Prediction of IPL Win Probability Using Machine Learning Techniques

Md Tanvirul Islam, Engr. Rohul Amin, Nadim Reza
Department of Computer Science & Engineering,
Bangladesh Army University of Science and Technology, Bangladesh
Emails: 220201002@baust.edu.bd, rohul@baust.edu.bd, nadimreza@baust.edu.bd

***Abstract*—In this project, We present a machine learning based approach that can predict the win probability of matches in the Indian Premier League (IPL) during the second innings. The proposed system uses mid-match features such as current score, wickets fallen, and current run rate to predict match outcomes in real-time. Logistic Regression, Support Vector Machine (SVM), and TabNet are applied and compared. For better understanding, we present evaluation metrics including accuracy, F1-score, recall, ROC-AUC, confusion matrices, and training/validation curves. The results demonstrate that the models provide meaningful win probability predictions while avoiding data leakage, outperforming previous methods reported in the literature.**

***Index Terms*—IPL, Win Probability, Machine Learning, Logistic Regression, TabNet, Support Vector Machine, Data Leakage**

I. Introduction

Cricket is a complex and dynamic sport where the situation changes dynamically on every ball, and predicting match outcomes requires understanding multiple variables, including team composition, player performance, and in-game dynamics. Predictive analytics can be useful for coaches, teams, and analysts in making real-time decisions during matches.

A. Previous Work

Several studies have applied machine learning techniques to predict IPL match outcomes using historical match, player, and team data. Commonly used algorithms include Logistic Regression, Random Forest, Gradient Boosting, and Support Vector Machines (SVM). Real-time prediction systems also incorporate live match data to improve predictive accuracy.

In recent years, substantial research has focused on IPL win probability detection. One study conducted in 2024 used data from the 2008–2023 IPL seasons and applied multiple machine learning models, with a particular focus on Random Forest Classifier [1].

Another comprehensive study in 2025 utilized Logistic Regression and Gradient Boosting algorithms for win probability prediction. The study also provided a web interface to allow real-time win probability predictions [2].

A separate study, also in 2025, employed Logistic Regression, Random Forest, SVM, and Gradient Boosting. This model not only predicted match outcomes but also provided insights into the factors that most influence winning probabilities [3].

Additionally, a 2024 study applied multiple machine learning algorithms, including Naïve Bayes, SVM, k-Nearest Neighbor, Random Forest, Logistic Regression, ExtraTreesClassifier, and XGBoost, to predict match win probabilities [4].

B. Importance and Significance

Win probability prediction helps in strategic decision-making, improving team performance during matches. It enables analysts to quantify the impact of in-game events, such as wickets and scoring rates.

C. Need for the Study

Many existing studies suffer from data leakage, where features reveal future outcomes, leading to artificially high accuracy. Some models lack interpret ability or are not suitable for real-time mid-match prediction.

D. Contribution of This Work

Implements safe feature engineering for mid-match prediction. Applies and compares Logistic Regression, SVM, and TabNet. Provides comprehensive evaluation using multiple metrics. Compares results with findings from previous studies.

II. Literature Review

Several studies have explored the use of machine learning techniques for predicting IPL match outcomes. Table I summarizes some key works, including their datasets, classes, methodology, and train-test splits.

TABLE I
Summary of Related Work

Paper	Dataset	Class	Methodology
[1]	Historical IPL	Binary	Logistic Regression, Random Forest and (SVM)
[2]	IPL 2008–2023	Binary	Logistic Regression and Gradient Boosting
[3]	Historical IPL	Binary	Logistic Regression, RF, SVM, and XGBoost
[4]	Kaggle	Binary	Naïve Bayes, SVM, k-NN, RF, LR,and XGBoost

A. Paper Summaries

This study used historical IPL match, player, and team data to predict win probability. Logistic Regression and Gradient Boosting models were applied, and exploratory data analysis helped identify key features. However, details on the train-test split and handling of mid-match predictions were not provided.

