that an individual user can track what they are most interested in.
* Players Tab and Individual players pages
  + Detailed below. But fundamentally a profile card listing of players ranked by their PIV. Further ability to click through to any player and deep dive into their PIV by role per side.
* Teams
  + Team ranking by PIV / TIR
* Match Predictor (AI-powered predictions)
  + Predictions based on a range of key metrics, including
    - Map pool matchup
    - Core strengths and weaknesses
    - Individual player factors
    - Form
    - Matchup history
* Scout (player recruitment tool)
  + Ability to match players to a team based on role fit, chemistry and value they can bring to a team

Players Page Layout

* Header Section: Search bar, role filter dropdown, sort options
* Player Grid: Responsive card grid (3-4 columns on desktop, 1-2 on mobile)

Player Cards:

* Display PIV with color coding: 90+ (gold), 75+ (blue), 60+ (green), below 60 (gray)
* Show role badges with T-side/CT-side indicators
* Hover effects showing PIV breakdown tooltip
* Click navigation to detailed player page
* Gradient header with role-based colors
* Player avatar circle with player images
* Team badge in top-right corner
* Role badges with icons
* PIV score prominently displayed with color coding
* Key metrics : T PIV, CT PIV & best metric for the player (eg. OSM, K/D, Flash assists etc)
* Hover effects with scale transforms and glow effects

Match Predictor Interface

* Team Selection: Two-panel layout for Team A vs Team B
* Prediction Display: Central area showing win probability with animated progress bars
* Factor Breakdown: Visual breakdown of prediction factors (PIV, team synergy, historical performance)
* Data Visualization Components
* Charts: Recharts library for clean, interactive charts
* Metrics Cards: Glassmorphism cards with gradient borders
* Progress Bars: Animated progress indicators for scores

Color Coding:

* Green (85+ PIV): Excellent performance
* Blue (70-84 PIV): Good performance
* Yellow (50-69 PIV): Average performance
* Orange (<50 PIV): Below average

Interactive Elements

* Hover States: Subtle scale transforms (1.02x-1.05x) with shadow changes
* Loading States: Skeleton loaders and spinner animations
* Transitions: 200-300ms duration with easing for smooth interactions

Technical Implementation Notes

* Framework: React with TypeScript
* Styling: Tailwind CSS with custom gradient classes
* Animations: Framer Motion for page transitions and micro-interactions
* Icons: Lucide React for consistent iconography
* State Management: React Query for data fetching and caching
* Key UI Patterns
* Cards: Rounded corners, backdrop blur, border glow effects
* Buttons: Gradient backgrounds with hover state changes
* Forms: Clean inputs with focus states and validation feedback
* Tables: Zebra striping with hover highlighting
* Badges: Small, rounded elements with role-specific colors

CORE PRODUCT FEATURES

{Phase 1}

PLAYERS → Ranks players by PIV value

PLAYER PAGES → Shows detailed data on an individual. Focuses on their core role metrics by t & ct sides.

TEAMS → Ranks team by TIR values

TEAM PAGES →

{Phase 2}

FUNDAMENTAL CALCULATION ARCHITECTURE

PIV (Player Impact Value) represents the most comprehensive player evaluation metric in competitive Counter-Strike 2 analytics, incorporating role-specific performance, individual consistency, team synergy contribution, and contextual factors.

Master Formula:

PIV = (RCS × ICF × SC × OSM) + Basic\_Metrics\_Bonus + Situational\_Modifiers + Map\_Specific\_Adjustments

PIV incorporates Role-Specific Metrics: Detailed stats tailored to key roles (IGL, AWPer, Spacetaker, Lurker, Anchor, Support) to evaluate individual performance in context.

A breakdown of the individual components to PIV and their relative weightings:

* RCS = Role Core Score (0.0 to 1.0)

This captures how effectively a player fulfills their role’s core responsibilities using key metrics.

Formula:  
RCS\_player = Sum(w\_k \* M\_k') from k=1 to n

* M\_k': A role-specific metric, normalized to 0–1 to ensure fair comparison across different scales. Normalization is calculated as:  
  M\_k' = (M\_k - min(M\_k)) / (max(M\_k) - min(M\_k)), where min and max are taken across all players in the dataset.
* w\_k: Weight assigned to each metric, summing to 1. Start with equal weights (e.g., 0.33 for 3 metrics) based on expert opinion, then adjust using data-driven methods like regression against total\_rounds\_won to reflect true impact.
* n: Number of metrics per role, typically 2–4, chosen to represent the role’s essence.
  + RCS focuses on an individual player's role in their team (assigned via Supabase, eg. AWP, Lurker, Rotator). Players always have 2 roles, 1 one attack (Terrorist or ‘t’ side) and one on defense (Counter terrorist or ‘ct’ side). The only exception is when a player is additionally assigned as an IGL (In-game leader).
* ICF = Individual Consistency Factor (0.0 to 2.0)
  + Measures player's consistency across matches and maps. Examples: Low variance in K/D, ADR, KAST across maps and matches.
* - SC = Synergy Contribution (0.0 to 1.0)
* - OSM = Opponent Strength Multiplier (0.8 to 1.2)
* - Basic\_Metrics\_Bonus = Fundamental statistical performance bonus (0.0 to 0.5)
* - Situational\_Modifiers = Context-based adjustments (-0.2 to +0.3)
* - Map\_Specific\_Adjustments = Map performance variance (-0.1 to +0.1)

The Player Impact Value (PIV) system is a comprehensive player rating model designed for CS2 that evaluates player performance based on their specific roles and playstyles. Unlike traditional statistics that focus solely on kills, deaths, and ADR, the PIV system uses a multi-dimensional approach to quantify a player's true impact on the game.

Core Components

The PIV calculation is based on the following core components:

1. Role Core Score (RCS)

The Role Core Score measures how effectively a player performs in their assigned role based on role-specific metrics.

* Calculation: Weighted average of normalized role-specific metrics
* Scale: 0-1 (higher is better)
* Role-specific metrics examples:
  + AWP: Opening Pick Success Rate, Multi Kill Conversion, AWP Flash Assistance
  + IGL: Strategic Success Rate, Utility Effectiveness, Team Performance Index
  + Support: Flash Assist Rate, Trade Kill Efficiency, Utility Usage Impact
  + Lurker: Isolated Kill Success, Map Control Value, Information Value
  + Spacetaker: Entry Success Rate, Space Creation Value, Trading Potential
  + Anchor: Site Hold Success, Multi-kill Potential, Defensive Utility Value
  + Rotator: Rotation Timing, Defensive Flexibility, Multi-site Impact

2. Individual Consistency Factor (ICF)

The Individual Consistency Factor measures a player's performance consistency across matches.

* Calculation: Based on statistical variance of key performance metrics across matches
* Scale: 0-1 (higher is more consistent)
* Adjustments:
  + IGLs receive a consistency bonus to account for cognitive load
  + High-fragging players (like ZywOo) receive a K/D weight adjustment

3. Synergy Contribution (SC)

The Synergy Contribution measures how well a player enhances team performance.

