SMcric: IPL Cricket Match Winning Prediction

Dr. S.J. Savita
Dept. of CSE(Data Science)
RNS Institute of Technology
Bangalore,India
savita.sj@rnsit.ac.in

Manoj V
Dept. of CSE(Data Science)
RNS Institute of Technology
Bangalore,India
1rn22cd042.manojv@rnsit.ac.in

Abhishek M
Dept. of CSE(Data Science)
RNS Institute of Technology
Bangalore, India
1rn22cd002.abhishek@rnsit.ac.in

***Abstract*—This paper presents a machine learning model for predicting cricket match outcomes using real-time and historical match data. The system employs an XGBoost and Random Forest classifier models, trained on Indian Premier League (IPL) match datasets, incorporating team performance metrics, venue statistics, toss decisions, and in-game situational variables (required run rate, remaining wickets, and overs). Feature engineering captures key predictive elements such as win rates, toss advantages, recent form, and venue-specific trends, while SelectFromModel optimizes feature selection. To enhance dynamic probability estimation, the model integrates Beta distribution for assessing chase difficulty and adjusts predictions based on pressure factors in critical match situations. The system demonstrates robust performance through cross-validation and test-set evaluation, effectively adapting predictions as matches progress. Experimental results highlight the model's capability to provide accurate, real-time win probability assessments, offering valuable insights for cricket analytics and decision-making.**

Keywords— Cricket prediction, Machine learning, XGBoost, Random Forest, Sports analytics, Probability estimation, Feature engineering

I. Introduction

The Indian Premier League (IPL), since its inception in 2008, has revolutionized cricket by blending sports and entertainment through its fast-paced Twenty20 (T20) format. What began as an ambitious domestic tournament has grown into a global sporting phenomenon, fundamentally transforming cricket's landscape. The league's success has not only popularized the T20 format but also generated unprecedented amounts of performance data, creating new opportunities for data-driven sports analytics.

Cricket match winning probability predictions are valuable for various stakeholders in the sport, including teams, fans, broadcasters, and betting industries. Cricket teams invest heavily in improving performance and strategy, requiring smart decision-making before and during matches. To aid this, a machine learning model is created that identifies the probability of winning for IPL teams based on past match statistics and in-game features in real-time. This allows team management, analysts, and fans to make data-driven decisions by having insight into match dynamics prior to play and throughout the game

Current cricket prediction models are too static - they analyze pre-match data but fail to adapt as the game unfolds. We need smarter systems that update predictions in real-time based on live match situations like wickets and run rates, giving teams and fans truly useful insights throughout the game.

IPL match win prediction project helps by providing smart, data-driven insights into match outcomes using machine learning. It blends historical match statistics with real-time match situations like required runs, overs, and wickets to deliver dynamic win probabilities. While currently limited to completed match data, the architecture is fully prepared for live predictions once API access is available. This makes your project valuable for fans, fantasy players, analysts, and even team strategists, offering them real-time, situation-aware match forecasts and enhancing their understanding of the game.

The research paper is organized into five key sections. In Section I, Introduction highlights the existing research gap, clearly defines the objectives and aims of the study. In Section II, Literature Survey reviews relevant existing work to provide context and justify the proposed research. In Section III, Methodology presents the problem in detail and explains the approach adopted to address it. In Section IV, Results and Discussion section analyzes the outcomes of the proposed solution and interprets their significance. Finally in Section V, the Conclusion and Future Work summarizes the findings and outlines potential directions for further research.

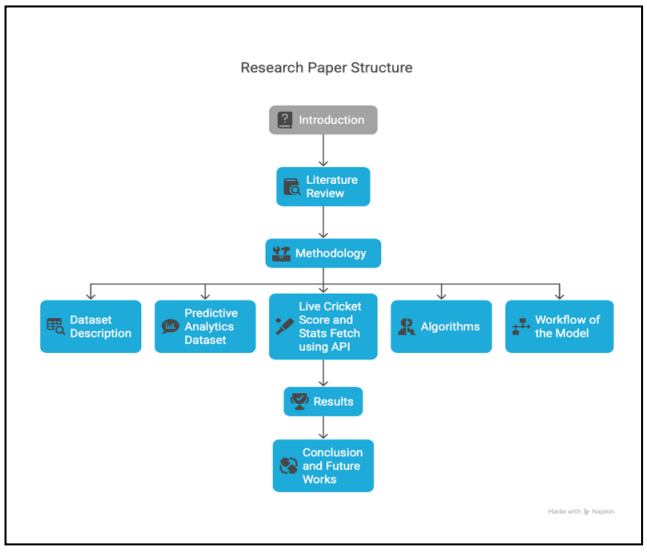

Figure 1: Research Paper Overview

II. Related Works

The prediction of cricket match outcomes has gained significant attention in recent years, driven by advancements

in machine learning (ML) techniques. Various studies have explored different ML algorithms and their effectiveness in predicting outcomes for different formats of cricket, including T20, ODI, and Test matches.

