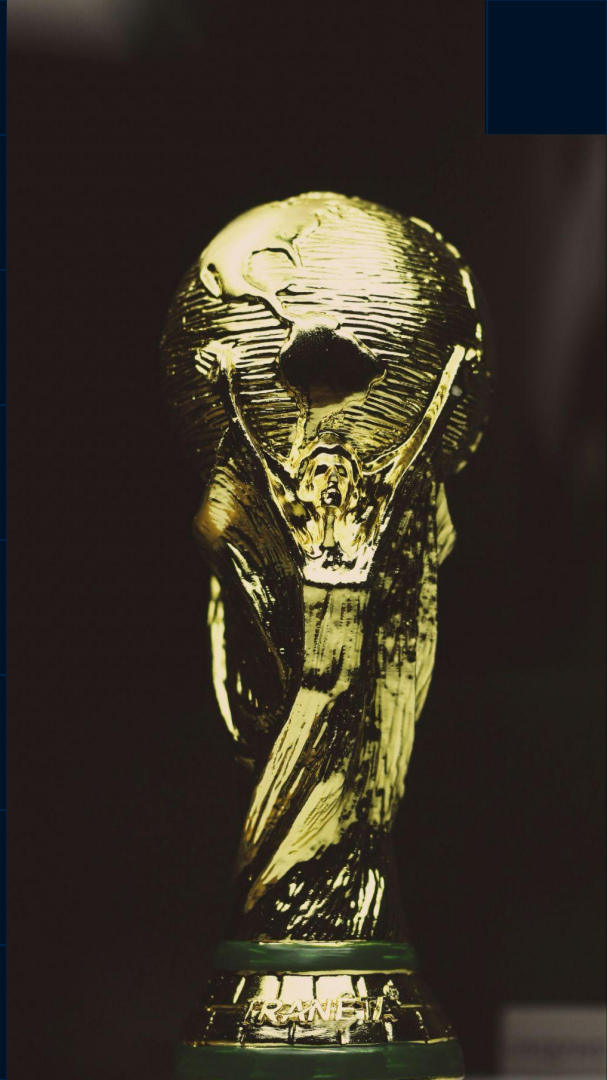
Predicting the 2026 World Cup

Akshat, Jacob, Kevin, Richard

Introduction

This project leverages data analytics and machine learning to predict match winners, analyze team and player performance, and enable interactive forecasting using historical data, strategies, and external factors.



Data Overview

Includes data for both Men and Women World Cups, from 1930-2022

Match Data

Historical records of match outcomes, scores, timings, and venue details from multiple World Cup tournaments

Player Data

Player statistics covering appearances, goals, bookings, and performance metrics to enable in-depth individual analysis.

