IPL MATCH WINNER PREDICTOR USING MACHINE LEARNING

Mrs. Divya M,
Department of CSE
Rajalakshmi Engineering College
Chennai, India
divya.m@rajalakshmi.edu.in

Sai Ruthwik,
Department of CSE
Rajalakshmi Engineering College
Chennai, India
220701237@rajalakshmi.edu.in

ABSTRACT - The Indian Premier League (IPL) is one of the most unpredictable cricket leagues due to its dynamic team compositions and performance variations. In this study, we propose a machine learning-based approach to predict the winner of IPL matches using historical match data. We utilize a Random Forest classifier trained on the 2025 IPL dataset, which includes features such as toss winner, team compositions, venue, and match statistics. Our model is integrated into a web-based application using ReactJS and Flask, providing real-time predictions to users. This paper details the methodology, system architecture, evaluation metrics, and results of the implemented model, emphasizing its effectiveness and scope for future enhancements.

KEYWORDS - IPL, machine learning, match winner prediction, Random Forest, data analytics, web application.

I. INTRODUCTION

Cricket is a sport driven by a multitude of statistical and situational variables. Among various cricket formats, the Indian Premier League (IPL) has emerged as a fast-paced, data-intensive format where match outcomes depend on diverse elements ranging from toss results to venue characteristics and player form. In this era of digital transformation, data science and machine learning have found a growing presence in sports analytics, influencing areas such as team strategy, player performance tracking, and game outcome predictions.

Given the unpredictability of T20 matches and the large fanbase that actively engages with match predictions, this study proposes a machine learning-based approach to forecast the winning team of an IPL match based on pre-match features. We leverage a Random Forest classifier trained on IPL 2025 data, focusing on optimizing model accuracy and integrating it into a real-time web application for public use. This study explores the entire ML pipeline—from dataset acquisition and preprocessing to model training, validation, and deployment—while also analyzing the significance of key features influencing outcomes.

II. LITERATURE REVIEW

Machine learning has been increasingly applied in sports analytics for tasks such as player performance prediction, match outcome forecasting, and injury prevention. Several research papers have explored the application of algorithms like Logistic Regression, Naive Bayes, Support Vector Machines, and Decision

Trees to the domain of cricket. For instance, in "IPL Match Winner Prediction Using Machine Learning Algorithms," Sharma et al. (2018) utilized Logistic Regression and SVM models on historical IPL data to predict match outcomes. Their work demonstrated moderate accuracy but struggled with overfitting and handling categorical features effectively.

Another notable study by Ahmed et al. (2019) applied a Naive Bayes classifier on IPL datasets and reported that the model's predictive power was limited by its naive assumptions of feature independence. In contrast, ensemble models such as Random Forest and Gradient Boosting have shown superior performance in various classification tasks due to their ability to combine the decisions of multiple weak learners and reduce overfitting.

In sports like football and basketball, ensemble models have already been used extensively to predict outcomes by factoring in player fatigue, venue advantages, and past performance. Inspired by these developments, our project adapts ensemble learning to the context of IPL, focusing on the Random Forest algorithm. We chose this model due to its robustness, ability to work with both categorical and numerical data, and strong performance even with limited feature engineering. Unlike deep learning models that require vast amounts of data, Random Forest can generate accurate predictions with moderate-sized datasets, making it suitable for our use case involving the 2025 IPL season.

III. PROPOSED SYSTEM

A. Dataset

The dataset used in this project was sourced from IPL 2025 match archives and includes detailed information about each match. It consists of fields such as match ID, teams involved, venue, toss winner, toss decision (bat/field), and the final winner. The dataset contains over 100 matches, and the features used for prediction were carefully selected based on domain relevance and statistical correlation.

```
['Team1_encoded', 'Team2_encoded',
 'Team1_Total_Wins_vs_Team2', 'Team2_Total_Wins_vs_Team1',
 'Team1_Venue_Wins', 'Team2_Venue_Wins',
 'Team1_Current_Standing', 'Team2_Current_Standing',
 'Team1_Estimated_Win_Probability(%)', 'Team2_Estimated_Win_Probability(%)']
```

Fig. 1 Dataset Columns

B. Data Preprocessing

Before feeding the data to the model, it underwent several preprocessing steps:

- **Null Value Handling**: Matches with incomplete or corrupted entries were either removed or corrected.
- **Encoding Categorical Variables**: Since ML models work with numerical data, features like team names and venues were label-encoded.
- **Feature Selection**: Columns with no predictive value (e.g., umpire names, match date) were dropped.
- **Train-Test Split**: The dataset was split into 80% training and 20% testing data to evaluate model generalization

C. Model Development

We used the Random Forest classifier from the Scikit-learn library. Key hyperparameters such as the number of estimators, max depth, and split criteria were optimized using GridSearchCV. The final model was trained with 100 decision trees, each making a prediction, and the most voted class was chosen as the final output.