The study did not discuss the dataset in detail or specify which factors were considered for decision-making, nor did it provide any machine learning model performance metrics [1].

Using ball-by-ball IPL data from 2008–2023, Random Forest and Logistic Regression were implemented. Feature engineering and categorical encoding were applied to improve model performance. The study used a 70%-30% train-test split but mainly focused on end-of-match predictions. However, it did not provide any information regarding the accuracy of the models [2].

Combining historical and real-time match data, Logistic Regression and Gradient Boosting models were applied. The system was deployed via Streamlit and Heroku, enabling near real-time predictions. Nevertheless, the study lacked discussion on data leakage prevention and advanced tabular learning methods. Furthermore, it did not clearly explain the dataset or how the model works [3].

This work considered team composition, venues, toss outcomes, and player statistics for the last seven IPL seasons. Multiple models were evaluated, including Logistic Regression, Random Forest, SVM, Gradient Boosting, RNN, and HMM. Oversampling and feature engineering were applied. The system was complex and computationally expensive, which limited its usability for real-time predictions [4].

B. Common Limitations

Among these, there are several common limitations are observed, Some studies suffered from data leakage when using end-of-match information for mid-match prediction. Limited focus on real-time or mid-innings prediction. Tree-based or neural models often lacked interpret ability and scalability.

III. Methodology

This section describes the approach used to predict IPL match outcomes using machine learning models, including Logistic Regression, TabNet, and SVM. The workflow consists of four major stages: data collection, preprocessing and feature engineering, model development, and evaluation.

A. Data Collection

The dataset for this study was obtained from historical IPL match and delivery-level data. Two CSV files were used:

matches.csv Contains match-level information such as teams, city, winner, and toss.

deliveries.csv: Contains ball-by-ball data including runs scored, wickets, overs, and players involved.

The dataset spans multiple IPL seasons and provides both aggregate and granular information necessary for win probability prediction.

B. Data Preprocessing and Feature Engineering

1) **Filtering Relevant Teams:** Only teams currently participating in IPL were included to reduce noise. Both the batting and bowling teams were checked against this list.
2) **Merging Match and Delivery Data:** Delivery-level data was merged with match-level information to include total match score, city, and winner in each delivery record.
3) **Inning Selection:** Only the second innings of each match was considered for prediction, as the target is to predict the probability of winning given the current state of the game.
4) **Calculating Dynamic Features:** Several mid-match features were computed: **Current score:** Cumulative runs scored by the batting team.
   Balls left: Number of deliveries remaining in the innings.
   Wickets remaining: Calculated as $10 -$ (wickets lost).
   Current Run Rate (CRR): Calculated as
   $$\text{CRR} = \frac{\text{current\_score} \times 6}{\text{balls\_bowled}}$$
   [5]
   Required Run Rate (RRR): Calculated as
   $$\text{RRR} = \frac{\text{runs\_left} \times 6}{\text{balls\_left}}$$
   where $\text{runs\_left} = \text{total\_runs\_team1} - \text{current\_score}$ [6].
5) **Handling Missing Values:** Missing values in the `player_dismissed` column were filled with 0. Any remaining missing values were either imputed using the median or removed.
6) **Encoding Categorical Variables:** Team names and city were one-hot encoded to convert them into numerical features suitable for machine learning models.
7) **Target Variable:** The target column `result` was defined as 1 if the batting team eventually won the match, otherwise 0.

C. Train-Test Split

To prevent data leakage and ensure the models learn genuine mid-match patterns, the split was performed based on matches rather than individual deliveries. Approximately 80% of matches were used for training and 20% for testing, ensuring that all deliveries from a given match belong entirely to either the training or testing set.