* Calculation: Based on role-specific team contribution metrics
* Scale: 0-1 (higher means better contribution to team synergy)
* Examples:
  + AWP: AWP Impact Rating (combining kills, opening picks, and utility)
  + IGL: Strategic Value (team performance with/without player)
  + Support: Support Effectiveness (flash assists, trade success)

4. Opponent Strength Multiplier (OSM)

The Opponent Strength Multiplier adjusts ratings based on the quality of opposition.

* Calculation: Based on opponent team ranking (TIR) and individual opponent skill
* Default: 1.0 (neutral)
* Range: 0.8-1.2 (lower for weaker opponents, higher for stronger opponents)

PIV Formula

The final PIV calculation combines these components in a weighted formula:

PIV = (RCS \* 0.35 + ICF \* 0.25 + SC \* 0.25) \* OSM \* RoleModifier \* 2

Where:

* RCS = Role Core Score (0-1)
* ICF = Individual Consistency Factor (0-1)
* SC = Synergy Contribution (0-1)
* OSM = Opponent Strength Multiplier (typically 0.8-1.2)
* RoleModifier = Role-specific adjustment factor:
  + AWP: 0.90x
  + Support: 1.08x
  + IGL: 1.05x
  + Spacetaker: 1.03x
  + Other roles: 1.00x

The result is multiplied by 2 to create a more intuitive scale, with values typically ranging from 1.0 to 2.5.

Side-specific PIV

The system calculates three PIV values for each player:

1. Overall PIV: The player's complete impact across all rounds
2. CT-side PIV: The player's impact on CT (Counter-Terrorist) side only
3. T-side PIV: The player's impact on T (Terrorist) side only

This allows for more nuanced analysis of player performance on different sides of the game.

Role Assignment System

Primary Role Detection

Players can have multiple roles:

1. CT-side Role: Role assigned for Counter-Terrorist side
2. T-side Role: Role assigned for Terrorist side
3. IGL Role: In-Game Leader (if applicable, overrides other roles)

For display purposes, the system determines the primary role using:

* For IGLs: Always displayed as IGL
* For non-IGLs: Determined by a weighted combination of CT and T roles (typically 50% CT / 50% T)

Role Assignment Process

1. Explicit Assignment: Roles are assigned based on the roles CSV file when available
2. Automatic Detection: When no explicit role data is available, roles are automatically detected based on play patterns:
   * Statistical patterns (e.g., AWP kill percentage, entry attempt frequency)
   * Position heat maps
   * Utility usage patterns
   * Known team roles (only one IGL per team)

Team Integration

The PIV system integrates with the Team Impact Rating (TIR) system:

1. Individual PIV feeds into team calculations
2. Role Coverage affects team synergy calculations
3. Team Composition Analysis uses PIV to identify optimal role combinations

Display Format

* PIV values are displayed as whole numbers (e.g., 0.798 → 80)
* K/D ratios are shown to 2 decimal places
* Player icons show team initials instead of player's first initial

Future Enhancements

Planned improvements to the PIV system include:

1. Economy Impact Factor: Measuring a player's economic impact and efficiency
2. Clutch Performance Index: Specialized rating for clutch situations
3. Map-specific Role Adjustments: Tailoring role evaluations to specific maps
4. Mental State Analysis: Incorporating signs of tilt or confidence
5. Role Synergy Modeling: Evaluating how well certain role combinations work together

T-SIDE ROLE DEFINITIONS & WEIGHTINGS

-

IF A PLAYER DOES NOT HAVE A ROLE ASSIGNED TO THEM IN THE SUPABASE DB, DISREGARD THEM ENTIRELY UNTIL THEY DO.

Recharts remains the charting engine yet every network call now flows through a single TanStack React Query client so dashboard interactions feel immediate even on shaky links.

→ QueryClient sits in \_app.tsx, supplies stale-while-revalidate caching, and retries failed reads up to three times with exponential back-off  
 → supabase-wrapped rpcs return promises that drop straight into useQuery('player-round', …); no component assembles SQL or handles fetch state by hand  
 → invalidation keys (match\_id, player\_id) refresh automatically when a new match arrives because the ingest worker emits supabase.realtime broadcasts that the query client maps to queryClient.invalidateQueries

This single-pattern fetch layer eliminates divergent caching hacks and ensures PIV drill-downs transition smoothly: React Query serves cached rows in <50 ms, fetches fresh Z-scores in the background, and Recharts redraws without a visible spinner.

With observability baked into the api and a unified data-fetch pattern in the ui the team gains tight feedback loops; modelling tweaks surface as latency graphs and front-end iterations no longer fight inconsistent state handling.

COMPREHENSIVE FEATURE SPECIFICATION

1. PLAYERS PAGE

HEADER: Players

SUB-HEADER: Ranked by Player Impact Value (PIV)

* A listicle of player cards, ranked by highest to lowest PIV, of the players present in the dataset.
* Real-time PIV calculations from Supabase data
* • Advanced filtering by role, team, PIV range
* • Role-based color coding and hierarchy
* • Search with autocomplete
* • Performance heat maps per card
* • Batch comparison (up to 6 players)

2. TEAMS PAGE

* • Team Impact Rating (TIR) calculations
* • Roster composition visualization
* • Team synergy metrics
* • Recent match results
* • Role distribution analysis
* • Map pool analysis
* • Strategic preferences breakdown

3. PLAYER DETAIL PAGE

* • PIV timeline with interactive details
* • Component breakdown (RCS, ICF, SC, OSM)
* • Performance radar charts
* • Match-by-match performance grid
* • Career highlights timeline
* • Role-specific metric breakdowns
* • Predictive analytics integration

4. TEAM DETAIL PAGE

* • Current roster with detailed cards
* • Role assignment optimization
* • Player chemistry matrix
* • Tactical analysis dashboard
* • Economic efficiency metrics
* • Anti-strat analysis
* • Performance impact of changes

5. ROLE WEIGHTINGS PAGE

* • Complete PIV formula explanations
* • Role-specific weight breakdowns
* • Interactive calculation tools
* • Component calculation methods
* • Normalization procedures
* • Update history tracking
* • A/B testing framework

6. DOCUMENTATION PAGE

* • Database schema explanations
* • API endpoint documentation
* • Data collection methodologies
* • User guides with tutorials
* • Advanced analytics interpretation
* • Troubleshooting guides
* • Methodology research papers

7. PLAYER COMPARISONS

* • Side-by-side player cards
* • Interactive metric selection
* • Performance timeline comparison
* • Role-adjusted comparisons
* • Radar chart overlays
* • Statistical significance testing
* • Market value analysis

8. MATCH PREDICTOR

* • Pre-match analysis algorithms
* • Team composition analysis
* • Map pool analysis with veto prediction
* • Live match integration
* • Real-time PIV updates
* • Prediction accuracy tracking
* • Post-match analysis

9. MATCH INFOGRAPHIC

* • Automated infographic generation
* • Match summary with highlights
* • Player performance spotlights
* • Timeline of key moments
* • Social media optimization
* • Brand integration capabilities
* • Real-time generation

