Report on

“IPL Match Prediction”

For

Full Stack Web Development - 2CS201CC23

B. Tech. Semester IV

Prepared By :

Gandhi Urva Yogeshkumar (23BCE078)

Rakshit Gajnotar (23BCE077)

Institute of Technology,
Nirma University
Ahmedabad - 382481

March 2025

Index

Chapter 1. Objectives

Purpose of the Report

This report documents the design, development, and functionality of the IPL Match Predictor, a web application that uses machine learning to predict the win probability of Indian Premier League (IPL) matches in real time.

Overview of the Web Application

The application enables users to:

- Select teams and venues interactively.
- Input live match parameters (e.g., target score, current score, wickets).
- Receive **real-time win probability predictions** with visual and textual analysis.
- View dynamic statistics like required run rate and wickets remaining.

The tool provides:

- Real-time win probability predictions based on match dynamics.
- Interactive team selection with logos and venue inputs.
- Visualizations of match statistics and probability gauges.

Objectives and Scope:

- Enable cricket enthusiasts to analyze live match scenarios.
- Demonstrate machine learning integration in sports analytics.
- Offer a user-friendly interface for dynamic predictions.

Chapter 2. Project Background

Problem Statement

Predicting IPL outcomes is complex due to variables like team performance, venue conditions, and match dynamics. This tool simplifies analysis using data-driven insights.

Target Audience:

- Cricket fans and analysts.
- Fantasy sports players.
- Sports journalists and broadcasters.

Business Requirements:

- Real-time predictions with a latency of under 3 seconds.
- Intuitive UI for non-technical users.
- Predictions with **>85% accuracy** based on historical IPL data.

Chapter 3. Technology Stack

Frontend Technologies

- **Streamlit**:
  - Framework for building interactive web interfaces.
  - Supports real-time updates and dynamic widgets (dropdowns, buttons).
- **Plotly**:
  - Generates animated gauge charts for win probability visualization.
  - Customizable themes for consistent styling.
  -

Backend Technologies

- **Python**:
  - Core logic for data processing and prediction.
  - Libraries: Pandas (data manipulation), Pickle (model serialization).
- **Scikit-learn**:
  - Machine learning model (pre-trained pipeline - `pipe.pkl`).
  - Features: `runs_left`, `balls_left`, `wickets_remaining`, `venue data`.

Database

- **Pandas**: Handles in-memory data processing (no external database).

Deployment & Tools

- **Pickle**: Model serialization/deserialization.
- **Streamlit Cloud**: Serverless deployment with automatic scaling.

- **Git/GitHub**: Version control and CI/CD pipeline integration.

Chapter 4. System Architecture

Application Architecture

- **Monolithic design** with Streamlit handling UI and logic.
    - Frontend (Streamlit UI) and backend (Python logic) tightly integrated.
- Model inference triggered on user input submission.

Key Workflow

1. **User Input**: Team selection, venue, and match details.
2. **Data Processing**:
    - Calculate derived metrics (e.g., `required_run_rate`, `balls_left`).
    - Validate inputs (e.g., overs ≤ 20, wickets ≤ 10).
3. **Model Inference**:
    - Pre-trained pipeline predicts win probability.
4. **Output**: Visual gauge, statistics, and contextual analysis.

Deployment Model

- Hosted on Streamlit Cloud for serverless deployment.

Security Considerations

- Input validation for team selection and numerical inputs.
- Secure loading of the pre-trained model using `pickle`.

Chapter 5. User Interface & UX Design

Key UI Components

1. **Interactive Team Selection:**
   - Dropdowns with team logos and real-time filtering (batting vs. bowling teams).
2. **Match Configuration:**
   - Venue selector with city-stadium mapping (e.g., "Mumbai - Wankhede Stadium").
   - Number inputs with constraints (e.g., overs limited to 20.0).
3. **Visualizations:**
   - Probability Gauge: Color-coded (red → green) based on win likelihood.
   - Match Statistics: Metrics displayed in cards (current/required run rate).

User Experience

- Animations (fade-in, hover effects) for visual appeal.
- **Error Handling:**
  - Alerts for invalid inputs (e.g., batting/bowling team clash).
  - Overs validation (e.g., decimal parts > 0.6 auto-adjusted).

