**Project Overview**

* **Objective**: Build a prediction model for ODI cricket matches and a simple UI displaying static match-related data (venue, weather, average scores by format).
* **Approach**: Use no-code platforms to simulate a team of AI agents, each with a specific role, and a final manager for validation.
* **Tools**: Google Sheets, Obviously AI (or BigML), Google Sites (or Bubble), and optionally Zapier for automation.
* **Team Structure**: Think of each task as an “AI agent” you’ll oversee, with a “final manager” (you or a tool feature) ensuring data quality.

**Step 1: Assemble Your AI Team**

We’ll simulate a team of AI agents using no-code tools, each assigned a specific task, just like people working on a project. Here’s your lineup:

1. **Data Collection Agent**: Gathers ODI match data.
2. **Data Cleaning Agent**: Cleans and organizes the data.
3. **Feature Engineering Agent**: Adds key predictive features.
4. **Prediction Model Agent**: Builds and trains the prediction model.
5. **UI Agent**: Creates a simple interface with static data.
6. **Final Manager**: Validates all data before it’s used in the model or displayed.

You’ll manage these “agents” using accessible tools—no coding required!

**Step 2: Data Collection (Data Collection Agent)**

* **Task**: Collect historical ODI match data.
* **How to Do It**:
  + **Option 1**: Download a free dataset from Kaggle, such as ["ODI Men's Cricket Match Data (2002-2023)"](https://www.kaggle.com/datasets/zeesolution/odi-mens-cricket-match-data-2002-2023). It includes teams, venues, winners, and more.
  + **Option 2**: Use **CricAPI** (a free cricket API) to fetch data. Since you’re a non-coder, pair it with **Zapier** to automatically save the data to Google Sheets without writing code.
* **Output**: A dataset with past ODI matches (e.g., teams, venues, toss outcomes, winners).
* **Why**: This data trains your prediction model and provides stats for the UI.

**Step 3: Data Cleaning (Data Cleaning Agent)**

* **Task**: Prepare the data by removing errors and inconsistencies.
* **How to Do It**:
  + Open your dataset in **Google Sheets** or **Excel**.
  + Check for:
    - Duplicates (remove them using the “Remove Duplicates” feature).
    - Missing values (fill with zeros, averages, or skip those rows).
    - Formatting issues (ensure dates, team names, etc., are consistent).
* **Output**: A clean, organized spreadsheet ready for analysis.
* **Why**: Clean data ensures your model makes accurate predictions.

**Step 4: Feature Engineering (Feature Engineering Agent)**

* **Task**: Enhance the data with features that improve predictions.
* **How to Do It**:
  + In Google Sheets, add columns for key factors like:
    - **Recent Form**: Calculate each team’s win percentage over their last 5 matches (e.g., use COUNTIF to count wins and divide by 5).
    - **Head-to-Head Record**: Compute how often Team A beats Team B historically.
    - **Venue Performance**: Track each team’s win rate at the venue.
  + Use simple formulas or filters to calculate these.
* **Output**: A dataset with added columns for predictive features.
* **Why**: These features (e.g., team form, venue stats) make your model smarter.

**Step 5: Build the Prediction Model (Prediction Model Agent)**

* **Task**: Create and train a model to predict ODI match winners.
* **How to Do It**:
  + Use a no-code platform like **Obviously AI** or **BigML**.
  + Steps:
    1. Upload your cleaned dataset.
    2. Select “winner” as the target variable.
    3. Include features like teams, venue, toss winner, and your engineered features (e.g., recent form).
    4. Let the platform build and train the model automatically.
* **Output**: A trained model ready to predict match outcomes.
* **Why**: This is the heart of your project, turning data into predictions.

**Step 6: Data Validation (Final Manager)**

* **Task**: Check and validate all data before it’s fed into the model or displayed.
* **How to Do It**:
  + Use the built-in validation tools in Obviously AI or BigML to flag inconsistencies (e.g., missing values or outliers).
  + Manually review key data points (e.g., recent match results, venue stats) for accuracy.
  + Before predicting a new match, ensure the input data (teams, venue, etc.) is correct.
* **Output**: Reliable, validated data for predictions and the UI.
* **Why**: This step ensures your predictions and displayed info are trustworthy.

**Step 7: Create the UI (UI Agent)**

* **Task**: Build a simple interface displaying static data: venue, weather, and average scores (day or day-night matches).
* **How to Do It**:
  + Use **Google Sites** (simplest) or **Bubble** (more interactive) to create a web page.
  + **Design**:
    - **Venue**: A text field showing the stadium name (e.g., “Lord’s”).
    - **Weather**: A field for manual input of the match day forecast (e.g., “Sunny, 25°C”).
    - **Average Scores**: Precomputed averages by venue and match type.
  + **For Average Scores**:
    - In Google Sheets, filter your dataset by venue and match type (day or day-night).
    - Calculate averages (e.g., AVERAGEIF for scores at Lord’s in day matches).
    - List them in your UI (e.g., “Lord’s Day Match Avg: 250, Day-Night Avg: 270”).
  + Update weather manually before each match since it’s static.
* **Output**: A basic webpage with venue, weather, and format-specific average scores.
* **Why**: This provides context for predictions and meets your static data goal.

**Step 8: Predicting New Matches**

* **Task**: Use the model to predict upcoming ODI outcomes.
* **How to Do It**:
  + For a new match, gather details (teams, venue, toss winner, etc.).
  + Input these into Obviously AI’s prediction interface (usually a form or “Predict” tab).
  + Get the predicted winner.
  + Optionally, add a “Predicted Winner” field in your UI and manually update it.
* **Output**: A prediction like “Team A has a 65% chance to win.”
* **Why**: This lets you apply your model to real-world matches.

**Step 9: Project Workflow**

Here’s how your “team” works together:

1. **Data Collection Agent** gathers raw ODI data.
2. **Data Cleaning Agent** processes it into a usable format.
3. **Feature Engineering Agent** adds predictive features.
4. **Final Manager** validates the data.
5. **Prediction Model Agent** trains the model and makes predictions.
6. **UI Agent** displays the static data (venue, weather, average scores).
7. For each new match:
   * Input details into the model.
   * Update the UI with static data and (optionally) the prediction.

You’re the project leader, coordinating these “agents” using no-code tools.

**Additional Tips**

* **Automation (Optional)**: Use **Zapier** to automate fetching new match data from CricAPI to Google Sheets, reducing manual work.
* **UI Options**:
  + **Google Sites**: Easiest—create a static page with text fields you update manually.
  + **Bubble**: More advanced—add dropdowns for venues and display precomputed averages (requires some learning).
* **Weather**: Check forecasts online (e.g., BBC Weather) and manually enter them. A weather API is possible with Bubble but optional.
* **Scaling Up**: Start simple, then explore integrating the model and UI if you get comfortable with tools.

**Why This Works for You**

* **No Coding**: Every step uses beginner-friendly tools like Google Sheets, Obviously AI, and Google Sites.
* **Team Structure**: The “AI agents” break the project into manageable tasks, with the “final manager” ensuring quality—just like a real team.
* **Static UI**: Venue, weather, and average scores are displayed as requested, tailored to day or day-night formats.
* **Flexibility**: You can predict new matches and update the UI manually, keeping it simple.

This plan gives you a fully functional ODI prediction system and UI, all without coding. Let me know if you’d like help with any specific step—like downloading data, calculating averages, or setting up the UI!