Zoha Ahsan et al. (2017) developed a prediction model for T20 and Test matches using various ML algorithms on data from 1980 to 2017. Gradient Boosted Trees achieved the highest accuracy, and SMOTE was used to balance the dataset. The study emphasized ML adaptability across formats and suggested exploring dynamic scenarios and larger datasets [1].

Daniel Mago Vistro et al. (2021) focused on predicting the winner of Indian Premier League (IPL) matches, utilizing historical data from 2008 to 2017. The study initially implemented a Decision Tree model, which was followed by the application of Random Forest and XGBoost algorithms. The results revealed that XGBoost delivered the highest accuracy among the tested models, underscoring the power of ensemble methods in sports outcome prediction. The study highlights the effectiveness of parameter tuning in improving the performance of tree-based algorithms, particularly for IPL match outcome prediction [2].

Inam Ul Haq et al. explored the prediction of One Day International (ODI) cricket match outcomes, using a dataset of 7,734 ODI matches collected from Kaggle. They evaluated the performance of K-Nearest Neighbor (KNN) and XGBoost algorithms, with KNN yielding the highest prediction accuracy. The study illustrated the application of ML in sports prediction, with the potential for extending the model to other cricket formats and even other sports. The researchers emphasized the importance of historical match data in generating accurate predictions [3].

Shristi Priya et al. investigated winning prediction in T20 cricket, particularly focusing on the IPL from 2008 to 2020. Key features such as team names, toss details, city, and venue were encoded and used for model training. Among various classifiers, Random Forest emerged as the most accurate. The study found that the inclusion of venue details and proper feature encoding significantly improved prediction accuracy, highlighting the critical role of data preprocessing and feature engineering in model performance [4].

Shilpi Agrawal et al. proposed a machine learning model to predict IPL T-20 match outcomes, employing classifiers such as Support Vector Machine (SVM), Naïve Bayes, and CTree. The study used ball-by-ball and match-level data, achieving high accuracy with Naïve Bayes. They emphasized the importance of data preprocessing and feature engineering, with features such as average run rate, power play performance, and toss results being integral to the model's success. This work illustrated the potential of ML in addressing cricket's inherent unpredictability [5].

Asif and McHale [1] developed a dynamic logistic regression (DLR) model aimed at estimating the win probability for the batting team in One-Day International (ODI) cricket. Their approach allows the model's coefficients to evolve smoothly throughout the match, capturing the changing dynamics and providing consistent, interpretable forecasts in real time. The model was trained using data from over 600 ODI matches played between 2004 and 2010 and was benchmarked against live betting odds to validate its predictive accuracy. Key match-specific features—such as runs required, balls remaining, wickets lost, and relative team strength—were used as covariates, with separate modeling for first and second innings scenarios[6].

H. Barot et al. analyzed IPL match outcomes by considering factors such as toss, weather, pitch conditions, and player performance. They introduced Batting and Bowling Indexes for evaluating player contributions and used Naïve Bayes, SVM, and Random Forest algorithms. Their results highlighted the significance of team form and strength in making accurate match predictions. Future work in this area was suggested to focus on assessing player value, which could further enhance prediction accuracy [7].

Nirmala et al. employed logistic regression for predicting IPL match outcomes, considering factors such as team strength, player form, and match conditions. Their research successfully predicted the probability of winning at various stages of a match. Similar studies, using algorithms like random forests and decision trees, have further demonstrated the potential of ML in improving match prediction accuracy. However, challenges remain in refining models to better handle complex dynamic factors [8].

The review shows an increasing use of machine learning for predicting cricket outcomes across formats, with ensemble methods like Random Forest and XGBoost performing well. Despite progress, challenges remain in feature selection and real-time data integration.

III. METHODOLOGY

Machine learning techniques play an important role in the field of cricket matches to predict the match outcomes. In this paper we used Random Forest Classifier and XGBoost algorithms to predict the match winning probabilities of IPL team based on their previous match stats.

A. Dataset Description

There are many datasets available on Kaggle for Cricket match prediction. We have taken an updated dataset from Kaggle which is Cricket winner prediction. The dataset scraped from Stasguru website of ESPN and contains 13 features such as match_id, city, player_of_match, venue, team1, team2, toss_winner toss_decision, winner, result, result_margin, target_runs, target_overs. The number of instances in this dataset is 1090 IPL matches.