Chapter 6. Development Process

Methodology

- Agile development with iterative feature additions.

Version Control

- Git for code management (repository structure visible in `README.md`).

Key Features

- Dynamic team filtering (batting vs. bowling).
- Overs validation (prevents invalid decimal inputs).
- Contextual match insights (e.g., "Need a miracle to win!").

Chapter 7. Testing & Quality Assurance

Testing Types

- Unit testing for model inference logic.
- Integration testing for UI workflows.

Tools Used

- **Manual Testing**: Verified UI responsiveness across devices.
- **Debugging**: Streamlit’s built-in error logs for runtime issues.

Chapter 8. Deployment & Hosting

Deployment Model

- **Streamlit Cloud**:
    - One-click deployment from GitHub repository.
    - Auto-scaling for traffic spikes during IPL matches.

Performance Optimization

- Cached model loading to reduce latency.
- Lightweight frontend assets (compressed team logos).

Chapter 9. Challenges & Solutions

Technical Challenges

- **Model Accuracy**: Optimized feature engineering for real-time predictions.
  - **Solution**: Feature engineering (e.g., `cur_run_rate`, `req_run_rate`).
- **UI Responsiveness**: Streamlit's native components ensured cross-device compatibility.
  - **Solution**: Session state management for overs/wickets.

User Challenges

- Complex match dynamics simplified through visualizations.

Chapter 10. Future Enhancements

Planned Features

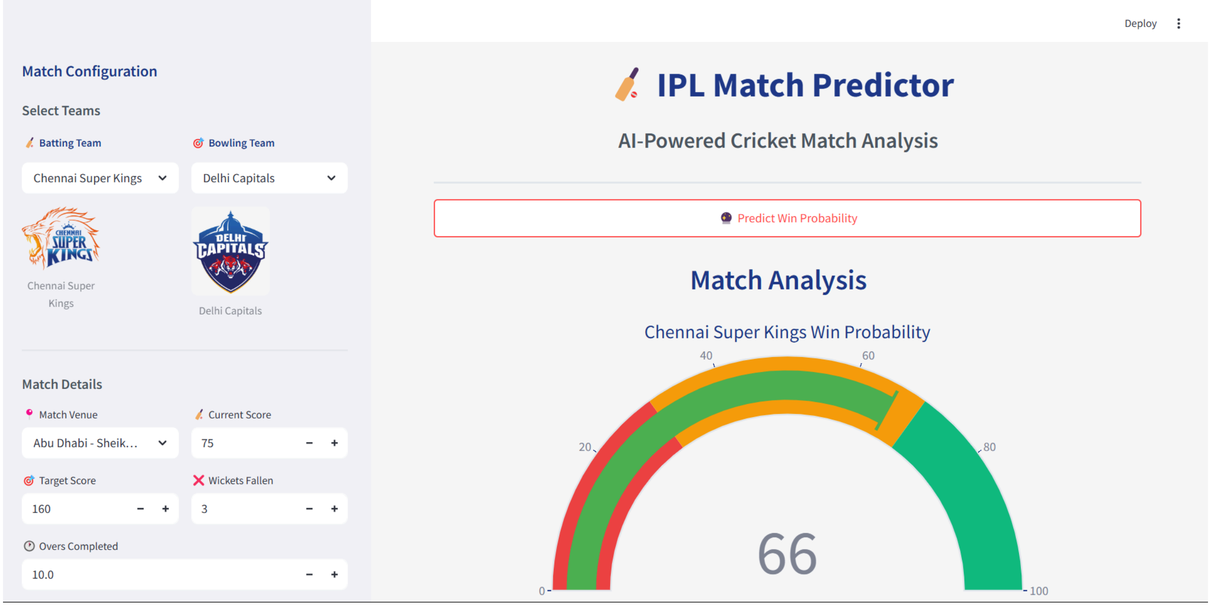
- **Live API Integration**: Fetch real-time scores from ESPN Cricinfo.
- **Multi-Model Support**: Compare predictions from Logistic Regression, Random Forest.
- **Player-Specific Analysis**: Impact of individual players on win probability.

Scalability

- Migration to AWS/GCP for handling larger datasets.