10. SCOUT PAGE

* • Advanced player discovery
* • Hidden gem identification
* • Team fit analysis
* • Budget-based search
* • Comprehensive scouting reports
* • Risk assessment for transfers
* • Development potential analysis

11. SEARCH PLAYERS

* • Multi-criteria filtering
* • Performance range sliders
* • Geographic filtering
* • AI-powered recommendations
* • Similar player suggestions
* • Saved search functionality
* • Export capabilities

12. STATISTICAL ANALYSIS

* • Correlation analysis tools
* • Regression modeling
* • Cluster analysis
* • Time series analysis
* • Hypothesis testing framework
* • Interactive statistical plots
* • Research tools

13. DATA VISUALIZATION

* • Real-time dashboard KPIs
* • Interactive timeline visualization
* • Geographic heat maps
* • Network diagrams
* • Customizable layouts
* • Export options
* • Mobile-optimized viewing

14. ADVANCED ANALYTICS

* • Machine learning integration
* • Performance prediction models
* • Meta game evolution tracking
* • Strategic trend analysis
* • Competitive intelligence
* • Experimental features
* • Blockchain integration

15. DASHBOARD PAGE

* • Personalized widget arrangement
* • Favorite players tracking
* • Custom metric calculations
* • Real-time updates
* • Live match integration
* • Cross-platform synchronization
* • Performance alerts

COMPLETE IMPLEMENTATION ROADMAP

1: CORE INFRASTRUCTURE: Database Architecture

* • Supabase schema implementation
* • Foreign key constraints
* • Data migration scripts
* • Query optimization
* • Backup procedures

2: Authentication & Security

* • User authentication system
* • Role-based access control
* • API security implementation
* • Password hashing
* • Admin panel creation

3: Core API Development

* • RESTful API endpoints
* • Database optimization
* • Error handling systems
* • API documentation
* • Testing framework

4: PIV Calculation Engine

* • Complete PIV algorithm
* • Role-specific calculations
* • Data normalization
* • Performance optimization
* • Accuracy testing

PHASE 2: CORE FEATURES

5: Players Page

* • Real-time PIV calculations
* • Advanced filtering
* • Search implementation
* • Player card design
* • Performance optimization

6: Player Detail Pages

* • Individual analysis
* • Performance timelines
* • Role breakdowns
* • Historical tracking
* • Predictive analytics

7: Teams Page

* • TIR calculations
* • Team detail pages
* • Roster analysis
* • Team comparisons
* • Strategic visualization

Week 8: Visualization System

* • Chart integration
* • Timeline visualization
* • Radar charts
* • Comparison tools

PIV CALCULATION IMPLEMENTATION DETAILS

Exact Formula Implementation:

RCS = Σ(normalized\_metric\_i × role\_weight\_i) for i in role\_specific\_metrics

ICF = base\_performance\_factor × consistency\_multiplier × (isIGL ? 1.15 : 1.0)

SC = role\_synergy\_metric × team\_coordination\_factor

OSM = 1.0 + (opponent\_avg\_ranking - 10) × 0.02 (clamped between 0.8 and 1.2)

PIV = (RCS × ICF × SC × OSM) × 100 (scaled to 0-100 range)

Normalization Requirements:

* • Use Z-score normalization: (value - mean) / standard\_deviation, then clamp to [0, 1]
* • Calculate normalization parameters per event\_id to ensure fair comparison within tournaments
* Handle outliers by applying 99th percentile caps before normalization
* Store normalization parameters in database for consistency across calculations

FRONTEND COMPONENT ARCHITECTURE

Data Loading States:

* • Skeleton loaders during API calls
* • Error boundaries for failed requests
* • Retry mechanisms for temporary failures
* • Optimistic updates for user interactions

DATABASE QUERY OPTIMIZATION REQUIREMENTS

* • Indexes Required: steam\_id (primary), team\_id, event\_id, role combinations
* • Query Patterns: Use prepared statements with parameter binding for all dynamic queries
* • Connection Pooling: Implement connection pooling with max 20 connections, 5 second timeout
* Supabase Edge Functions already offer row-level cache-control. Lean on that before introducing another service
* • Batch Operations: Group related queries using transactions for consistency

INTERACTION PATTERNS

* Loading states for all async operations
* Error states with actionable messaging
* Success feedback for user actions
* Intuitive navigation with clear active states

## DATABASE SCHEMA DEFINITIONS:

Below is the schema definition for the tables available in the Supabase database.

kill\_stats:

create table public.kill\_stats (

steam\_id bigint not null,

kills integer null,

headshots integer null,

wallbang\_kills integer null,

no\_scope integer null,

through\_smoke integer null,

airbone\_kills integer null,

blind\_kills integer null,

victim\_blind\_kills integer null,

awp\_kills integer null,

pistol\_kills integer null,

first\_kills integer null,

ct\_first\_kills integer null,

t\_first\_kills integer null,

first\_deaths integer null,

ct\_first\_deaths integer null,

t\_first\_deaths integer null,

event\_id integer not null,

constraint kill\_stats\_pkey primary key (steam\_id, event\_id),

constraint unique\_kill\_event unique (steam\_id, event\_id),

constraint unique\_steam\_event\_kill\_stats unique (steam\_id, event\_id),

constraint fk\_kill\_stats\_event foreign KEY (event\_id) references events (event\_id),

constraint kill\_stats\_steam\_id\_fkey foreign KEY (steam\_id) references players (steam\_id)

) TABLESPACE pg\_default;

genral\_stats:

create table public.general\_stats (

steam\_id bigint not null,

assists double precision null,

deaths double precision null,

trade\_kills double precision null,

trade\_deaths double precision null,

kd double precision null,

k\_d\_diff double precision null,

adr\_total double precision null,

adr\_ct\_side double precision null,

adr\_t\_side double precision null,

kast\_total double precision null,

kast\_ct\_side double precision null,

kast\_t\_side double precision null,

total\_rounds\_won double precision null,

t\_rounds\_won double precision null,

ct\_rounds\_won double precision null,

event\_id integer not null,

constraint general\_stats\_pkey primary key (steam\_id, event\_id),

constraint unique\_general\_event unique (steam\_id, event\_id),

constraint unique\_steam\_event\_general\_stats unique (steam\_id, event\_id),

constraint fk\_general\_stats\_event foreign KEY (event\_id) references events (event\_id),

constraint general\_stats\_steam\_id\_fkey foreign KEY (steam\_id) references players (steam\_id)