Column Name	**Description**
match_id	Unique identifier for each IPL match
city	City where the match was played
player_of_match	Player awarded "Player of the Match"
venue	Stadium or venue where the match took place
team1	First team listed in the fixture
team2	Second team listed in the fixture
toss_winner	Team that won the toss

toss_decision	Toss decision made by toss winner (bat or field)
winner	Team that won the match
result	Type of win (runs or wickets)
result_margin	Win margin (either by number of runs or wickets)
target_runs	Total runs set for the team batting second
target_overs	Number of overs available to chase (generally 20, adjusted if DLS applies)

Table 1: Summary of IPL Match Dataset Features

B. Predictive Analytics Dataset

i. Basic Data Pre-processing

Firstly, we loaded the Dataset 'IPL_Matches.csv' on Jupyter notebook and observed team name variations during each season and standardization of team names like: Rising Pune Supergiants → Rising Pune Supergiant, Deccan Chargers → Sunrisers Hyderabad, Delhi Daredevils → Delhi Capitals, Kings XI Punjab → Punjab Kings, Royal Challengers Bangalore → Royal Challengers Bengaluru.

Fixing city names City Name Fixes: Bangalore → Bengaluru, Navi Mumbai → Mumbai

Handling the missing values like:

- city: filled with 'Unknown'
- player_of_match, target_runs, target_overs, toss_decision, winner: filled with 'Not Available'
- result_margin: filled with the column's mean
- Rows with missing winners were dropped, as they do not contribute to predictive modeling

Dropped Unnecessary Columns: Removed irrelevant or incomplete columns such as umpire1, umpire2, method, match_type, etc.

ii. Exploratory Data Analysis (EDA)

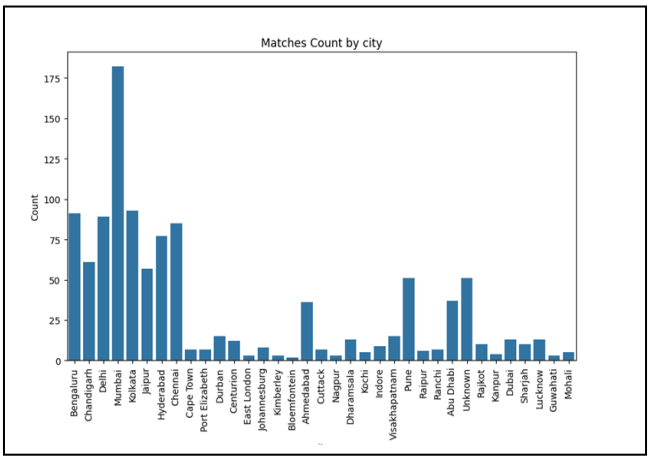

Figure 2: City-wise Match Count

In the given Figure 2.1, the city-wise match count is visualized using a bar chart created with seaborn.countplot. This visualization helps in identifying the cities that have hosted the most IPL matches, offering insights into venue popularity and regional preferences for match hosting.

In Figure 2.2, a heatmap of head-to-head wins between IPL teams is shown, where each cell represents the number of times Team1 has beaten Team2. The color intensity highlights the dominance, with higher win counts represented by warmer tones.

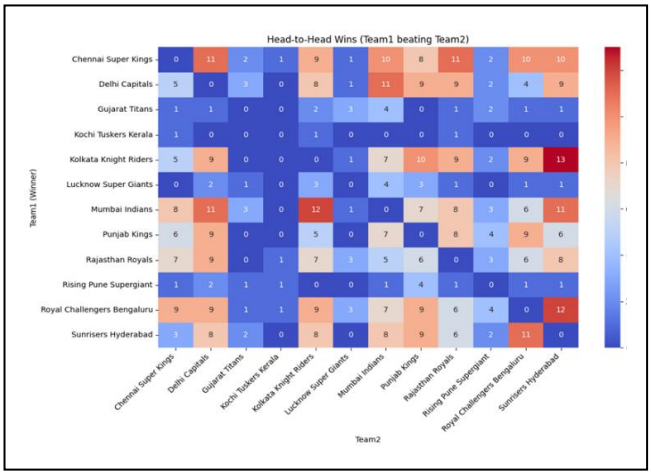

Figure 3: Head to head win for each team

C. Live cricket score and stats fetch using API

i. Implementation Details (Live Data Integration)