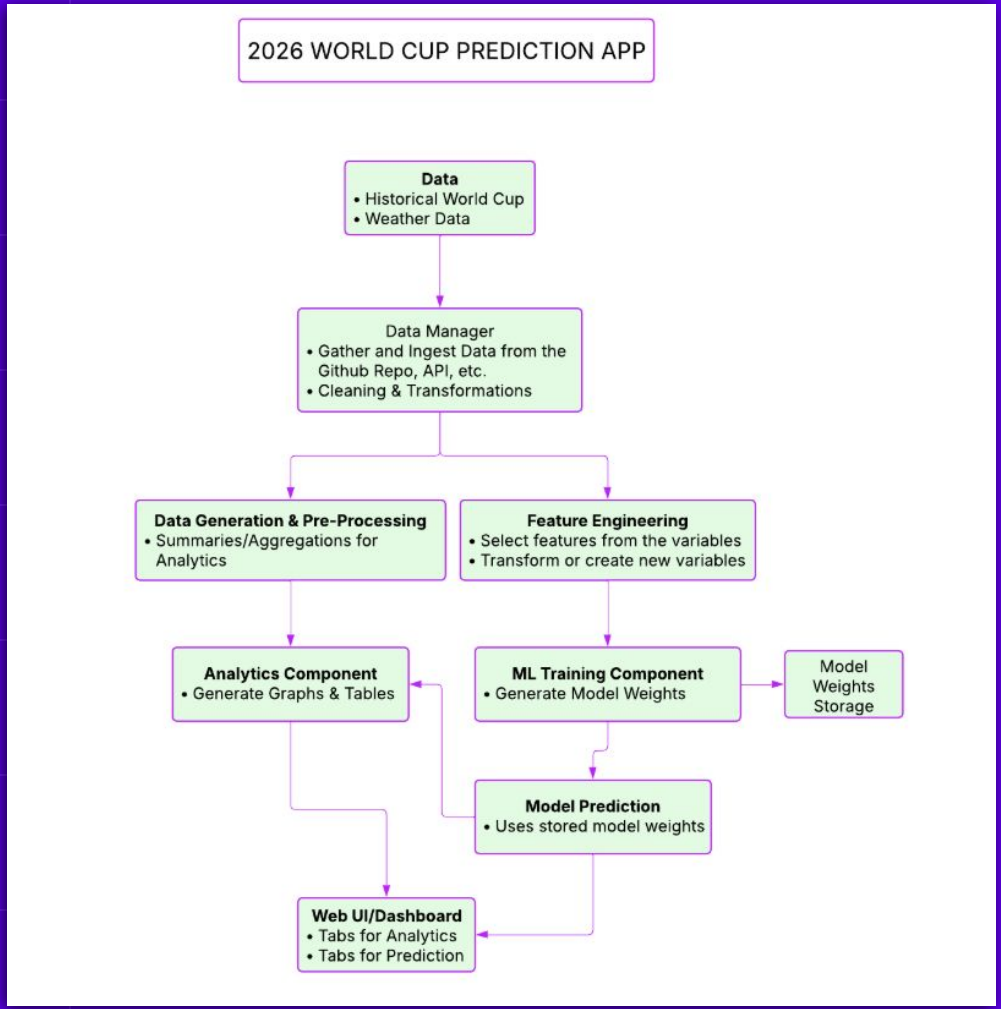
Weather Data

Average temperature of the city in year when the game took place.

FIFA Rankings

Fifa rankings to show historical and current team performance and competitive strength.

Architecture Diagram



Web App Overview

Homepage

- Overview of our World Cup 2026 Prediction App
- Features an interactive map of host cities

Analytics Page

- In-depth visualizations and leaderboards
- Detailed insights into historical player and team performance.

Predictions Page

- Uses ML to predict match outcomes
- Uses user inputs, such as weather and FIFA rankings to predict match outcome

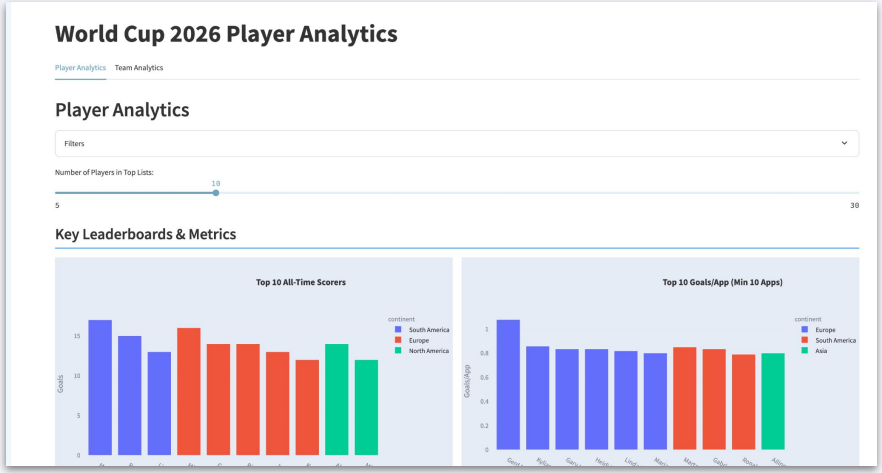
Feature Overview

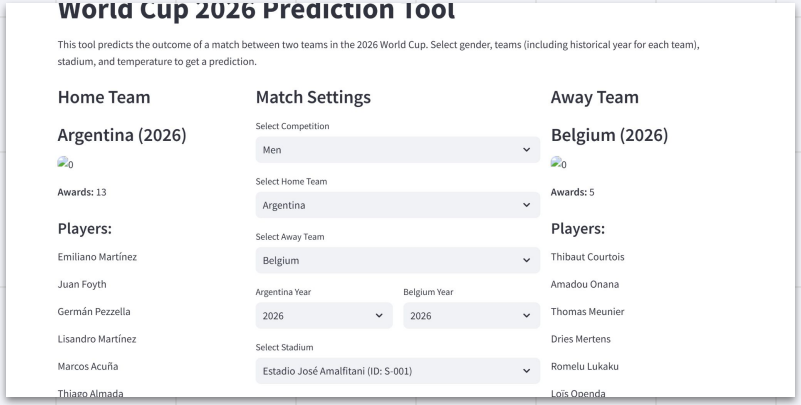
Player Analytics

- Player leaderboards on key metrics such as appearances, top goal scorers
- Compares two players based on their historical performances