) TABLESPACE pg\_default;

utility\_stats:

create table public.utility\_stats (

steam\_id bigint not null,

assisted\_flashes integer null,

flahes\_thrown integer null,

ct\_flahes\_thrown integer null,

t\_flahes\_thrown integer null,

flahes\_thrown\_in\_pistol\_round integer null,

he\_thrown integer null,

ct\_he\_thrown integer null,

t\_he\_thrown integer null,

he\_thrown\_in\_pistol\_round integer null,

infernos\_thrown integer null,

ct\_infernos\_thrown integer null,

t\_infernos\_thrown integer null,

infernos\_thrown\_in\_pistol\_round integer null,

smokes\_thrown integer null,

ct\_smokes\_thrown integer null,

t\_smokes\_thrown integer null,

smokes\_thrown\_in\_pistol\_round integer null,

util\_in\_pistol\_round integer null,

total\_util\_thrown integer null,

total\_util\_dmg integer null,

ct\_total\_util\_dmg integer null,

t\_total\_util\_dmg integer null,

event\_id integer not null,

constraint utility\_stats\_pkey primary key (steam\_id, event\_id),

constraint unique\_steam\_event\_utility\_stats unique (steam\_id, event\_id),

constraint unique\_utility\_event unique (steam\_id, event\_id),

constraint fk\_utility\_stats\_event foreign KEY (event\_id) references events (event\_id),

constraint utility\_stats\_steam\_id\_fkey foreign KEY (steam\_id) references players (steam\_id)

) TABLESPACE pg\_default;

teams:

create table public.teams (

id serial not null,

team\_clan\_name text not null,

constraint teams\_pkey primary key (id),

constraint teams\_team\_clan\_name\_key unique (team\_clan\_name)

) TABLESPACE pg\_default;

players:

create table public.players (

steam\_id bigint not null,

user\_name text null,

constraint players\_pkey primary key (steam\_id)

) TABLESPACE pg\_default;

player\_match\_summary:

create table public.player\_match\_summary (

steam\_id bigint not null,

file\_id integer not null,

team\_id integer null,

event\_id integer not null,

constraint player\_match\_summary\_pkey primary key (steam\_id, file\_id, event\_id),

constraint unique\_steam\_file\_event unique (steam\_id, file\_id, event\_id),

constraint fk\_player\_match\_summary\_event foreign KEY (event\_id) references events (event\_id),

constraint player\_match\_summary\_event\_id\_fkey foreign KEY (event\_id) references events (event\_id),

constraint player\_match\_summary\_file\_id\_fkey foreign KEY (file\_id) references matches (file\_id),

constraint player\_match\_summary\_steam\_id\_fkey foreign KEY (steam\_id) references players (steam\_id)

) TABLESPACE pg\_default;

rounds:

create table public.rounds (

id serial not null,

round\_num integer null,

start integer null,

freeze\_end integer null,

"end" integer null,

official\_end integer null,

winner text null,

reason text null,

bomb\_plant double precision null,

bomb\_site text null,

ct\_team\_clan\_name text null,

t\_team\_clan\_name text null,

winner\_clan\_name text null,

ct\_team\_current\_equip\_value double precision null,

t\_team\_current\_equip\_value double precision null,

ct\_losing\_streak integer null,

t\_losing\_streak integer null,

ct\_buy\_type text null,

t\_buy\_type text null,

advantage\_5v4 text null,

file\_id integer null,

event\_id integer null,

match\_name text null,

constraint rounds\_pkey primary key (id),

constraint unique\_round\_per\_match unique (round\_num, match\_name),

constraint rounds\_file\_id\_fkey foreign KEY (file\_id) references matches (file\_id)

) TABLESPACE pg\_default;

matches:

create table public.matches (

file\_id integer not null default nextval('matches\_file\_id\_seq'::regclass),

match\_name text not null,

event\_id integer null,

constraint matches\_pkey primary key (file\_id),

constraint unique\_file\_per\_event unique (match\_name, event\_id),

constraint matches\_event\_id\_fkey foreign KEY (event\_id) references events (event\_id)

) TABLESPACE pg\_default;

events:

create table public.events (

event\_id integer not null,

event\_name text not null,

constraint events\_pkey primary key (event\_id)

) TABLESPACE pg\_default;

player\_history:

create table public.player\_history (

id serial not null,

steam\_id bigint not null,

team\_id integer not null,

constraint player\_history\_pkey primary key (id),

constraint unique\_steam\_team unique (steam\_id, team\_id),

constraint fk\_steam foreign KEY (steam\_id) references players (steam\_id),

constraint fk\_team foreign KEY (team\_id) references teams (id)

) TABLESPACE pg\_default;

Initial MVP for full clarity and potential knowledge gaps:  
  
**MVP Structure: Player Impact Value (PIV {for scouting}) & Team Impact Rating (TIR{for predicting})**

**Core Components**

* **Role-Specific Metrics**: Detailed stats tailored to key roles (IGL, AWP, Spacetaker, Lurker, Anchor, Support) to evaluate individual performance in context.
* **Player Impact Value (PIV)**: A composite score per player that combines role effectiveness, consistency, synergy, and opponent strength adjustment to rank players for scouting.
* **Team Impact Rating (TIR)**: A team-level metric that aggregates PIVs and adds a synergy bonus to predict match outcomes.

**Output**

* **Scouting**: Rank players by PIV to identify top talent in each role.
* **Match Prediction**: Compare TIRs between teams to forecast winners (e.g., TIR\_A > TIR\_B → Team A favored).

**Role-Specific Metrics**

Below are the metrics for each role, calculated directly from your data where possible, with proxies noted.

**General IGL**

* **Tactical Timeout Success (TTSR)**: Rounds won after timeout / Total rounds after timeout.
  + **Data**: total\_rounds\_won post-timeout (assumes timeout flag exists in demos).
  + **Why**: Measures IGL’s ability to shift momentum with strategic pauses.
* **Utility Setup Optimization (USO)**: (assisted\_flashes + he\_thrown + infernos\_thrown) / total\_util\_thrown.
  + **Data**: Utility stats from demos.
  + **Why**: Reflects IGL’s effectiveness in enabling team plays through utility.
* **Clutch Conversion Rate (CCR)**: Sum(1vX\_won) / Sum(1vX).
  + **Data**: 1v1\_won, 1v2\_won, etc., from HLTV/demo.
  + **Why**: Captures IGL’s adaptability in high-pressure situations.

**T-Side IGL**

* **Opening Play Success Rate (OPSR)**: bomb\_planted\_count / t\_rounds\_won.
  + **Data**: Bomb plants and T-side wins from demos.
  + **Why**: Gauges success of initial T-side strategies.
* **Economy Optimization Index (EOI)**: t\_rounds\_won / (inferred buy cost from t\_flahes\_thrown + t\_infernos\_thrown).
  + **Data**: T-side wins and utility usage as a cost proxy.
  + **Why**: Tracks efficient resource management.
* **Fake Engagement Efficiency (FEE)**: (t\_flahes\_thrown or t\_smokes\_thrown in non-plant rounds) / (t\_rounds\_won from fakes).
  + **Data**: Utility in non-execute rounds tied to wins (proxy).
  + **Why**: Measures deception effectiveness.

