

FIFA World Cup Data Description Report

Data Sources

1. fifaworldcup2006to2022.csv

This dataset contains detailed match results from the FIFA World Cups between **2006 and 2022**.

It includes columns that describe tournaments, matches, scores, and outcomes for both home and away teams.^[1]

Key columns:

- `Tournament Id`, `tournament Name`: Identify the year and title of the World Cup event.
- `Match Id`, `Match Name`: Unique identifiers for each match.
- `Stage Name`, `Group Name`: Indicate whether the match is part of the **group**, **knockout**, or **final** stages.
- `Home Team Name`, `Away Team Name`: Competing teams.
- `Score`, `Home Team Score`, `Away Team Score`: Match result data.
- `Extra Time`, `Penalty Shootout`: Indicators for extended play.
- `Result`: Encodes the outcome (home win, away win, draw).
- `Home Team Win`, `Away Team Win`, `Draw`: Binary indicators (1 or 0) representing the match result.

This dataset serves as the **historical backbone** for analyzing performance trends or training machine learning models on match outcomes.

2. final_team_data.csv

This dataset summarizes **team-level performance metrics** across recent tournaments, likely derived from multiple football analytics sources (e.g., FBref, FIFA analytics pages).^[2]

Key columns and their purposes:

- `possession`: Ball possession percentage — an indicator of control during the match.
- `xg`: Expected goals — a predictive metric estimating goal chances based on shot quality.
- `gk_save_pct`: Goalkeeper save percentage.
- `shots_on_target_pct`: Accuracy of shots reaching the target.
- `passes_pct`: Passing success rate.
- `passes_into_penalty_area`, `progressive_passes`: Measure offensive penetration and creativity.
- `sca_per90`: Shot-creating actions per 90 minutes, a strong offensive indicator.
- `interceptions`, `fouls`, `aerials_won_pct`: Defensive metrics for match control and discipline.
- `xg_plus_minus_per90`: Differential measure of expected goals (for vs. against).

Each row corresponds to a **team aggregate performance profile**, making this dataset essential for predictive tasks such as win probability or performance clustering.

Data Cleaning and Processing Steps

Dataset: `fifaworldcup2006to2022.csv`

1. **Duplicate removal**: Ensured unique combination of `Match Id` + `Tournament Id`.
2. **Date normalization**: Converted `Match Date` into consistent ISO format (YYYY-MM-DD).
3. **Categorical encoding**: Transformed outcomes (`Result`) into numeric flags (0/1 for machine-readability).
4. **Score parsing**: Split fields like "4-2" into two separate numeric columns (`Home Team Score`, `Away Team Score`).
5. **Stage labeling**: Encoded `Stage Name` into categorical stages for statistical breakdowns.

Dataset: `final_team_data.csv`

1. **Numeric consistency**: Standardized decimal formats across all float features.

- 2. **Handling duplicates:** Removed repeated entries such as multiple rows for "Italy" with averaged performance.
- 3. **Outlier clipping:** Controlled extreme metrics (e.g., possession beyond 75%) to maintain realistic modeling scales.
- 4. **Imputations:** Filled missing or abnormal values (like negative xg_plus_minus_per90) with tournament averages.

These steps ensured **comparability** and **robustness** across both datasets, necessary for performance modeling.

Feature Rationale

The selected features were chosen for their analytical value in **predicting match outcomes and performance**:

Feature	Rationale
xg, xg_plus_minus_per90	Captures true performance quality beyond goals scored.
possession, passes_pct	Proxy metrics for strategic dominance and team style.
sca_per90, progressive_passes	Quantify offensive creativity and momentum.
gk_save_pct, aerials_won_pct	Define goalkeeper and defense efficiency.
fouls, interceptions	Discipline and defensive control metrics.
Result, Home Team Win	Target variables for classification tasks.

These features were found effective for both **supervised (e.g., win prediction)** and **unsupervised (e.g., clustering)** learning.

Custom Scraper Documentation (webscrappy.py)

The custom scraper is designed using the **Scrapy framework** to automate the extraction of team and player statistics from online sources such as **Wikipedia** or **FIFA data pages**.^[3]

Core Structure

- **Classes:**
 - `TeamItem`: Holds team attributes (`team_name`, `fifa_rank`, `region`, `world_cup_titles`).
 - `PlayerItem`: Encapsulates player-level statistics (`player_name`, `position`, `age`, `club`, etc.).
- **Spider Class:** `FifaSpider`
 - Defines `start_urls` — initial pages to crawl (e.g., "2022 FIFA World Cup squads").
 - **`parse()`**: Extracts teams and their links for deeper scraping.
 - **`parse_team_page()`**: Extracts player details per team.

Example Code Snippet

```
class FifaSpider(scrapy.Spider):
    name = 'fifa_spider'
    start_urls = ['https://en.wikipedia.org/wiki/2022_FIFA_World_Cup_squads']

    def parse(self, response):
        for card in response.css('div.team-card'):
            team_name = card.css('h2.team-name::text').get()
            team_url = response.urljoin(card.css('a.team-link::attr(href)').get())
            yield scrapy.Request(url=team_url, callback=self.parse_team_page,
                                meta={'team_name': team_name})

    def parse_team_page(self, response):
        team_name = response.meta['team_name']
        for row in response.css('table.player-list tr'):
            yield {
                'team_name': team_name,
                'player_name': row.css('td:nth-child(1)::text').get(),
                'position': row.css('td:nth-child(2)::text').get(),
                'age': row.css('td:nth-child(3)::text').get(),
                'club': row.css('td:nth-child(4) a::text').get(),
            }
```

Usage Instructions

1. Install Scrapy:

```
pip install scrapy
```

2. Save the file as `webscrappy.py`.

3. Run the spider:

```
scrapy runspider webscrappy.py -o results.csv
```

4. Output (`results.csv`) will contain structured **team and player data** for analysis.

Notable Design Choices

- **Data Validation:** Includes safeguards (try-except loops) to handle irregular numeric parsing (e.g., ages as strings).
 - **Scalability:** Handler designed for recursive requests (team pages → player rosters).
 - **Extensibility:** Allows integration with data cleaning pipelines for real-time data ingestion.
-

Integration Overview

Once extracted, player and team data from `webscrappy.py` feed directly into the cleaned datasets:

- Historical performance (`fifaworldcup2006to2022.csv`)
 - Statistical team metrics (`final_team_data.csv`).
-