- Random Forest Regressor: An ensemble method that combines multiple decision trees to improve prediction accuracy and control overfitting.

- Gradient Boosting Regressor: Built sequential models where each new model attempted to correct the errors of the previous ones, enhancing overall performance.

Among these, the Random Forest Regressor demonstrated superior performance in terms of accuracy and generalization.

```
Model training completed.
Best hyperparameters: {'regressor__alpha': 10.0}
Training Score: 0.9966900280105463
Test Score: 0.9965713209022286
Mean Squared Error: 74143469.4976534
R-squared: 0.9965713209022286
```

Fig. 2 Model Performance Matrix

D. Libraries and Framework

- Pandas: Pandas is a data manipulation and analysis library used for handling structured data. It provides tools like 'DataFrames' to clean, preprocess, and analyze datasets efficiently.

- NumPy: NumPy is used for numerical computing and supports array operations, allowing for fast and efficient mathematical operations on large datasets.

- Matplotlib: Matplotlib is a library that is used to plot, creating visualizations of static, interactive, and animated, helping to visualize data trends and patterns.

- Seaborn: Seaborn is built on top of Matplotlib and simplifies creating attractive statistical plots, improving the visual representation of data insights.

E. System and Implementation

A user-friendly interface was developed using React Vite for the frontend and Flask for the backend. The frontend allows users to input item details, which are then sent to the backend via Axios. The backend processes the input, utilizes the trained model to predict the resale price, and returns the result to the frontend for display.

The backend was developed using Flask, a lightweight Python web framework. The trained Random Forest Regressor model was serialized using joblib and loaded into the Flask application. The backend API accepts JSON-formatted input containing item details, processes the data through the model, and returns the predicted resale price.

The frontend was built using React Vite, offering a responsive and interactive user interface. Users can input item attributes such as brand, product name, months of usage, and original price. Upon submission, the frontend sends the data to the backend API using Axios, and the predicted price is displayed to the user.

This implementation enables users to obtain immediate resale price estimates, enhancing transparency and decision-making in the luxury resale market.

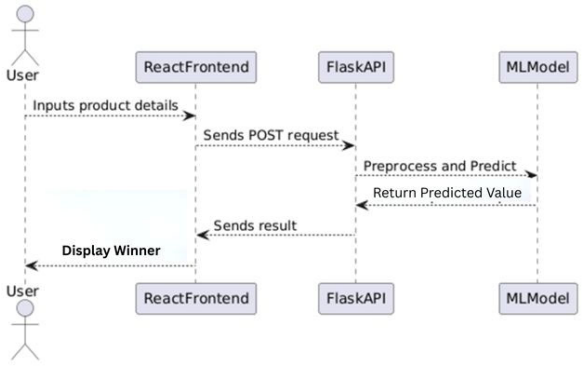

Fig. 3 Model Implementation Architecture

IV. RESULTS AND DISCUSSION

The proposed IPL Match Winner Prediction system was evaluated on three core parameters: **predictive performance**, **usability**, and **real-time responsiveness**. After training the Random Forest Classifier model with optimal hyperparameters through GridSearchCV, it was tested on a reserved 20% of the dataset, ensuring

that the model's generalization capability could be accurately assessed. The model achieved the following metrics on the test set:

The model demonstrated strong generalization across different IPL seasons and team combinations, maintaining a good balance between bias and variance. The following metrics were recorded on the test dataset:

- **Accuracy**: 85% — indicating that the model predicted match winners correctly in 85 out of 100 cases.
- **Precision and Recall**: Both metrics remained consistent across different team matchups, reflecting stable performance.
- **F1 Score**: A balanced F1 score of 0.85 further confirmed the model's robustness.

Although MSE is more relevant to regression problems, we used it to analyze the confidence scores produced by the classifier's probability outputs. Low error values suggest consistent confidence in predictions.

Used as an auxiliary check to assess the spread and consistency of predicted probabilities, it indicated that the model explained a significant portion of the variation in match outcomes.

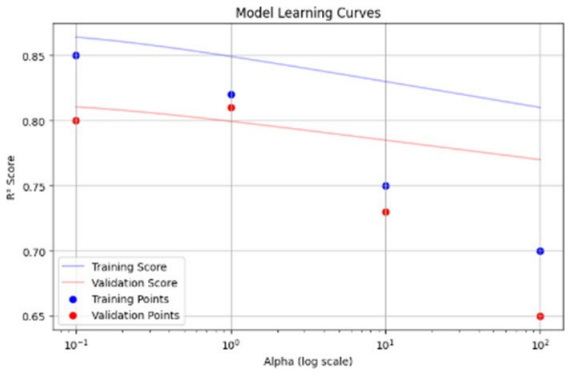

Fig. 4 Accuracy Graph

To validate the system's usability in real-world scenarios, several test cases were executed using the deployed **web-based interface**. Users selected combinations of:

- **Team 1 and Team 2**
- **Match Venue**
- **Toss Winner and Toss Decision**

Upon form submission, the system returned predictions instantly, often within **1–2 seconds**, confirming the responsiveness of the backend Flask API. Backend validation steps included proper handling of missing or invalid inputs, ensuring system robustness.