To integrate real-time match outcomes into our predictive system, we used the RapidAPI platform to fetch cricket match scores and results. The API response data was parsed and selectively filtered to extract only the necessary fields. These were then presented in custom-designed score containers on the frontend, displaying key match outcomes and team performances. Although the API provided comprehensive data, we opted to retrieve only essential results to keep the system lightweight and responsive.

ii. Limitations (Data Access Constraints)

One of the primary limitations of our project was the restricted access to complete live match data due to budget constraints. While the Rapid API service offered rich cricket data, our limited plan only allowed access to final match results, excluding live stats such as ball-by-ball updates, player performance metrics, and real-time events. This constraint limited the scope of real-time probability updates during ongoing matches. However, we successfully utilized the available results to validate and improve the model's post-match predictions.

Despite this limitation, the integration successfully enabled our system to stay updated with the latest IPL match outcomes, which was essential for validating and testing our win prediction model.

D. Algorithms

i. Random Forest Classifier

Random Forest is like having a team of cricket experts voting independently on the match outcome. Each expert (decision tree) looks at the data separately, and the final prediction comes from their majority vote. This approach is:

- Quick to train (all experts work at the same time)
- Hard to trick (resistant to overfitting)
- Easy to understand (clear feature importance) But sometimes all the experts might miss subtle patterns that develop over time.

ii. XGBoost (Extreme Gradient Boosting)

XGBoost works more like a coach analyzing a match inning by inning. It learns from each mistake and adjusts its strategy:

- Builds simpler models that improve sequentially
- Handles tricky situations (like imbalanced data) better
- Can squeeze out extra accuracy with fine-tuning
- Works faster with GPU support The tradeoff? It requires more careful setup and is slightly harder to interpret at first glance.

Our system automatically runs both models and picks the best model. Before each prediction, it:

1. Tests both models on recent match data
2. Checks which perform better (using CV scores)
3. Selects the best model for that specific prediction

This smart selection means you always get us the most accurate predictions possible and Models that adapt to different match conditions. Whether it's a high-stakes playoff or a regular season game, our dual-model system ensures we're always using the best tool for the job. The Random Forest gives us reliable baseline predictions, while XGBoost steps in when we need that extra edge in tight situations.

E. Workflow of the Model

The IPL Win Prediction System follows a structured machine learning pipeline to forecast match outcomes based on historical and real-time data. The system architecture includes five core stages: Data Preparation and Feature Engineering, Automated Model Selection, Model Training and Hyperparameter Tuning, Base Probability Generation and Live Match Probability Prediction.

i. Data Preparation and Feature Engineering

The data preparation phase plays a critical role in ensuring the reliability and relevance of the inputs used for model training. Historical IPL match data is first acquired from structured CSV files containing details such as participating teams, venue and city, toss outcomes, match results, runs scored, overs bowled, and wickets lost.

To make the data compatible with machine learning algorithms, categorical variables including team names, cities, and toss decisions are encoded numerically using label encoding. The feature set is then enriched through statistical derivations and domain-informed metrics.

Key features include:

- **Team-Level Statistics:** Metrics such as overall win rate, toss win impact, and win probability when batting first are calculated to capture long-term team performance and strategic tendencies.
- **Recent Form:** The short-term performance of each team is assessed using win rates over the last ten matches.
- **Head-to-Head Records:** Historical win rates between every team pairing are computed to reflect the strength of specific rivalries.
- **Venue-Specific Trends:** Each venue's batting-first win rate and average first-innings score are included to capture ground-dependent behavior.
- **Dynamic Match Variables:** Features such as normalized target (adjusted by venue averages) and required run rate are derived to capture situational match difficulty.
- **Feature Pruning:** Redundant or low-importance features are removed through manual inspection and correlation analysis to reduce dimensionality and prevent overfitting.

This comprehensive feature engineering process ensures that the model captures both static contextual factors and dynamic in-game variables, thereby improving its generalizability and predictive accuracy across different match scenarios.

ii. Automated Model Selection and Execution

An automated workflow was implemented to evaluate and compare multiple classification algorithms, including XGBoost and Random Forest, based on validation accuracy. The system handles end-to-end processes such as data loading, model training, evaluation, and prediction. After testing on both the complete historical dataset and a subset comprising the most recent four IPL seasons, the Random Forest model consistently outperformed XGBoost in terms of classification accuracy. Based on these results, Random Forest was selected as the final model for generating pre-match win probabilities

iii. Model Training and Hyperparameter Tuning

Following feature engineering, a structured training phase is conducted to build a robust predictive model for IPL match outcomes. This phase includes model selection, hyperparameter tuning, feature refinement, and performance evaluation.