D. Model Development

Three models were implemented and evaluated:

1) **Logistic Regression:** A baseline model for binary classification. A pipeline was used to combine preprocessing and model fitting.
2) **TabNet:** A deep learning model for tabular data, capable of learning sequential decision rules and attention over features. TabNet was trained with Adam optimizer and early stopping to prevent overfitting.
3) **Support Vector Machine (SVM):** A traditional classifier using a linear kernel. The SVM model was trained on the same features as Logistic Regression and tuned for maximum accuracy.

E. Evaluation Metrics

The models were evaluated on both training and test sets using multiple metrics to ensure a comprehensive performance analysis:

Accuracy: Fraction of correctly predicted outcomes.

F1 Score: Harmonic mean of precision and recall to balance false positives and false negatives.

ROC-AUC: Area under the Receiver Operating Characteristic curve, showing the trade-off between true positive and false positive rates.

Confusion Matrix: Visual representation of true positives, false positives, true negatives, and false negatives.

Training and Validation Curves: Loss and accuracy curves were plotted for TabNet to monitor convergence and potential overfitting.

F. Implementation Workflow

Figure 1 illustrates the overall workflow from raw data to model evaluation:

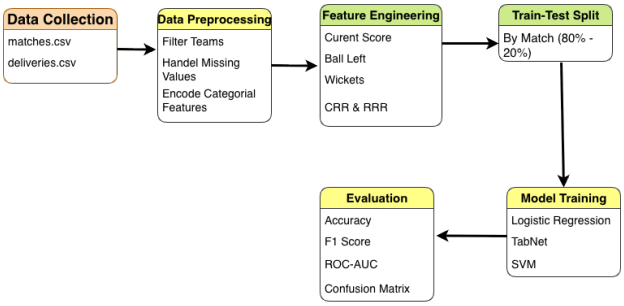

Fig. 1. Workflow of IPL Win Probability Prediction System

The methodology ensures that the models learn from mid-match features, making predictions realistic and interpretable for live match scenarios. The combination of traditional ML and TabNet provides a comparison of classical and deep tabular learning approaches for IPL win probability prediction.

IV. System Deployment Using Streamlit

After training the machine learning models for IPL win prediction, the system was deployed as an interactive web application using **Streamlit**. This allows real-time prediction during a live match scenario while maintaining a lightweight deployment structure suitable for rapid prototyping.

A. Model Serialization

The trained model pipeline was exported as a binary file using Python's pickle. The serialized file stores Preprocessing pipeline (One-Hot Encoding, Scaling) Trained ML classifier (Logistic Regression) Category mappings used during training.

B. Streamlit Interface Design

Streamlit was used to design an interactive interface with dropdowns and number inputs for: Batting and bowling teams, Match city, Current score, Overs bowled, Wickets remaining

This probability is then displayed in a visually appealing format, including color-coded probability bars and summary text.

Fig. 2. User Interface of the IPL Win Prediction System Built Using Streamlit

C. Backend Workflow

The deployed application follows this sequence:

1) User enters match details through the Streamlit UI.
2) Streamlit constructs a DataFrame based on the input.
3) The pre-trained pipeline preprocesses the values.
4) The machine learning model predicts win probability.
5) Results are displayed as: Win probability percentage

D. Deployment Flow Diagram

The following workflow was used for deployment:

Fig. 3. User Interface of the IPL Win Prediction System Built Using Streamlit

This flow is represented visually in the deployment diagram.

V. Result Analysis

This section presents the performance of the three models Logistic Regression, Support Vector Machine (SVM), and TabNet on the held-out test dataset. The evaluation focuses on accuracy, F1-score, recall, ROC-AUC, and confusion matrix interpretation.

A. Overall Model Performance

TABLE II
Model Performance Metrics

Model	Accuracy	F1-Score	Recall	ROC-AUC
Logistic Regression	0.8145	0.8248	0.8324	0.8962
SVM	0.8262	0.8394	0.8680	0.9102
TabNet	0.8434	0.8519	0.8602	0.9317

Summarizes the performance of all models. Among them, TabNet achieved the highest scores across all metrics, demonstrating its ability to learn complex nonlinear relationships

present in ball-by-ball match data. SVM performed competitively, outperforming Logistic Regression but falling short of TabNet. Logistic Regression, while highly interpretable, showed lower predictive capability due to its linearity.