**CT-Side IGL**

* **Site Hold Efficiency (SHE)**: (ct\_rounds\_won - bomb\_planted\_count) / ct\_rounds\_won.
  + **Data**: CT wins and bomb plants from demos.
  + **Why**: Assesses defensive setup strength.
* **Save Call Efficiency (SCE)**: (Rounds with low kill\_count but high survival) / (next-round total\_rounds\_won).
  + **Data**: Survival and subsequent wins from demos (proxy).
  + **Why**: Evaluates economic preservation.
* **Utility Setup Optimization (USO)**: (assisted\_flashes + ct\_he\_thrown + ct\_infernos\_thrown) / total\_util\_thrown.
  + **Data**: CT-side utility stats.
  + **Why**: Tracks utility impact on defense.

**T-Side AWP**

* **Opening Pick Success Rate (OPSR)**: t\_first\_kills / (t\_first\_kills + t\_first\_deaths).
  + **Data**: T-side first kills/deaths, AWP inferred from kills\_x.
  + **Why**: Measures entry kill success.
* **Multi-Kill Conversion (MKC)**: (multi-kill rounds) / (rounds with kill\_count ≥ 1).
  + **Data**: kills\_x filtered for AWP multi-kills.
  + **Why**: Captures ability to turn picks into round wins.
* **Flash-Assist Synergy (FAS)**: assisted\_flashes / kills\_x (T-side only).
  + **Data**: Flash assists and T-side kills.
  + **Why**: Reflects coordination with teammates.

**CT-Side AWP**

* **Site Lockdown Rate (SLR)**: (ct\_rounds\_won - bomb\_planted\_count) / ct\_rounds\_won where AWP has kill\_count.
  + **Data**: CT wins, bomb plants, kills\_x.
  + **Why**: Assesses site denial.
* **Early Pick Security (EPS)**: ct\_first\_kills / (ct\_first\_kills + ct\_first\_deaths).
  + **Data**: CT-side first kills/deaths, AWP inferred.
  + **Why**: Tracks opening duel wins.
* **Entry Denial Efficiency (EDE)**: ct\_first\_kills / ct\_first\_deaths.
  + **Data**: CT-side first engagements.
  + **Why**: Measures duel success against entries.

**T Spacetaker**

* **Opening Duel Success Rate (ODSR)**: t\_first\_kills / (t\_first\_kills + t\_first\_deaths).
  + **Data**: T-side first kills/deaths.
  + **Why**: Evaluates entry aggression.
* **Trade Conversion Rate (TCR)**: trade\_kills / trade\_deaths.
  + **Data**: Trade stats from demos.
  + **Why**: Tracks trading efficiency.
* **Aggression Efficiency Index (AEI)**: (kill\_count - death\_count) / (t\_first\_kills + t\_first\_deaths).
  + **Data**: Kills, deaths, engagements.
  + **Why**: Balances aggression with survival.

**T Lurker**

* **Flank Success Rate (FSR)**: through\_smoke / kill\_count.
  + **Data**: through\_smoke as a flank proxy.
  + **Why**: Gauges flank impact.
* **Clutch Conversion Rate (CCR)**: Sum(1vX\_won) / Sum(1vX).
  + **Data**: 1vX stats from HLTV/demo.
  + **Why**: Measures late-round clutch ability.
* **Information Gathering Efficiency (IGE)**: t\_rounds\_won with low kill\_count / total rounds.
  + **Data**: Wins with minimal kills (proxy).
  + **Why**: Reflects intel contribution.

**CT Anchor**

* **Site Hold Success Rate (SHSR)**: (ct\_rounds\_won - bomb\_planted\_count) / ct\_rounds\_won.
  + **Data**: CT wins and bomb plants.
  + **Why**: Assesses site defense.
* **Opponent Entry Denial Rate (OEDR)**: ct\_first\_kills / ct\_first\_deaths.
  + **Data**: CT-side first duels.
  + **Why**: Tracks entry prevention.
* **Multi-Kill Defense Ratio (MKDR)**: (multi-kill rounds) / ct\_rounds\_won.
  + **Data**: kills\_x on CT side.
  + **Why**: Captures multi-kill impact.

**T Support**

* **Flash Assist Synergy (FAS)**: assisted\_flashes / t\_flahes\_thrown.
  + **Data**: Flash assists and throws.
  + **Why**: Measures utility aiding kills.
* **Bomb Plant Utility Coverage (BPUC)**: bomb\_planted\_count with t\_smokes\_thrown / t\_rounds\_won.
  + **Data**: Plants and smokes.
  + **Why**: Tracks plant support.
* **Utility Setup Efficiency (USE)**: (bomb\_planted\_count + assisted\_flashes) / total\_util\_thrown.
  + **Data**: Utility and objective stats.
  + **Why**: Evaluates entry facilitation.

**CT Support**

* **Anti-Exec Utility Success (AEUS)**: (ct\_rounds\_won - bomb\_planted\_count) / ct\_rounds\_won with ct\_util\_thrown.
  + **Data**: CT wins, plants, utility.
  + **Why**: Measures execute disruption.
* **Crossfire Setup Rate (CSR)**: assisted\_flashes leading to kill\_count / ct\_rounds\_won.
  + **Data**: Flash assists and kills.
  + **Why**: Tracks crossfire support.
* **Retake Utility Coordination (RUC)**: bomb\_defused\_count with ct\_flahes\_thrown / retake attempts.
  + **Data**: Defuses and flashes (proxy).
  + **Why**: Assesses retake aid.

**Team Macro**

* **Chain-Kill Conversion (CKC)**: (multi-kill chains) / total\_kill\_count.
  + **Data**: kills\_x with timestamp clustering.
  + **Why**: Reflects team momentum.

**MVP Implementation Plan**

**Scope**

* Analyze 10 matches (~200 rounds) from top teams (e.g., NAVI, FaZe) using your dataset.

**Metric Calculation**

* **Output**: Per-player metric table (e.g., ZywOo: OPSR = 0.78, MKC = 0.50, FAS = 0.40).
* **Process**:
  + Extract raw stats (e.g., kill\_count, t\_first\_kills) from HLTV and demos.
  + Compute metrics as defined above, normalizing where specified (e.g., AEI).

**Role Assignment**

* **Infer Roles**: Assign based on data patterns:
  + **IGL**: High total\_util\_thrown + Sum(1vX\_won), moderate kd (not lowest fragger).
  + **AWP**: High kills\_x with AWP inferred, high t\_first\_kills (T) or ct\_first\_kills (CT).
  + **Spacetaker**: High t\_first\_kills + trade\_kills.
  + **Lurker**: High through\_smoke + 1vX\_won, low t\_first\_kills.
  + **Anchor**: High ct\_first\_kills + (ct\_rounds\_won - bomb\_planted\_count).
  + **Support**: High assisted\_flashes + bomb\_planted\_count (T) or bomb\_defused\_count (CT).
* **Why**: Ensures roles align with playstyles using available stats.