- **Model Selection:** A Random Forest Classifier is chosen due to its ability to handle diverse feature types, reduce overfitting through ensemble learning, and provide interpretable feature importance scores. Its resilience to noise and minimal preprocessing requirements makes it well-suited for sports analytics applications.
- **Hyperparameter Optimization:** Hyperparameter tuning is a critical step in improving both the accuracy and reliability of the predictive model. Rather than relying on default values, this process systematically searches for the optimal combination of model parameters that yield the best performance.

For the Random Forest classifier, key hyperparameters such as the number of trees, maximum tree depth, and the minimum number of samples required to split or retain a node are fine-tuned using a grid search strategy. The GridSearchCV method is used to perform an exhaustive evaluation over a predefined parameter grid, employing cross-validation to assess model performance for each configuration. The best-performing set of hyperparameters is selected based on validation accuracy, ensuring that the final model generalizes well to unseen data and avoids both underfitting and overfitting.

- **Feature Selection:** To reduce complexity and highlight influential inputs, a feature selection process based on model-derived importance scores is applied. Using SelectFromModel, only features with importance values above the median are retained. This helps enhance generalizability while minimizing overfitting.
- **Training and Evaluation:** The data is split into training and testing subsets using stratified sampling to maintain balanced representation of match outcomes. The model is trained on the training data and evaluated on the test set using several performance metrics:
    - **Accuracy:** Measures the proportion of correctly predicted outcomes.
    - **Classification Report:** Includes precision, recall, and F1-score for each class, offering deeper insight into prediction quality.
    - **Confusion Matrix:** Summarizes correct and incorrect predictions across classes.
    -
    - **Cross-Validation Scores:** Reports the mean and variance of accuracy across multiple folds to assess model stability and generalizability.

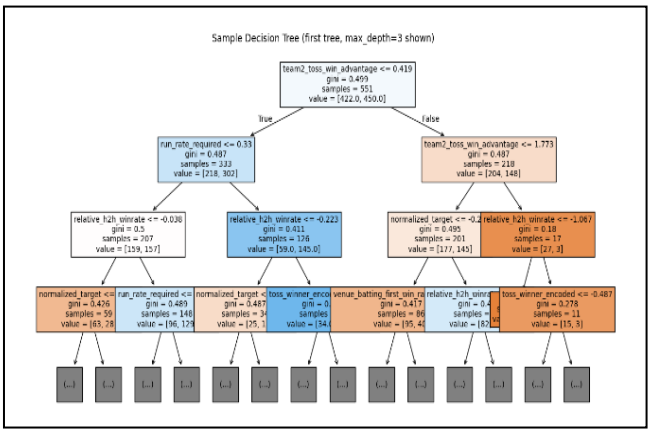

Figure 5: Sample Decision tree

This structured approach ensures the development of a reliable and well-validated model capable of accurately predicting match outcomes under varied conditions.

iv. Base Probability Generation

After training, the model is employed to generate base (pre-match) win probabilities for teams in upcoming match scenarios. This process begins with constructing a feature vector for the new match, using the same encoding and feature engineering logic as applied during training. Inputs include team identities, venue details, toss outcome, and relevant statistical indicators.

The trained Random Forest classifier then predicts the probability of each team winning using the predict_proba method. These outputs reflect historical performance trends, contextual strengths, and venue effects, independent of in-game developments.

The resulting base probabilities act as an informed starting point for win prediction. They are later blended with live match data to produce dynamic, context-aware forecasts as the game progresses.

v. Live Match Probability Prediction

Cricket match outcome prediction is a complex task, especially in the dynamic context of T20 leagues such as the Indian Premier League (IPL). Traditional machine learning models, such as Random Forests trained on historical data, provide pre-match probabilities that reflect team strengths, venue effects, and other static features. However, as a match progresses, the real-time context—required runs, balls left, and wickets in hand—becomes more critical to accurate probabilistic forecasting.

To bridge this gap, our approach takes the classifier's pre-match probability and continuously adjusts it using live match information:

1. Base Probability Estimation

The live prediction model builds on pre-match win probabilities generated by the Random Forest classifier trained on historical IPL data. These base probabilities, computed using predict_proba(), reflect each team's relative strength based on contextual features such as team form, venue trends, and head-to-head performance, and serve as the foundation for in-game updates.