B. Metric-wise Comparison

1) Accuracy: TabNet achieved the highest accuracy (0.8434), followed by SVM (0.8262). Logistic Regression obtained an accuracy of 0.8145. This indicates that TabNet captures in-game match dynamics more effectively than classical machine learning models.

2) F1-Score and Recall: TabNet produced the highest F1-score (0.8519) and recall (0.8602), meaning it correctly identifies winning scenarios while minimizing false negatives. SVM also performed well with an F1-score of 0.8394, whereas Logistic Regression showed comparatively lower performance.

3) ROC-AUC: TabNet achieved the best ROC-AUC score of 0.9317, indicating excellent class separability. SVM followed with 0.9102, and Logistic Regression achieved 0.8962. These results confirm the improved robustness of TabNet when distinguishing between winning and losing game states.

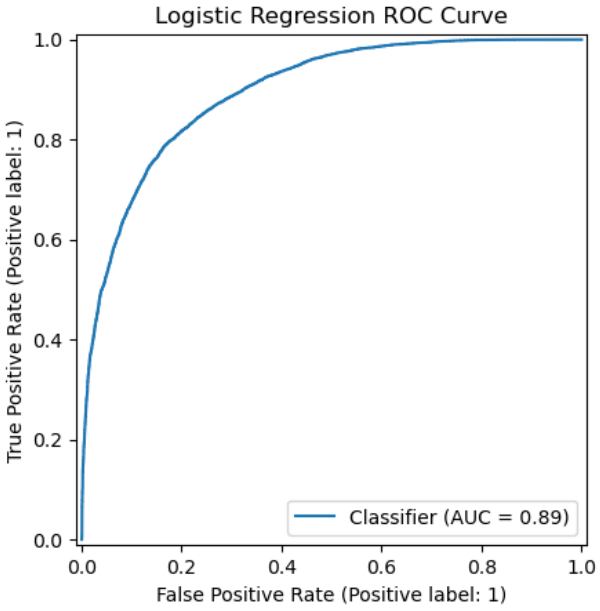

Fig. 4. ROC Curve of Logistic Regression

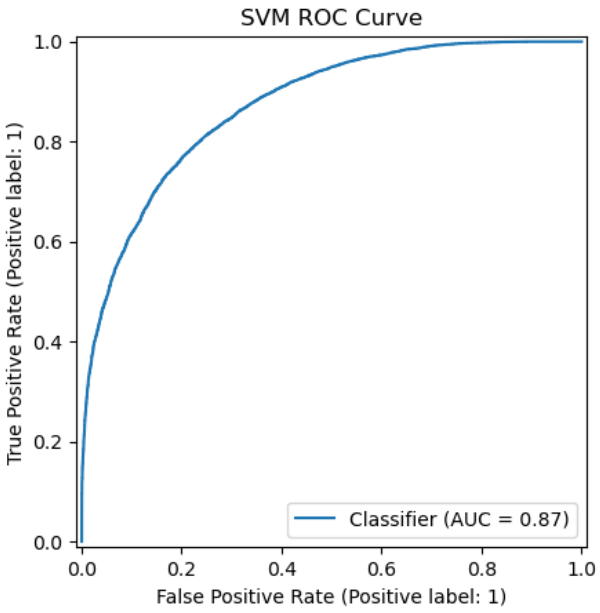

Fig. 5. ROC Curve of SVM

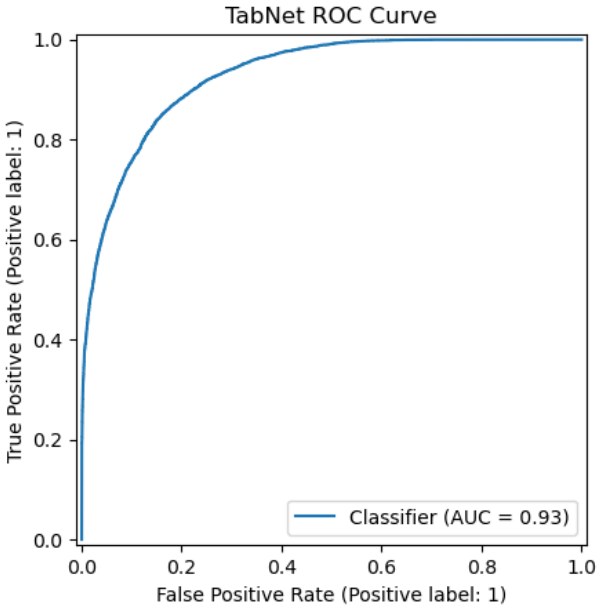

Fig. 6. ROC Curve of TabNet

C. Confusion Matrix Interpretation

