Project Report: IPL Deep Dive Analytics & Winner Predictor

An End-to-End Data Science Application for Cricket Analytics

Author: Amogh Pai
Date: October 6, 2025
Project Status: Completed & Deployed

Abstract

The **IPL Deep Dive Analytics & Winner Predictor** is a comprehensive data science project that provides **in-depth pre-match analysis and predictive insights** for the Indian Premier League (IPL).

This project converts **over 1,170 raw, nested JSON files** into a **structured, interactive application** that enables:

- Visualization of historical match and player performance data
- Analysis of team statistics and head-to-head comparisons
- Prediction of match outcomes using **machine learning algorithms**

The report covers the **entire project lifecycle**, including:

1. **Data Engineering & Pre-processing** – Cleaning, flattening, and structuring complex datasets for analysis
2. **Feature Engineering & Machine Learning** – Creating predictive features and training an **XGBoost model** to forecast match winners
3. **Interactive Web Application** – Developing a **Streamlit dashboard** for real-time analytics and visualizations
4. **Deployment** – Hosting the application online for public access, optimized for performance and reliability

Data Source: Raw IPL data was sourced from **Cricsheet**, ensuring authenticity and completeness of historical match records

Table of Contents

1. Introduction

1.1 Project Overview

The **IPL Deep Dive Analytics & Winner Predictor** is a full-stack data science application designed to act as an **expert pre-match analyst** for the Indian Premier League (IPL). It transforms **historical, ball-by-ball cricket data** into actionable insights and predictive analytics, presented through an **interactive web dashboard**.

1.2 Problem Statement

The project began with a dataset of **over 1,170 individual, nested JSON files**. While detailed, this format was unsuitable for performant analysis or web deployment due to:

- **High I/O Overhead**: Loading and parsing thousands of files is extremely slow.
- **Complex Structure**: Nested data is difficult to query or analyze directly.
- **Inconsistent Data**: Old team names and defunct franchises compromised data integrity.

Core Challenge: Build a robust data pipeline to process, clean, and structure the data efficiently for analysis and deployment.

1.3 Objectives

- **Process Raw Data**: Ingest, parse, and structure 1,170+ JSON files into a clean tabular format.
- **Standardize Data**: Merge renamed franchises and remove defunct teams for consistency.
- **Build Predictive Model**: Train a machine learning model to predict match winners.
- **Develop Interactive UI**: Build a Streamlit dashboard for matchup analysis and predictions.

- **Deploy Online**: Host the application for public access.

1.4 Technology Stack

- **Language**: Python
- **Data Manipulation**: Pandas, NumPy
- **Machine Learning**: Scikit-learn, XGBoost
- **Web Framework**: Streamlit
- **Data Visualization**: Plotly Express
- **Deployment**: Streamlit Community Cloud, GitHub

2. Data Engineering & Pre-processing

2.1 Data Source & Initial Challenges

Data was sourced from **Cricsheet**, providing ball-by-ball details for every IPL match in separate JSON files.

Challenges:

- **Volume**: 1,170+ files with nested structures.
- **Nested Objects**: Each JSON file had multiple levels, making direct analysis difficult.
- { "info": { "teams": [...] }, "innings": [...] }
- **Server Limitation**: Processing all nested JSONs directly on a low-resource environment would crash the application.

2.2 The Pre-processing Pipeline

A Python script (create_csv_files.py) was built to execute a multi-stage pipeline:

1. **Data Ingestion**: Unzipped ipl_json.zip and recursively located all JSON files.
2. **Data Structuring (Flattening)**:
    - **Matches DataFrame**: One row per match with summary info (teams, venue, winner).
    - **Deliveries DataFrame**: One row per ball with batter, bowler, and runs scored.

2.3 Data Cleaning & Standardization

- **Merging Renamed Teams**: Team names were standardized for consistency.

Old Name	New Name
Delhi Daredevils	Delhi Capitals
Kings XI Punjab	Punjab Kings
Royal Challengers Bangalore	Royal Challengers Bengaluru

- **Removing Defunct Teams**: Filtered out franchises like Deccan Chargers and Gujarat Lions.