2. Real-Time Situation Modeling

As the match progresses, static pre-match probabilities lose relevance, requiring adaptation to the evolving match context. To address this, a live situation model is employed to estimate win probabilities based on real-time match variables. The model incorporates key in-play features, including:

- Required runs
- Remaining overs and balls
- Wickets lost (and consequently, wickets in hand)
- Batting resources remaining

3. Resource Estimation

To quantify the batting resources remaining for the chasing team, a function named get_DLS_resource is used. This function returns the proportion of resources left based on the current number of overs remaining and wickets lost. It leverages a truncated version of the Duckworth–Lewis–Stern (DLS) resource table, which maps the state of play defined by overs and wickets to a corresponding resource percentage. This resource value serves as a critical input in the real-time probability estimation model.

4. Pressure Factor Adjustment

To account for situational pressure during the chase, a pressure adjustment is applied using the calculate_pressure_factor function. This function quantifies the impact of high-pressure scenarios—such as elevated required run rates combined with limited wickets in hand—by returning a multiplier in the range [0, 1]. The multiplier effectively reduces the win probability in unfavorable match conditions, enhancing the realism and sensitivity of the live prediction model.

5. Live Win Probability Calculation

The live win probability for the chasing team is computed as follows:

- **Heuristic Rules:** If the required runs are comfortably less than balls left (and wickets in hand are high), the win probability is set near certainty (e.g., 0.99). Conversely, if the situation is extremely unfavorable (few wickets and high required run rate), the probability is set low (e.g., 0.10).
- **Logistic Model:** For estimating the in-play win probability of the chasing team, a logistic (sigmoid) function is employed. The score function is defined as:

$$\text{score} = 3.0 - 8.0 \cdot \text{rr\_ratio} + 0.25 \cdot \text{wickets\_in\_hand} + 7 \cdot \text{resources\_remaining} \tag{1}$$

$$P_{\text{chase}} = \frac{1}{1 + e^{-\text{Score}}} \tag{2}$$

where rr_ratio is the ratio of the current required runs per ball to the original required run rate per ball, wickets_in_hand represents the remaining wickets of the chasing team, and resources_remaining is obtained from the Duckworth–Lewis–Stern (DLS) resource table as a proportion (0 to 1). This formulation ensures that the win probability increases with more wickets and available resources, and decreases when the required run rate increases, effectively capturing pressure in a chasing scenario.

Our logistic regression model is inspired by the dynamic in-play framework introduced by Asif and McHale [1], which estimates win probabilities in One-Day Internationals (ODIs) using evolving match covariates. While their model adapts coefficients over time using dynamic smoothing, we adopt a simplified, static version suited for T20 formats like the IPL. The coefficients in Eq. (1) were manually calibrated to reflect typical match dynamics, allowing the logistic function in Eq. (2) to output context-aware win probabilities. A pressure factor is later applied to refine the estimate further based on situational volatility.

6. Probability Blending

To ensure a smooth transition between pre-match and in-game win probabilities, a convex combination of the pre-match model output and the live match logistic estimate is computed. The blending weights are dynamically adjusted based on match progression.

Let the match progress be defined as:

$$\text{match\_progress} = 1 - \frac{\text{remaining\_overs}}{\text{total\_overs}} \tag{2}$$

Then, the weights for blending are defined as follows:

- **Pre-match weight**:

$$w_{\text{pre}} = \max(0.05,\ 1 - \text{match\_progress}) \tag{3}$$

- **Live probability weight**:

$$w_{\text{live}} = 1 - w_{\text{pre}}$$

The final win probability for the chasing team is given by:

$$P_{\text{chase}} = w_{\text{pre}} \cdot P_{\text{base}} + w_{\text{live}} \cdot P_{\text{live}} \tag{4}$$

where P_base is the pre-match win probability (from a trained classifier) and P_live is the in-game probability estimated using the logistic function based on match context.

For the team bowling second, the win probability is simply:

$$P_{\text{bowl}} = 1 - P_{\text{chase}} \tag{5}$$

This weighted approach enables the model to rely more heavily on historical data early in the match and gradually shift toward real-time dynamics as the game progresses.

7. Output

The proposed method returns the live win probabilities for both teams, along with key diagnostic indicators. These include the required run rate, remaining resources, the unblended live chase probability, the pre-match weight, and the identified chasing team. This design enables the model to produce predictions that are both historically informed and context aware. By combining static pre-match probabilities with dynamic in-play factors, the system mitigates the limitations of relying solely on historical models or situational heuristics.game progresses.

IV. Results

The performance of the proposed IPL match win prediction model was evaluated using historical match data. The model was implemented using a Random Forest Classifier trained on a comprehensive set of engineered features, including team form, venue-specific statistics, head-to-head win rates, and dynamic match situation indicators.

