This guide provides a comprehensive walkthrough for future researchers to set up, process, and analyze the football tracking data using this project's codebase. It details the file structure, the sequential order of operations, and the specific role of each script.

Note - if you wish to reproduce our project, please consider that in some parts of the cleaning process we had to tag and clean some parts manually, if you only wish to reproduce the experiments and the analysis you can use the final clean data + datasets we uploaded here, if you decide to do so, you can skip to phase 4 in this doc.

1. Project Setup & Data Structure

Before running any code, you must organize your raw data exactly as the scripts expect. The core structure is defined in the comments of `data_clean.py` and this is the structure the script expects.

Required Directory Hierarchy: Create a root folder and structure it as follows

```
[Project Root]
├── Data
│   ├── GPS
│   │   ├── [Season Name] (e.g., ipl2324)
│   │   │   ├── [GameName](format:YYYY-MM-DD-GameName-RawDataExtended)
│   │   │   │   ├── [Player_Name]-Entire-Session.csv (Raw GPS files)
│   │   │   │   └── ...
│   ├── OPTA
│   │   ├── [Season Name]
│   │   │   ├── [Game_Name].csv (OPTA event data)
│   ├── Lineups & Subs
│   │   ├── [Season Name]
│   │   │   ├── [Game_Name]-lineup.csv
│   │   │   ├── [Game_Name]-substitutes.csv
├── Metadata.csv (Critical: Contains game start/end timestamps)
├── [All Python Scripts]
```

In addition, the module assumes GPS data folders holds the same name as the OPTA file for that match.

Lineups and Subs files hold the same name with the addition of "-lineup" | "-subtitutes" at the end of the file name.

GPS game folder's name pattern assumed: "year-month-day-monthString day, year-RawDataExtended.csv"

Players GPS data file's name pattern assumed: "year-month-day-player_name-Entire-Session.csv"

Configuration:

- **`constants.py`**: Open this file first. It contains global definitions like `POSITIONS_MAPPING` (mapping roles like 'RB' to 'defender') and sprint speed thresholds (e.g., `ZONE_6_SPRINT_THRESHOLD` = 6.94 m/s). change anything if you want.
- `feature_extraction_from_clean_games_csv.py`: Open this file and configure the acceleration_threshold constant as you wish.
- **Paths**: You will need to update absolute paths (e.g., `base_GPS_folder_path`) in `data_clean.py`, `run_vis.py` and `run_feature_extraction.py` to match your local machine.

2. Phase 1: Data Cleaning Pipeline

Run these scripts in the following order to standardize raw data.

Step 0: Create a Metadata File

This file should contains the games names, start and end time of each half and any additional info you want to clean (e.g. players who got red cards).

Note - in the process of creating this file we had to manually tag different things, you can find the final data here

Step 1: Parse Lineups

- **File:** `parse_sofa_hunt.py`
- **Action:** Run `parse_sofa_hunt_data(csv_paths)` on the lineups and subs folder.
- **What it does:** Parses lineup and substitution files (which was scraped from SofaScore). It encodes player names to match the GPS data format (e.g., "M. Kandil" -> "RB_11") and standardizes the substitution logs.

Step 2: Clean GPS Data

- **File:** `data_clean.py`
- **Action:** Run the `clean_pipeline()` function.
- **What it does:** It iterates through every game folder in the GPS directory. Using the start/end times found in `Metadata.csv`, it trims the raw "Entire-Session" player CSVs to strictly include only match time.
- **Input:** Raw GPS CSVs + `Metadata.csv` + Lineups & Subs files
- **Output:** Overwrites raw GPS files with cleaned versions.

Step 3: Clean OPTA Data

- **File:** `clean_OPTA_data.py`
- **Action:** Use functions like `delete_unrealated_games` and `clean_OPTA_data`.
- **What it does:** Filters the event files to remove games involving other teams, keeps only "Maccabi Haifa" events (`delete_other_team_chances`), and removes static play events like penalties (`delete_static_chances`).
- **Output:** Cleaned OPTA CSV files.

3. Phase 2: Feature Extraction

Once the data is clean, you need to extract physical performance metrics.

Step 4: Extract Physical Metrics

- **File:** `run_feature_extraction.py` (executes logic from `feature_extraction_from_clean_games_csv.py`)
- **Action:** Update the `game_folder_path` and `opta_folder_path` variables, then run the script.
- **What it does:**
  - It calls `get_game_features_csv` which processes every player's GPS file.
  - It calculates specific metrics defined in `feature_extraction_from_clean_games_csv.py`, such as accelerations, decelerations, and distances.
  - It aggregates these into specified intervals (the default is 5-minute chunks).
  - It label each time chunk with the relative xG created by the team.
- **Output:** Generates a `features_[GameName].csv` file inside each game's folder.
- **Optional:** use **`expected_points_simulator.py to label base on xPts`**:
  - **Usage:** Use `simulate_expected_points` to calculate xPts (Expected Points). It uses binomial or Poisson simulations based on xG (Expected Goals) data from the OPTA files to determine the probability of winning, drawing, or losing a match.

Step 5: Create Master Datasets

- **File:** `create_full_dataset.py`
- **Action:** Run `create_half_time_dataset()`.
- **What it does:** It recursively searches for all `features_*.csv` files generated in the previous step and concatenates them into master datasets.
- **Output:** Produces `full_features_dataset.csv` and `halves_features_dataset.csv` for model training.