**Step 1: Player Impact Value (PIV)**

* **Formula**: PIV\_player = [(RCS\_player \* ICF) + SC\_player] \* OSM
  + **RCS\_player**: Sum(w\_k \* M\_k') from k=1 to n (e.g., 0.33 \* OPSR + 0.33 \* MKC + 0.33 \* FAS).
    - Normalize M\_k' to 0–1: M\_k' = (M\_k - min(M\_k)) / (max(M\_k) - min(M\_k)).
    - Initial w\_k = equal (e.g., 0.33), tune later via regression.
  + **ICF**: 1 / (1 + sigma\_role), sigma\_role from role-specific per-round scores (e.g., kill\_count + trade\_kills for Spacetaker).
  + **SC\_player**: Sum(v\_j \* S\_j) from j=1 to m (e.g., v\_j = 1, S\_j = UES = 0.7).
  + **OSM**: (Opponent HLTV Rating) / (League Avg HLTV Rating).
* **Output**: Ranked players by PIV (e.g., ZywOo PIV = 1.25).

**Step 2: Team Impact Rating (TIR)**

* **Formula**: TIR = SumPIV + TSL
  + **SumPIV**: Sum(PIV\_player) for all 5 players.
  + **TSL**: alpha \* CKC, alpha = 0.1, CKC = (multi-kill chains) / total\_kill\_count.
* **Prediction**: TIR\_A > TIR\_B → Team A favored (e.g., NAVI TIR = 5.11 vs. G2 TIR = 4.95).
* **Output**: Team TIRs for match comparisons.

**Step 3: Validation**

* **Test**:
  + PIV vs. mvp awards: Do high-PIV players align with MVPs?
  + TIR vs. match outcomes: Does higher TIR predict wins >50% of the time?
* **Adjust**: Refine weights (w\_k, v\_j, alpha) if correlations are weak, using regression on total\_rounds\_won.

**Step 4: Presentation**

* **Dashboard**:
  + Player Rankings: Table of top PIVs per role (e.g., ZywOo T AWP- PIV = 1.25).
  + Match Predictions: TIR\_A vs. TIR\_B with win probability (e.g., NAVI 52% vs. G2).

**CS2 Rating System: Player Impact Value (PIV) & Team Impact Rating (TIR)**

| Player Name | Role | Metric 1 (M\_1) | Metric 2 (M\_2) | Metric 3 (M\_3) | RCS (Sum w\_k \* M\_k') | Sigma\_role | ICF (1 / (1 + Sigma)) | SC (Synergy) | OSM (Opponent Strength) | PIV ([RCS \* ICF + SC] \* OSM) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Player A** | **Spacetaker** | **ODSR = 0.60** | **AEI = 0.9 (0.75 norm)** | **TCR = 0.70** | **0.68 (0.33 \* 0.60 + 0.33 \* 0.75 + 0.33 \* 0.70)** | **1.2** | **0.45** | **0.7 (UES)** | **1.2 (HLTV 1.2 / 1.0)** | **1.20 ([0.68 \* 0.45 + 0.7] \* 1.2)** |
| **Player B** | **T AWP** | **OPSR = 0.75** | **MKC = 0.50** | **-** | **0.63 (0.5 \* 0.75 + 0.5 \* 0.50)** | **0.8** | **0.56** | **0.4 (FAS)** | **1.0 (HLTV 1.0 / 1.0)** | **0.75 ([0.63 \* 0.56 + 0.4] \* 1.0)** |
| **Player C** | **IGL** | **OPSR = 0.55** | **USO = 0.65** | **-** | **0.60 (0.5 \* 0.55 + 0.5 \* 0.65)** | **1.0** | **0.50** | **0.6 (IIR)** | **1.1 (HLTV 1.1 / 1.0)** | **0.99 ([0.60 \* 0.50 + 0.6] \* 1.1)** |

**1. Player Impact Value (PIV)**

The PIV measures a player’s total contribution to their team by combining their role-specific performance, consistency across rounds, synergy with teammates, and adjustment for opponent strength. It’s designed to scout top talent by evaluating players in their specific roles and predict match outcomes when aggregated.

**A. Role-Core Score (RCS)**

This captures how effectively a player fulfills their role’s core responsibilities using key metrics.

**Formula**:  
RCS\_player = Sum(w\_k \* M\_k') from k=1 to n

* M\_k': A role-specific metric, normalized to 0–1 to ensure fair comparison across different scales. Normalization is calculated as:  
  M\_k' = (M\_k - min(M\_k)) / (max(M\_k) - min(M\_k)), where min and max are taken across all players in the dataset.
* w\_k: Weight assigned to each metric, summing to 1. Start with equal weights (e.g., 0.33 for 3 metrics) based on expert opinion, then adjust using data-driven methods like regression against total\_rounds\_won to reflect true impact.
* n: Number of metrics per role, typically 2–4, chosen to represent the role’s essence.

**Example Metrics by Role**:

* **Spacetaker**:
  + ODSR = first\_kill\_count / (first\_kill\_count + first\_death\_count): Measures success in opening duels.
  + AEI = (kill\_count - death\_count) / (first\_kill\_count + first\_death\_count): Balances aggression with survival (normalized).
  + TCR = trade\_kills / trade\_deaths: Tracks ability to convert teammate deaths into trades.
* **Lurker**:
  + FSR = through\_smoke / kill\_count: Gauges flank effectiveness (proxy due to limited positional data).
  + CCR = Sum(1vX\_won) / Sum(1vX): Assesses clutch success in 1vX scenarios.
* **T-Side AWP**:
  + OPSR = t\_first\_kills / (t\_first\_kills + t\_first\_deaths): Success rate of opening picks with AWP.
  + MKC = (multi-kill rounds) / (rounds with kill\_count ≥ 1): Multi-kill conversion from kills\_x.
* **IGL**:
  + OPSR = bomb\_planted\_count / t\_rounds\_won: Success of initial T-side strategies.
  + USO = (assisted\_flashes + he\_thrown + infernos\_thrown) / total\_util\_thrown: Utility effectiveness in setups.

Compute each M\_k from our data (e.g., first\_kill\_count, kills\_x), normalize across players, assign initial w\_k (e.g., 0.5, 0.5 for 2 metrics), and sum.

**Normalize**: Metrics like ODSR (0–1) and AEI (unbounded) have different ranges. Normalization ensures each contributes equally to RCS, avoiding skew from unscaled values.

**B. Individual Consistency Factor (ICF)**

This reflects how steady a player’s performance is round-to-round, rewarding reliability—a key trait in pro CS2.

* **Formula**:  
  ICF = 1 / (1 + sigma\_role)
  + sigma\_role: Standard deviation of a role-specific per-round performance score, calculated across all rounds in a match or tournament.
  + Scaled to 0–1 naturally (e.g., high sigma = low ICF, low sigma = high ICF). Optionally scale to 0–100 by multiplying by 100 for easier interpretation.

The 1 / (1 + sigma) structure penalizes inconsistency (high sigma lowers ICF) while keeping it simple and bounded. It mirrors your original CRCI concept but adapts it for role-specific needs.