A. Classification Performance

The model's predictive capability was assessed on a held-out test set. The confusion matrix in Fig. 1 illustrates the distribution of correct and incorrect predictions for both outcome classes:

- **True Positives (TP):** Correct predictions for Team 1 win,
- **True Negatives (TN):** Correct predictions for Team 2 win,
- **False Positives (FP):** Incorrectly predicted Team 1 win,
- **False Negatives (FN):** Incorrectly predicted Team 2 win.

As shown in the matrix, the majority of predictions fall into the correct categories, indicating strong classification performance.

B. Accuracy and Cross-Validation

The model achieved a **test set accuracy of 85.11%**. Additionally, 5-fold cross-validation yielded the following accuracy scores:

[0.8511, 0.7021, 0.5957, 0.6304, 0.8043]

with a **mean accuracy of 71.67%**, suggesting consistent, albeit slightly variable, generalization across different folds.

C. Precision, Recall, and F1-Score

The classification report reveals balanced results across both classes:

- **Class 0 (e.g., Team 1 wins):** Precision = 0.86, Recall = 0.83, F1-score = 0.84
- **Class 1 (e.g., Team 2 wins):** Precision = 0.84, Recall = 0.88, F1-score = 0.86
- **Macro and Weighted Averages:** All exceed 0.85, reflecting consistent model behavior.

```
Final Model Accuracy: 0.8511

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.83      0.84        23
           1       0.84      0.88      0.86        24

    accuracy                           0.85        47
   macro avg       0.85      0.85      0.85        47
weighted avg       0.85      0.85      0.85        47


Confusion Matrix:
[[19  4]
 [ 3 21]]

Cross-Validation Scores:
[0.85106383 0.70212766 0.59574468 0.63043478 0.80434783]
CV Mean Accuracy: 0.7167
```

Figure 6: Confusion matrix and classification metrics for IPL win prediction model

These results validate the effectiveness of the model's architecture and feature design. The classifier demonstrates reliable pre-match and in-game prediction capability, with robust performance metrics across all evaluation dimensions. The ability to integrate contextual and statistical factors enables the model to provide valuable probabilistic insights into IPL match outcomes.

V. Conclusion and Future works

Our IPL prediction system marks a significant advancement in cricket analytics, combining machine learning with deep cricket understanding to deliver dynamic, match-aware insights. The model achieved up to 85.11% accuracy in predicting match outcomes, leveraging smart feature engineering and historical data. We successfully integrated the RapidAPI service to fetch completed match results, enabling real-time score-based predictions in retrospective scenarios. The system intelligently adjusts base probabilities using live match factors such as required run rate, remaining wickets, and overs left, allowing for situation-aware predictions. Although the architecture supports full live-match integration, current budget constraints restrict access to live scoring APIs, limiting real-time updates. However, the entire framework is in place and ready to be activated once appropriate resources are secured.

References

[1] S. Priya, A. K. Gupta, A. Dwivedi and A. Prabhakar, "Analysis and Winning Prediction in T20 Cricket using Machine Learning," in Proc. 2nd Int. Conf. on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2022, pp. 1–4, doi: 10.1109/ICAECT54875.2022.9807929.

[2] D. M. Vistro, F. Rasheed and L. G. David, "The Cricket Winner Prediction with Application of Machine Learning and Data Analytics," Int. J. of Scientific & Technology Research, vol. 8, pp. 985–990, 2019.

[3] I. U. Haq, I. U. Hassan and H. A. Shah, "Machine Learning Techniques for Result Prediction of One Day International (ODI) Cricket Match," in Proc. IEEE 8th Int. Conf. for Convergence in Technology (I2CT), Lonavla, India, 2023, pp. 1–5, doi: 10.1109/I2CT57861.2023.10126241.

[4] S. Agrawal, S. P. Singh and J. K. Sharma, "Predicting Results of Indian Premier League T-20 Matches using Machine Learning," in Proc. 8th Int. Conf. on Communication Systems and Network Technologies (CSNT), Bhopal, India, 2018, pp. 67–71, doi: 10.1109/CSNT.2018.8820235.

[5] Kumar, R. Kumar and P. Kumar, "Outcome Prediction of ODI Cricket Matches using Decision Trees and MLP Networks," in Proc. 1st Int. Conf. on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 2018, pp. 343–347, doi: 10.1109/ICSCCC.2018.8703301.

[6] M. Asif and I. G. McHale, "In-play forecasting of win probability in One-Day International cricket: a dynamic logistic regression model," Int. J. Forecasting, vol. 32, no. 1, pp. 34–43, 2016.