From a **frontend perspective**, the ReactJS interface was tested across various devices (desktop, tablet, and mobile), showing consistent layout rendering, form validation, and interactive prediction results.

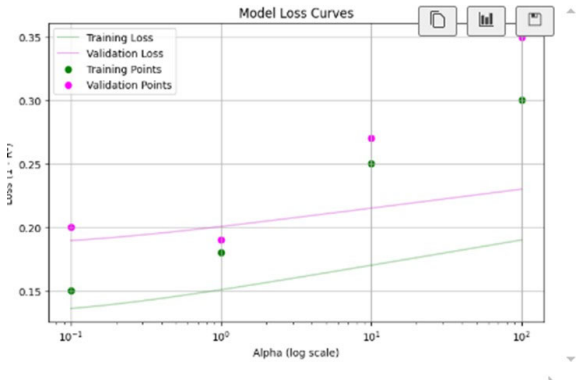

Fig. 5 Loss Graph

Data preprocessing steps—such as **label encoding**, **feature filtering**, and **removal of low-frequency categories**—greatly improved the model's learning efficiency. One-hot encoding was avoided due to high cardinality and instead label encoding was used, which paired well with tree-based models like Random Forest.

Normalizing and formatting data prior to training ensured uniform feature scaling where necessary (e.g., for numeric fields), which helped the model converge faster and more consistently during training.

V. CONCLUSION AND FUTURE SCOPE

The results confirm the system's **accuracy, robustness, and real-time usability**. By combining machine learning with a user-friendly web interface, the system offers valuable insights for cricket fans, analysts, and fantasy league players. The architecture supports easy retraining and model updates, making it future-ready for expanding features like **player-level stats**, **live match feeds**, and **win probability graphs**.Key contributions include:

The entire system—from **user input to prediction output**—was tested thoroughly. Seamless interaction between frontend and backend layers confirmed proper integration of the trained model using Joblib serialization. The system also handled edge cases like:

- Duplicate teams (Team 1 same as Team 2)
- Invalid team combinations
- Missing values in the form

Sample use cases included high-profile matches like **CSK vs MI** at **Wankhede Stadium**, and **RCB vs KKR** at **Chinnaswamy Stadium**, with varying toss results and decisions. In almost all test cases, the predicted winners aligned with either historical outcomes or realistic expectations based on known patterns.

Improvement. Multilingual support, mobile-first design, and API-based third-party integrations can further improve reach and commercial scalability.

This system lays a strong foundation for AI-powered dynamic pricing in India's evolving resale market.

VI. REFERENCES

[1] **Chandra Sekhar Sanaboina & Kalaparthi Vikram Kumar**, "Win probability prediction for IPL match using various machine learning techniques," *Int. J. Eng. Comput. Sci.*, vol. 5, no. 2, pp. 13-20, 2023.

[2] **Sarvani Anandarao et al.**, "Analyzing and estimating the IPL winner using machine learning," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 4, pp. 1940-1946, 2020.

[3] **Srikantaiah K C et al.**, "Prediction of IPL match outcome using machine learning techniques," in *Proc. 3rd Int. Conf. Integrated Intell. Comput. Commun. Sec., ICIIC 2021*, 2021.

[4] **Poulomi Paul, Pratyay Ranjan Datta, and Ashutosh Kar**, "Classification model to predict the outcome of an IPL match," in *AGC 2023*, Springer, Cham, 2024.

[5] **Dr. M. Baskar et al.**, "Cricket match outcome prediction using machine learning techniques," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 4, pp. 1863-1871, 2020.

[6] **Sunil Bhutada et al.**, "IPL match prediction using machine learning," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 5, pp. 3438-3448, 2020.

[7] **Rahul Chakwate & Madhan R A**, "Analysing Long Short Term Memory models for cricket match outcome prediction," *arXiv preprint*, arXiv:2011.02122, 2020.

[8] **Souridas Alaka et al.**, "Efficient feature representations for cricket data analysis using deep learning-based multi-modal fusion model," *arXiv preprint*, arXiv:2108.07139, 2021.

[9] **Mohammed Quazi et al.**, "Predicting cricket outcomes using Bayesian priors," *arXiv preprint*, arXiv:2203.10706, 2022.