* **Role-Specific Performance Scores**:
  + Spacetaker: kill\_count + trade\_kills per round: Focuses on aggressive output.
  + Lurker: 1vX\_won + through\_smoke per round: Highlights clutch and stealth impact.
  + AWP: kill\_count (AWP kills if tagged in kills\_x) per round: Emphasizes fragging consistency.
  + IGL: total\_util\_thrown + assist\_count per round: Captures strategic influence over frags.
  + Anchor: ct\_first\_kills + (ct\_rounds\_won - bomb\_planted\_count) per round: Reflects site defense stability.
  + Support: assisted\_flashes + bomb\_planted\_count (T) or bomb\_defused\_count (CT) per round: Utility and objective support.
* **Implementation**:
  + Compute a per-round score
  + Calculate sigma\_role across rounds (e.g., Python’s numpy.std()).
  + Apply the formula. Example: If sigma = 1.2, ICF = 1 / (1 + 1.2) = 0.45.

**Why Role-Specific?**: Generic scores (e.g., kills + assists) might undervalue Supports or IGLs who don’t frag heavily but excel in their niche.

**C. Synergy Contribution (SC)**

This quantifies a player’s direct teamwork with specific teammates, rewarding synergy that boosts team success.

* **Formula**:  
  SC\_player = Sum(v\_j \* S\_j) from j=1 to m
  + S\_j: A synergy metric involving the player (e.g., pair or trio stats).
  + v\_j: Weight for each synergy metric, summing to 1. Start equal (e.g., 0.5 for 2 metrics), tune via regression on total\_rounds\_won.
  + m: Number of synergy metrics, typically 1–3, based on key teammate interactions.
* **Why Only Direct Synergy?**: SC focuses on the player’s specific contributions to pairs or trios (e.g., Spacetaker + Support), leaving broader team synergy to TIR’s TSL.
* **Example Metrics**:
  + **Spacetaker + Support**: UES = (assisted\_flashes by Support) / (t\_first\_kills by Spacetaker) in same rounds: Measures utility aiding entries.
  + **Lurker + IGL**: IIR = (t\_rounds\_won with Lurker through\_smoke) / (IGL total\_util\_thrown rounds): Links intel to strategy success.
  + **AWP + Anchor**: ROR = (bomb\_defused\_count with AWP kills\_x) / (retake attempts): Tracks retake coordination.
* **Implementation**:
  + Use proxies due to limited pairing data: e.g., assisted\_flashes (Support) tied to t\_first\_kills (Spacetaker) in winning rounds.
  + Assign v\_j (e.g., 1 if only one metric), sum for SC.
  + Example: UES = 0.7, v\_j = 1, SC = 0.7.

**D. Opponent Strength Multiplier (OSM)**

This adjusts PIV for the quality of opponents faced, ensuring fair comparisons.

* **Formula**:  
  OSM = (Opponent HLTV Rating) / (League Average HLTV Rating)
  + HLTV Rating from your data (HLTV or HLTV 2.0).
  + League average computed across all teams in the dataset.
* **Why OSM?**: A PIV earned against tier-1 teams (e.g., NAVI) is more valuable than against tier-3, enhancing scouting accuracy.
* **Implementation**: Extract opponent HLTV from match data, divide by a precomputed league average (e.g., 1.0), apply as a multiplier.

**E. Final Player Impact Value (PIV)**

This combines all components into a single, comprehensive score.

* **Formula**:  
  PIV\_player = [(RCS\_player \* ICF) + SC\_player] \* OSM
  + RCS\_player \* ICF: Scales role performance by consistency, reflecting stable output.
    - SC\_player: Adds value from teammate synergy.
    - OSM: Adjusts for opponent difficulty.
* **Interpretation**: Higher PIV = greater impact. Example: PIV = 1.2 (strong vs. tough opponents) vs. 0.8 (average vs. weaker teams).
* **Implementation**: Compute RCS, ICF, SC, OSM separately, then plug into the formula.

**2. Team Impact Rating (TIR)**

The **TIR** assesses a team’s collective strength by summing individual PIVs and adding a teamwide synergy bonus.

**A. Aggregate Player Values (SumPIV)**

This totals the PIVs of the starting five players.

* **Formula**:  
  SumPIV = Sum(PIV\_player) for all 5 players
* **Why?**: Provides a baseline of total "star power" plus individual synergy from PIVs.
* **Implementation**: Add the five PIVs calculated above.

**B. Teamwide Synergy Layer (TSL)**

This adds a bonus for full-team coordination beyond individual pairings.

* **Formula**:  
  TSL = alpha \* CKC
  + CKC = (multi-kill chains) / (total kill\_count): Measures team momentum via chained kills (from kills\_x).
  + alpha = 0.1: Weighting factor, starting small to avoid overemphasis, tunable via regression.
* **Why CKC?**: It’s a simple, computable proxy for team synergy using your data, capturing how well the team capitalizes on kills. More complex metrics (e.g., SEI, ZCO) need positional data we lack.
* **Implementation**: Count multi-kill chains (tight kill clusters from kills\_x), divide by team total\_kill\_count, multiply by alpha.

**C. Final Team Impact Rating (TIR)**

This combines individual and team factors into one score.

* **Formula**:  
  TIR = SumPIV + TSL
  + SumPIV: Total individual impact.
  + TSL: Bonus for team unity.
* **Interpretation**: Higher TIR = stronger team. Used to predict match outcomes (e.g., Team A TIR > Team B TIR).
* **Implementation**: Add SumPIV and TSL.

**3. Example Calculation Flow**

**Spacetaker Example**

* **RCS**:
  + ODSR = 0.56, AEI = 0.9 (normalized to 0.75), TCR = 0.65.
  + w\_k = 0.33, RCS = (0.33 \* 0.56) + (0.33 \* 0.75) + (0.33 \* 0.65) = 0.185 + 0.248 + 0.215 = 0.65.
* **ICF**:
  + Per-round score = kill\_count + trade\_kills, sigma = 1.2 (std dev across rounds).
  + ICF = 1 / (1 + 1.2) = 0.45.
* **SC**:
  + UES = 0.7 (Support flashes aiding Spacetaker kills), v\_j = 1 (only one synergy metric).
  + SC = 0.7.
* **OSM**:
  + Opponent HLTV = 1.2, League Avg = 1.0, OSM = 1.2 / 1.0 = 1.2.
* **PIV**:
  + PIV = [(0.65 \* 0.45) + 0.7] \* 1.2 = (0.2925 + 0.7) \* 1.2 = 0.9925 \* 1.2 = 1.191.