2.4 Final Data Output: CSV Transformation

Two clean CSV files were generated: all_matches.csv and all_deliveries.csv. This reduced the dataset from **1,170+ files to just 2 files**, a critical optimization for deployment.

3. Machine Learning Model Development

3.1 Objective: Match Winner Prediction

The goal of this stage was to **predict the winning team** for a match based on historical data. This is a **binary classification** problem (team1 wins = 1, loses = 0).

3.2 Feature Engineering

- **Encoding Categorical Features**: LabelEncoder was used for team names and venues.
- **Advanced Feature – Team Form**: Calculated the win percentage over the last 5 matches. .shift(1) was applied to prevent data leakage, ensuring the model only used prior match data.

 # Calculate rolling win percentage

team_matches['form_win_pct'] =
team_matches.groupby('team')['is_win'].rolling(5).mean()

 # Shift to prevent data leakage

team_matches['form_win_pct_prior'] =
team_matches.groupby('team')['form_win_pct'].shift(1)

3.3 Model Selection: XGBoost

The **XGBoost** algorithm was chosen for its **high performance on tabular data**. It builds sequential decision trees where each new tree corrects errors of the previous one.

3.4 Model Training, Evaluation & Persistence

- **Train-Test Split**: 80% training, 20% testing.
- **Training**: XGBClassifier was trained on the engineered features.
- **Evaluation**: Achieved **52.61% accuracy** on unseen test data, providing a slight predictive edge over random chance (50%) in a highly variable sport.
- **Persistence**: Model and encoders were saved using joblib for instant loading in the web app:

joblib.dump(model, 'ipl_winner_model.pkl')

4. The Streamlit Web Application

4.1 Architecture and Performance Optimization

App performance was optimized using **Streamlit caching mechanisms**:

- @st.cache_data → Loads large CSV files only once.
- @st.cache_resource → Loads the machine learning model and encoders only once.

4.2 User Interface and Key Features

- **Sidebar**: Allows users to select competing teams, venue, and toss results.
- **Main Panel**:
    - **Head-to-Head Insights**: Shows total matches played, win counts, and toss impact.
    - **Match Prediction**: Encodes user inputs, feeds them to the model, and displays win probabilities with **pie charts** and textual justification.

4.3 Visual Styling with CSS

Custom CSS was applied using st.markdown() to achieve a **professional, dark-themed look**, styling player cards and metrics.

5. Deployment

5.1 The Deployment Challenge: Server Limitations

Initial deployment failed because the free-tier server exceeded its **Inotify instance limit** when monitoring all 1,170+ JSON files.

5.2 The Professional Solution: Pre-processing

All heavy data processing was performed **offline**, so the deployed app now reads **only two small CSV files**, ensuring fast, reliable performance.

5.3 Final Deployment Workflow

1. **Organize Repository**: GitHub repository included the app script, CSV files, model files, and requirements.txt.

2. **Connect to Cloud**: Repository connected to Streamlit Community Cloud.
3. **Launch**: Streamlit installed dependencies automatically and launched the application for **public access**.

6. Conclusion & Future Scope

6.1 Project Summary

This project successfully demonstrates a **complete end-to-end data science workflow**:

- Processed a **large, complex, and messy dataset** efficiently.
- Built and evaluated a predictive **XGBoost model** for match winner prediction.
- Developed an interactive and visually engaging **Streamlit dashboard**.
- Overcame deployment challenges to host the application for **public access**.

This project serves as a strong portfolio piece showcasing **proficiency in data engineering, machine learning, and application deployment**.

6.2 Future Enhancements

- **Individual Player Metrics**: Include player strike rates, economy rates, and recent form.
- **Player vs. Player Analysis**: Analyze historical matchups between key batsmen and bowlers.
- **Score Prediction Model**: Develop a regression model to predict first-innings scores based on teams, venue, and lineups.

Thank You!

I sincerely thank everyone who contributed to this project and supported its development. This includes **mentors, peers, and resources** that helped in data collection, model development, and deployment.

The completion of this project demonstrates not only technical skills but also **dedication, persistence, and problem-solving ability** in building an end-to-end data science solution.