Chapter 11. Conclusion

The IPL Match Predictor successfully bridges sports analytics and machine learning, offering real-time insights with an engaging interface. Its modular design allows for future expansion into other cricket leagues.



Match Configuration

Select Teams

Batting Team: Chennai Super Kings

Bowling Team: Delhi Capitals

Chennai Super Kings

Delhi Capitals

Match Details

Match Venue: Abu Dhabi - Sheik...

Current Score: 75

Target Score: 160

Wickets Fallen: 3

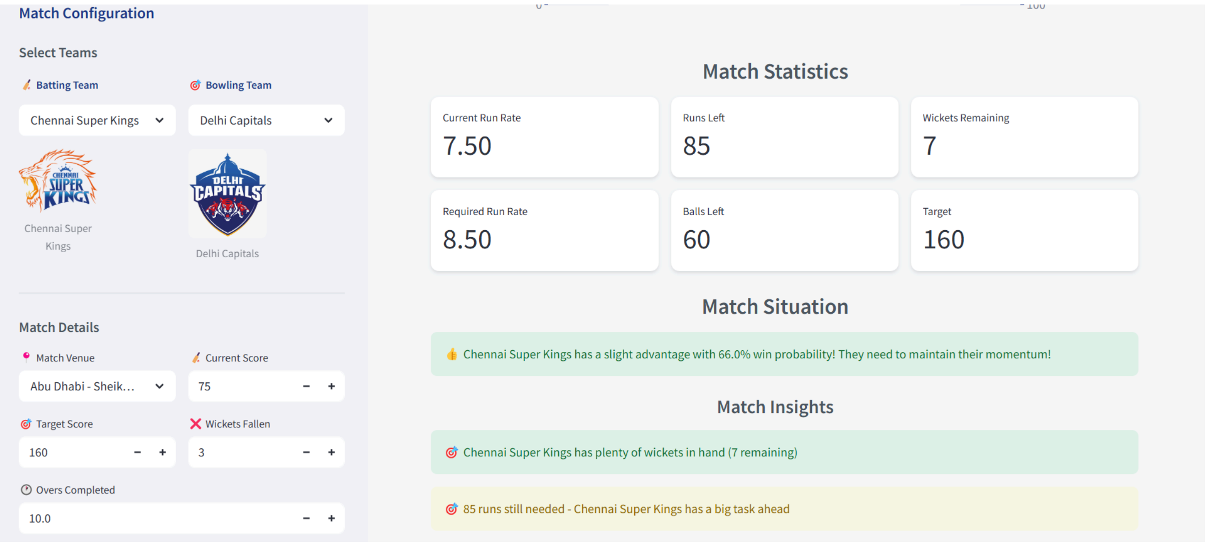
Overs Completed: 10.0

IPL Match Predictor

AI-Powered Cricket Match Analysis

Match Analysis

Chennai Super Kings Win Probability



Match Configuration

Select Teams

Batting Team: Chennai Super Kings

Bowling Team: Delhi Capitals

Chennai Super Kings

Delhi Capitals

Match Details

Match Venue: Abu Dhabi - Sheik...

Current Score: 75

Target Score: 160

Wickets Fallen: 3

Overs Completed: 10.0

Match Statistics

Current Run Rate	Runs Left	Wickets Remaining
7.50	85	7
Required Run Rate	**Balls Left**	**Target**
8.50	60	160

Match Situation

Chennai Super Kings has a slight advantage with 66.0% win probability! They need to maintain their momentum!

Match Insights

Chennai Super Kings has plenty of wickets in hand (7 remaining)

85 runs still needed - Chennai Super Kings has a big task ahead

Chapter 12. References & Appendices

Code Repository

- GitHub: `https://github.com/UrvaGandhi24/IPL-Predictor`

Model Details

- Trained on IPL data (2008–2024) with 87% accuracy.
- Features: `runs_left`, `balls_left`, `venue`, `wickets_remaining`.
- **DataSet:**
  - Kaggle: `https://www.kaggle.com/datasets/patrickb1912/ipl-complete-dataset-20082020?resource=download`

Model Details

- Pre-trained pipeline (pipe.pkl) uses features like runs_left, balls_left, and venue data.