**Team Example**

* **SumPIV**:
  + PIVs = [1.191, 0.95, 1.05, 0.88, 1.00], SumPIV = 5.071.
* **TSL**:
  + CKC = 0.4 (team multi-kill chains / total\_kill\_count), alpha = 0.1, TSL = 0.1 \* 0.4 = 0.04.
* **TIR**:
  + TIR = 5.071 + 0.04 = 5.111.

import pandas as pd

import numpy as np

# Sample data creation

data = {

'player': ['Player1', 'Player2', 'Player3', 'Player4', 'Player5'],

'role': ['Spacetaker', 'Lurker', 'T-Side AWP', 'IGL', 'Support'],

'kill\_count': [25, 18, 22, 15, 12],

'death\_count': [18, 20, 19, 17, 16],

'assist\_count': [5, 4, 3, 8, 10],

'first\_kill\_count': [8, 3, 7, 4, 2],

'first\_death\_count': [5, 4, 3, 3, 2],

't\_first\_kills': [8, 3, 7, 4, 2],

't\_first\_deaths': [5, 4, 3, 3, 2],

'trade\_kills': [6, 4, 3, 2, 5],

'trade\_deaths': [4, 3, 2, 3, 4],

'through\_smoke': [1, 5, 2, 0, 1],

'1v2\_won': [1, 2, 1, 1, 0],

'1v2': [2, 3, 2, 2, 1],

'1v3\_won': [0, 1, 0, 0, 0],

'1v3': [1, 2, 1, 1, 0],

'assisted\_flashes': [2, 1, 3, 4, 6],

't\_flahes\_thrown': [10, 8, 12, 15, 20],

'total\_util\_thrown': [25, 20, 30, 40, 35],

'bomb\_planted\_count': [3, 2, 4, 5, 6],

't\_rounds\_won': [10, 10, 10, 10, 10],

'he\_thrown': [5, 4, 6, 8, 7],

'infernos\_thrown': [3, 2, 5, 7, 6],

'HLTV': [1.15, 1.05, 1.10, 0.95, 0.90] # Player HLTV ratings

}

df = pd.DataFrame(data)

# Opponent and league averages (example values)

opponent\_hltv = 1.2 # Opponent team avg HLTV rating

league\_avg\_hltv = 1.0 # League avg HLTV rating

# Function to normalize metrics to 0-1 range

def normalize\_metric(series):

min\_val = series.min()

max\_val = series.max()

return (series - min\_val) / (max\_val - min\_val) if max\_val > min\_val else series

# Function to calculate Role-Core Score (RCS)

def calculate\_rcs(row):

role = row['role']

weights = {'Spacetaker': [0.33, 0.33, 0.34], 'Lurker': [0.5, 0.5],

'T-Side AWP': [0.5, 0.5], 'IGL': [0.5, 0.5], 'Support': [0.5, 0.5]} # Equal weights for MVP

if role == 'Spacetaker':

odsr = row['first\_kill\_count'] / (row['first\_kill\_count'] + row['first\_death\_count'] + 1e-5)

aei = (row['kill\_count'] - row['death\_count']) / (row['first\_kill\_count'] + row['first\_death\_count'] + 1e-5)

tcr = row['trade\_kills'] / (row['trade\_deaths'] + 1e-5)

metrics = [odsr, normalize\_metric(pd.Series([aei, 0, 1]))[0], tcr] # Normalize AEI

return sum(w \* m for w, m in zip(weights['Spacetaker'], metrics))

elif role == 'Lurker':

fsr = row['through\_smoke'] / (row['kill\_count'] + 1e-5)

ccr = row[['1v2\_won', '1v3\_won']].sum() / (row[['1v2', '1v3']].sum() + 1e-5)

return sum(w \* m for w, m in zip(weights['Lurker'], [fsr, ccr]))

elif role == 'T-Side AWP':

opsr = row['t\_first\_kills'] / (row['t\_first\_kills'] + row['t\_first\_deaths'] + 1e-5)

mkc = row['kill\_count'] / (row['t\_rounds\_won'] + 1e-5) # Proxy for multi-kills

return sum(w \* m for w, m in zip(weights['T-Side AWP'], [opsr, mkc]))

elif role == 'IGL':

opsr = row['bomb\_planted\_count'] / (row['t\_rounds\_won'] + 1e-5)

uso = (row['assisted\_flashes'] + row['he\_thrown'] + row['infernos\_thrown']) / (row['total\_util\_thrown'] + 1e-5)

return sum(w \* m for w, m in zip(weights['IGL'], [opsr, uso]))

elif role == 'Support':

fas = row['assisted\_flashes'] / (row['t\_flahes\_thrown'] + 1e-5)

use = (row['bomb\_planted\_count'] + row['assisted\_flashes']) / (row['total\_util\_thrown'] + 1e-5)

return sum(w \* m for w, m in zip(weights['Support'], [fas, use]))

return 0 # Default for unassigned roles

# Function to calculate Individual Consistency Factor (ICF)

def calculate\_icf(row):

role = row['role']

if role == 'Spacetaker':

score = row['kill\_count'] + row['trade\_kills']

elif role == 'Lurker':

score = row[['1v2\_won', '1v3\_won']].sum() + row['through\_smoke']

elif role == 'T-Side AWP':

score = row['kill\_count']

elif role == 'IGL':

score = row['total\_util\_thrown'] + row['assist\_count']

elif role == 'Support':

score = row['assisted\_flashes'] + row['bomb\_planted\_count']

else:

score = row['kill\_count'] # Default

# Simulate per-round scores (assuming total\_rounds\_won as proxy for rounds played)

rounds = row['t\_rounds\_won']

sigma = np.std([score / rounds] \* rounds) if rounds > 1 else 0 # Simplified sigma

return 1 / (1 + sigma)

# Function to calculate Synergy Contribution (SC)

def calculate\_sc(row, df\_team):

role = row['role']

if role == 'Spacetaker':

support = df\_team[df\_team['role'] == 'Support'].iloc[0]

ues = support['assisted\_flashes'] / (row['t\_first\_kills'] + 1e-5)

return ues # v\_j = 1 for simplicity

elif role == 'Lurker':

igl = df\_team[df\_team['role'] == 'IGL'].iloc[0]

iir = row['through\_smoke'] / (igl['total\_util\_thrown'] + 1e-5) if row['t\_rounds\_won'] > 0 else 0

return iir

elif role == 'T-Side AWPer' or role == 'IGL' or role == 'Support':

return 0 # Simplified: no SC for these roles in MVP

return 0

# Calculate PIV

def calculate\_piv(row, df\_team):

rcs = calculate\_rcs(row)

icf = calculate\_icf(row)

sc = calculate\_sc(row, df\_team)

osm = opponent\_hltv / league\_avg\_hltv

piv = ((rcs \* icf) + sc) \* osm

return piv

# Apply calculations to DataFrame

df['RCS'] = df.apply(calculate\_rcs, axis=1)

df['ICF'] = df.apply(calculate\_icf, axis=1)

df['SC'] = df.apply(lambda row: calculate\_sc(row, df), axis=1)

df['PIV'] = df.apply(lambda row: calculate\_piv(row, df), axis=1)

# Calculate Team Impact Rating (TIR)

sum\_piv = df['PIV'].sum()

ckc = df['kill\_count'].sum() / (df['t\_rounds\_won'] + 1e-5) # Proxy for chain kills

tsl = 0.1 \* ckc

tir = sum\_piv + tsl

# Display results

print("Player Impact Values (PIV):")

print(df[['player', 'role', 'RCS', 'ICF', 'SC', 'PIV']].to\_string(index=False))

print("\nTeam Impact Rating (TIR):")

print(f"SumPIV: {sum\_piv:.3f}")

print(f"TSL: {tsl:.3f}")

print(f"TIR: {tir:.3f}")