4. Phase 3: Visualization and Verify Cleaning Process

These files can be used for validation after the cleaning process is done.

- **`run_vis.py`** & **`projection_animation.py`**:
    - **Usage:** Run `run_vis.py` to visualize player movement. It uses `concat_frames` to prepare the data and `projection_animation` to render a 2D pitch with moving player dots. Use this to visually verify that your data cleaning (Phase 1) correctly synchronized the GPS coordinates.
    - **Action:** Change the files and directories configuration in `run_vis.py`
    - This visualization tool is based on `stadium_projection.py` script , this script contains the real world GPS coordinates of all the stadiums of the first division of the israeli football league, additionally, this script normalizes the GPS coordinates of the players and projects them to 2D football stadium coordinates.

5. Phase 4: Analysis & Model's Training

You can find the various analysis and model's training we did in the jupyter notebook folder, please refer to the README.md file for reference on what each notebook does.

We used Google Colab to write and run those notebooks, we strongly suggest you to use this platform too, but feel free to choose another one.A broad explanation of each notebook:

1. Data Exploration & Visualization

These notebooks are the first step after creating your datasets. Use them to understand the distribution of your data and detect outliers.

`datasets_features_target_histogram.ipynb`

- **Purpose:** Visualizes the distribution of features and target variables across the entire dataset.
- **Input Data:** `full_games_dataset.csv`
- **What it does:**
    - Loads the aggregated full-game dataset.
    - Generates histograms for physical metrics (distance, sprints) and target variables (xG, xPts).
    - Helps verify that the data is normally distributed or identifies skewness before modeling.

`datasets_feature_target_histogram_upper_lower.ipynb`

- **Purpose:** Compares data distributions specifically between games against "Upper Playoff" teams vs. "Lower Playoff" teams.
- **Input Data:** `upper_playoff_dataset.csv`, `lower_playoff_dataset.csv`
- **What it does:**
    - Loads two separate datasets representing opponent quality.

    - Plots comparative histograms to see if physical performance or match outcomes differ significantly based on the opponent's tier.

2. Feature Relevance & Correlation Analysis

These notebooks focus on determining *which* physical metrics actually impact match outcomes (xG or xPts).

`relevance_analysis_9_10_2025_meeting.ipynb`

- **Purpose:** A general relevance analysis, likely prepared for a specific progress meeting.
- **Input Data:** `full_games_dataset.csv`
- **Focus:** "Does player's sprints correlate to Total xG?"
- **What it does:**
    - Calculates **Pearson correlations** between physical features and Expected Goals (TotalxG).
    - Computes **Feature Importance** using Random Forest and Permutation Importance to rank which metrics (e.g., "Sprints in Zone 5") are most predictive of creating scoring chances.

`relevance_analysis_xPts.ipynb`

- **Purpose:** Similar to the above, but specifically targets **Expected Points (xPts)**.
- **Input Data:** `full_games_dataset.csv`
- **Focus:** Determining which physical efforts contribute most to winning points (Win/Draw probability).
- **What it does:**
    - Runs correlation and feature importance analysis against the `xPts` target variable.
    - Outputs tables showing P-values and importance scores to validate if running more actually leads to more points.

`Upper_Lower_playoff_teams_analysis_xPts.ipynb`

- **Purpose:** A specialized analysis splitting the relevance checks by opponent level.
- **Input Data:** `upper_playoff_dataset.csv`, `lower_playoff_dataset.csv`
- **Focus:** "Do physical metrics matter more against weak teams vs. strong teams?"
- **What it does:**
    - Separates the analysis into two contexts: Upper Playoff vs. Lower Playoff opponents.
    - Checks correlations to xPts within these specific subsets to see if the "formula for winning" changes based on the opponent.

3. Predictive Modeling

These notebooks are used to train machine learning models and evaluate their performance.

`models_analysis_full_dataset_with_shap_for_trees.ipynb`

- **Purpose:** The comprehensive modeling suite for the full-game dataset.
- **Input Data:** `full_games_dataset.csv`
- **Key Models:** Linear Regression, Lasso, Ridge, Random Forest.
- **What it does:**
    - Trains multiple regressors to predict match outcomes.
    - **SHAP Analysis:** Uses SHAP (SHapley Additive explanations) values to explain *why* the tree-based models made specific predictions, offering a deep dive into feature contribution (e.g., "High distance covered increased the predicted xG by 0.2").

`models_analysis_xPts.ipynb`

- **Purpose:** Evaluation of models specifically trained to predict **xPts**.
- **Input Data:** `full_games_dataset.csv`
- **What it does:**
    - Focuses on optimizing models for the Expected Points metric.
    - Reports performance metrics like **R^2 (R-squared)** and **MSE (Mean Squared Error)** to quantify how well the physical data explains the variation in points won.

`halves_2_seasons_models.ipynb`

- **Purpose:** Training models on **Half-Time** intervals rather than full games.
- **Input Data:** `halves_features_dataset.csv`
- **What it does:**
    - Treats each half as an independent data point. This doubles the sample size and allows for analysis of performance consistency (e.g., "Does performance in the 1st half predict the 2nd half result?").