Analysis of the confusion matrices for all models shows two trends, TabNet generated fewer false negatives, meaning it rarely predicted a loss when the team actually won. Logistic Regression produced more misclassifications overall due to its linear nature.

D. Comparison with Existing Work

Anushka Tammannappa Maigu (2024) implemented Logistic Regression and Gradient Boosting on historical IPL data but did not consider mid-match predictions. Our models, in contrast, predict win probability dynamically during the second innings [1].

Zohaib et al. (2025) used Random Forest and Logistic Regression with a 70%-30% train-test split for end-of-match predictions. Our approach avoids data leakage by splitting based on matches, ensuring that all deliveries from a match are entirely in either training or test sets [2].

Vishwanath et al. (2025) combined historical and real-time match data with Logistic Regression and Gradient Boosting, but lacked explicit mechanisms to prevent data leakage. Our

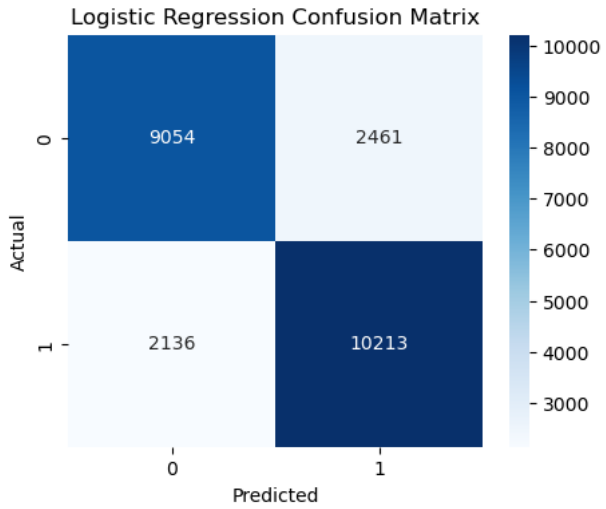

Fig. 7. Confusion Matrix of Logistic Regression

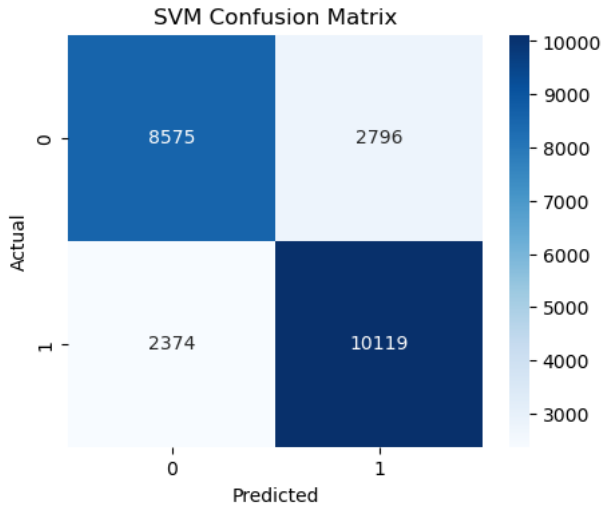

Fig. 8. Confusion Matrix of SVM

models implement safe feature engineering to ensure mid-match predictions remain realistic [3].