Team Analytics

- View Overall Team Figures
- View information on one team, or compare
- Visualize historical team trends and key tournament trends





Match Predictions

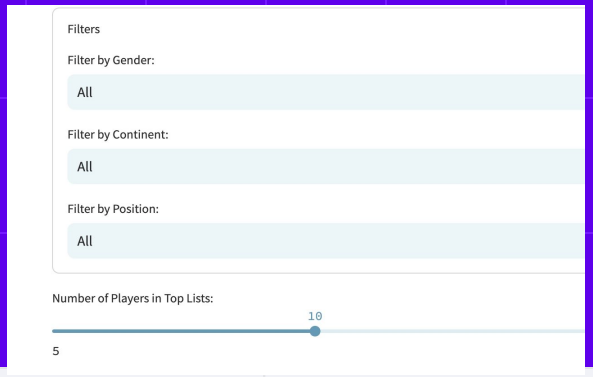
- Analyzes game events and weather conditions
- Predicts match outcomes

Analytics Workflow

```
Codeium: Refactor | Explain | X
def _prepare_player_base(players_df):
    """
    Creates the initial 'player_stats' DataFrame with core columns like full_name, birth_dat
    """
    # Ensure 'given_name' and 'family_name' columns exist before fillna/apply:
    if "given_name" not in players_df.columns:
        players_df["given_name"] = ""
    else:
        players_df["given_name"] = players_df["given_name"].fillna("").apply(_fix_name)

    if "family_name" not in players_df.columns:
        players_df["family_name"] = ""
    else:
        players_df["family_name"] = players_df["family_name"].fillna("").apply(_fix_name)

    players_df["full_name"] = (
        players_df["given_name"] + " " + players_df["family_name"]
        ).str.strip()
    players_df.loc[players_df["full_name"] == "", "full_name"] = "Unknown"
```



Data

Combine data sets available, enrich to create new statistics

User Inputs

Built ability for user to filter based on Country, Gender, and Year of Interest

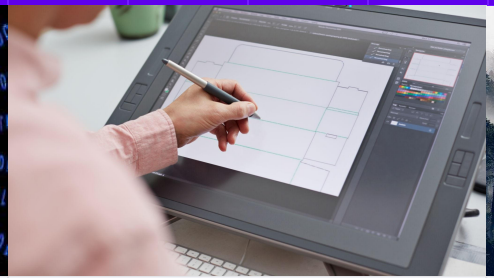
Visuals

On demand visuals created based on users desired filters

Match Predictions Workflow

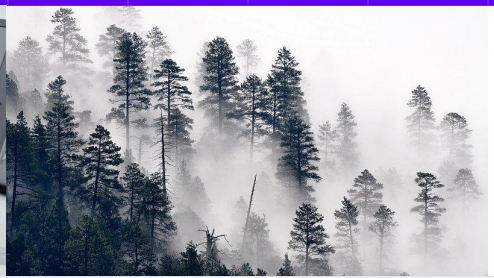
Data

Utilize past performances, weather, and player data to create training data set



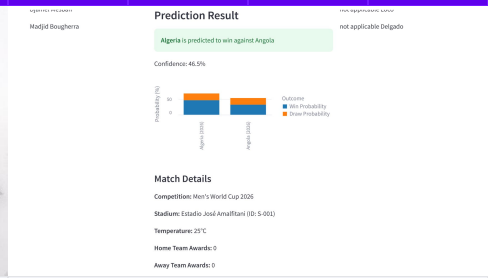
User Inputs

Allow users to select teams, eras, stadiums, as well as the weather



Model

Our Random Forest model utilizes all the data to identify the outcome of the match



UI

Predicted winner alongside confidence stats are displayed to the user

Lessons Learned

- Benefits of Modularity
  - Made it easy to split among group members
- **Learning** before *implementing*
  - Wasting time on a quick solution before understanding options packages supply
- Clear project goals
  - Setting clear low, mid, high level goals helps the project stay focused
- Properly setting Github repo with CI/CD pipeline
  - various files to check incoming merges is time consuming, but nice to have
- Testing and pylint are a challenge if not considered along the way

Future Work

- Add in-game events as model outputs
- World Cup Simulation using prediction feature
- More to come

Demo!