[7] R. Suguna, Y. P. Kumar, J. S. Prakash, P. S. Neethu and S. Kiran, "Utilizing Machine Learning for Sport Data Analytics in Cricket: Score Prediction and Player Categorization," in Proc. IEEE 3rd Mysore Sub Section Int. Conf. (MysuruCon), Hassan, India, 2023, pp. 1–6, doi: 10.1109/MysuruCon59703.2023.10396955.

[8] M. M. Hatharasinghe and G. Poravi, "Data Mining and Machine Learning in Cricket Match Outcome Prediction: Missing Links," in Proc. IEEE 5th Int. Conf. for Convergence in Technology (I2CT), Bombay, India, 2019, pp. 1–4, doi: 10.1109/I2CT45611.2019.9033698.

[9] H. Barot, A. Kothari, P. Bide, B. Ahir and R. Kankaria, "Analysis and Prediction for the Indian Premier League," in Proc. Int. Conf. for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1–7, doi: 10.1109/INCET49848.2020.9153972.

[10] A. P. Nirmala, B. Gogoi, V. Asha, A. Naveen, A. Prasad and D. P. Reddy, "Analysis and Predictions of Winning Indian Premier League Match using Machine Learning Algorithm," in Proc. IEEE 12th Int. Conf. on Communication Systems and Network Technologies (CSNT), Bhopal, India, 2023, pp. 152–157, doi: 10.1109/CSNT57126.2023.10134747.

[11] S. Chakrabarty, S. Jana and P. Baral, "A Data-Driven Approach for the Prediction of Team Potential and Result During Cricket Match," in Proc. 2nd World Conf. on Communication & Computing (WCONF), Raipur, India, 2024, pp. 1–5, doi: 10.1109/WCONF61366.2024.10691967.

[12] N. Perattur, V. K., V. S. Kushwah, T. P. Sai, S. Rekha and R. Subramanyam, "Algorithms for Performance Prediction in Cricket – Factors Influencing the Player," in Proc. IEEE 16th Int. Conf. on Computational Intelligence and Communication Networks (CICN), Indore, India, 2024, pp. 1047–1051, doi: 10.1109/CICN63059.2024.10847437.

[13] A. Kumar, B. Hassan and M. F. Wasiq, "CricPredict: Resource-Aware Prediction of T20 Cricket Match," in Proc. Int. Conf. on Electrical Engineering and Computer Science (ICECOS), Palembang, Indonesia, 2024, pp. 418–423, doi: 10.1109/ICECOS63900.2024.10791075.

[14] M. A. Pramanik, M. M. H. Suzan, A. A. Biswas, M. Z. Rahman and A. Kalaiarasi, "Performance Analysis of Classification Algorithms for Outcome Prediction of T20 Cricket Tournament Matches," in Proc. Int. Conf. on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 01–07, doi: 10.1109/ICCCI54379.2022.9740867.

[15] Z. Ahsan, S. Ghumman, A. U. Rehman, S. Javaid, T. M. Ali and A. Mir, "A Comprehensive Prediction Model for T20 and Test Match Outcomes Using Machine Learning," in Proc. Int. Conf. on Engineering & Computing Technologies (ICECT), Islamabad, Pakistan, 2024, pp. 1–7, doi: 10.1109/ICECT61618.2024.10581259.

[16] P. Singh, J. Kaur and L. Singh, "Predicting IPL Victories: An Ensemble Modeling Approach Using Comprehensive Dataset Analysis," in Proc. 2nd Int. Conf. on Artificial Intelligence and Machine Learning Applications (AIMLA), Namakkal, India, 2024, pp. 1–6, doi: 10.1109/AIMLA59606.2024.10531489.

[17] T. Singh, V. Singla and P. Bhatia, "Score and winning prediction in cricket through data mining," in Proc. Int. Conf. on Soft Computing Techniques and Implementations (ICSCTI), Faridabad, India, 2015, pp. 60–66, doi: 10.1109/ICSCTI.2015.7489605.

[18] M. Subburaj, G. R. K. Rao, B. Parashar, I. Jeyabalan, H. Semban and S. Sengan, "Artificial Intelligence for Smart in Match Winning Prediction in Twenty20 Cricket League Using Machine Learning Model," in Artificial Intelligence for Smart Healthcare, P. Agarwal, K. Khanna, A. A. Elngar, A. J. Obaid and Z. Polkowski, Eds. Cham, Switzerland: Springer, 2023, pp. 33–47, doi: 10.1007/978-3-031-23602-0_3.