Tripathi and Islam (2024) evaluated multiple models including Logistic Regression, Random Forest, SVM, Gradient Boosting, RNN, and HMM, but the system was complex and computationally expensive, limiting real-time usability. Our TabNet model provides a more efficient yet highly accurate alternative for real-time prediction [4].

Overall, TabNet demonstrates superior capability in capturing nonlinear relationships and dynamic match conditions, making it the most effective model for real-time win probability prediction.

VI. Conclusion

This study presents a machine learning–based framework for predicting the win probability of IPL matches during the second innings using real-time, mid-match features. By

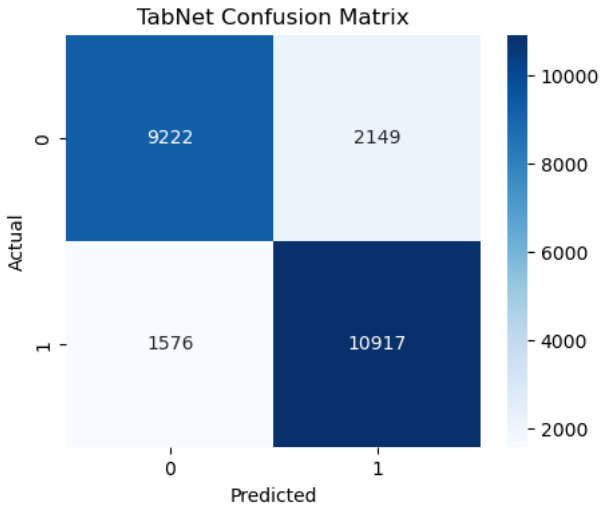

Fig. 9. Confusion Matrix of TabNet

constructing features such as runs left, balls remaining, wickets lost, current run rate, and required run rate, the system ensures that predictions are grounded in the evolving game state.

Three models Logistic Regression, SVM, and TabNet were trained and evaluated. Among them, TabNet achieved the highest performance across all metrics, including accuracy, F1-score, recall, and ROC-AUC, demonstrating its superior ability to capture nonlinear and feature-interaction–based relationships in tabular cricket data. SVM also performed competitively, while Logistic Regression provided a strong interpretable baseline.

Overall, the results validate that win probability can be estimated reliably using machine learning even before the match is completed, enabling informed strategic decision-making for teams, analysts, and broadcasters. Future work may include incorporating player-specific statistics, modeling momentum shifts using time-series or sequence models, and deploying the system for real-time live match prediction.

References

[1] A. T. Maigu, "IPL Win Probability Predictor using Machine Learning Techniques," *International Journal of Research Publication and Reviews*, vol. 5, no. 9, pp. 3146–3151, Sep. 2024.

[2] S. M. Zohaib, N. Sharma, R. Singh, and M. Sonia, "IPL Win Probability Prediction System using Machine Learning Techniques," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Paper ID: IJRASET69906, Apr. 28, 2025, ISSN: 2321-9653.

[3] A. Vishwanath, C. M. Gulzar, J. S. Reddy, E. Manjunath, and M. S. Sukumar, "Analyzing and Estimating IPL Winner Prediction Using Machine Learning," *International Journal of Scientific Research & Engineering Trends*, vol. 11, no. 2, pp. –, Mar.-Apr. 2025, ISSN (Online): 2395-566X.

[4] A. Tripathi, R. Islam, V. Khandor, and V. Murugan, "Prediction of IPL Matches using Machine Learning while Tackling Ambiguity in Results," *Indian Journal of Science and Technology*, vol. 13, no. 38, pp. 4013–4035, 2020.

[5] "Run rate," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Run_rate. [Accessed: 24-Nov-2025].

[6] "Required run rate," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Required_run_rate. [Accessed: 24-Nov-2025].