Cricket Score Prediction in One Day Internationals Using Machine Learning Approach

Kavan Vadodariya
Computer Engineering
Marwadi University
Rajkot, Gujarat, India
kavan.vadodariya120040@marwadiuniversity.ac.in

Jenish Ambaliya
Computer Engineering
Marwadi University
Rajkot, Gujarat, India
jenish.ambaliya118291@marwadiuniversity.ac.in

Het Dhingani
Computer Engineering
Marwadi University
Rajkot, Gujarat, India
het.dhingani120384@marwadiuniversity.ac.in

Urvi Dhamecha
Computer Engineering
Marwadi University
Rajkot, Gujarat, India
urvi.dhamecha@marwadieducation.edu.in

Abstract-- Cricket, particularly in its One Day International (ODI) format, generates extensive data, enabling the application of predictive analytics for score forecasting and match outcome prediction. Accurate score prediction remains a challenging task due to dynamic match conditions such as pitch behavior, wicket falls, and momentum shifts. This study aims to compare the performance of three machine learning algorithms—Linear Regression, Random Forest, and XGBoost—for predicting first-innings scores in ODIs using historical ball-by-ball data. The dataset, sourced from cricket repositories, spans 2003–2017 and includes key features such as overs completed, wickets lost, venue, and runs in the last five overs. After preprocessing and feature encoding, models were trained on 70% of the data and tested on 30%, evaluated using R^2, RMSE, MAE, and MSE metrics. Results indicate that Linear Regression performed inadequately ($R^2 = 0.6191$), Random Forest showed substantial improvement ($R^2 = 0.8230$), and XGBoost outperformed both ($R^2 = 0.8615$), demonstrating its capability to handle non-linear relationships and complex feature interactions. The findings emphasize the superiority of ensemble and boosting techniques for real-time, context-aware predictions in sports analytics, providing valuable insights for teams, broadcasters, and stakeholders.

Keywords— Cricket prediction, One Day International (ODI), Machine learning, Real-time score prediction, Linear Regression, Random Forest, XGBoost, Sports analytics.

I. INTRODUCTION

Cricket is the most popular sport in India and is played all around the world. Cricket, in its One Day International format has gained popularity among fans, with fans closely associating themselves with the scores of each game and performances of each player. Over the time it has become a data-rich sport where analysis plays a vital role in strategy making. Nowadays, most teams are utilizing predictive analysis to get the best possible denouement. Various aspects, including the final scores can now be predicted by certain methods. By using machine learning techniques, we can use predictive analysis to make deductions on player's performance, match outcomes or other strenuous inferences like momentum shifts. One Day Internationals (ODIs) is one of the ideal format for predictions and analysis. By getting access to historical datasets, it has become easier to find meaningful patterns and trends that can help in predicting match outcomes as well as total score.

Predicting the final score is still a challenging task, especially in ODIs where each team plays 50 overs, predicting the final score solely based on run rate is quite challenging. These approaches lack to adapt to match conditions like wickets falling, pitch behavior, etc. As a result, these predictions are mostly inaccurate and are not according to the match context. Therefore, there is a need for a solution that uses historical data and match specific variables to provide accurate score according to real-time conditions. The principal objective of this paper is to compare various machine learning algorithms that can be used in order to formulate score predictions with critical features associated with ODI matches.

The algorithms that are predominantly being used in today's cricket world include Lasso Regression, Random Forest Regression, Gradient Boosting (XGBoost), ARIMA / LSTM (Long Short-Term Memory), KNNs and RNNs. Of these, the methods that are widely being used are Linear Regression, XGBoost and Random Forest method. These models are trained using various match features such as overs completed, wickets lost, venue, wickets lost in last five overs, runs scored in last five overs and more.[1] Each model has been provided with the accurate past match details, helping the system to learn the patterns of variables of overs, runs, wickets and ball-by-ball update in real time, etc. By comparing the performances of these algorithms, it determines the most effective model for accurate and context-aware score prediction. These models are not only used in the field of sports analytics, but also demonstrate how machine learning models can be utilized for real time quick decisional environment as diverse as ODI cricket match. The comparative implementation and use of the models mentioned above further highlights the strengths and limitations, providing exceptional insights for future models to be developed. "Runs prediction and win prediction in cricket is a useful tool for teams, broadcasters, and other stakeholders in the game to make educated decisions and increase their engagement with the sport. This work used machine learning models to predict the runs scored by a team in the first innings and to predict the winner of a cricket match on a real-time basis."[2]

The proposed approach contributes to the field of sports analytics by rule-based prediction systems and data-driven

frameworks. The study chiefly centers on computing the impact of key predictors amalgamating it into the system. Also, this research highlights the use of machine learning models in delivering the real-time predictions which are accurate and meaningful and can support in strategy making in modern cricket.

II. Literature Review

This review highlights key contributions, methodologies, and findings from existing works, with a focus on the application of regression and ensemble learning models such as Random Forest and XGBoost.

More recent research has strengthened these findings. Sherilyn et al. (2023) applied XGBoost to predict T20 scores considering match context (overs, wickets, venue, team strength). They achieved an R^2 of 0.82 and 85% accuracy, highlighting the adaptability of boosting algorithms.[1]

Sudhini et al. (2025) analyzed score prediction in ODIs and T20s using Random Forest and Gradient Boosting. Their study reported high accuracy with R2≈0.93, RMSE of 11.5, and MAE of 8.9, demonstrating the effectiveness of ensemble learning with real match data.[3]

Similarly, Chandru and Prasath (2024) emphasized the importance of preprocessing by applying weighted K-Means clustering before model training. Their findings suggested that feature clustering enhances both prediction accuracy and model interpretability, showing that data preparation plays a vital role alongside algorithm choice.[4]

Comparative studies have also received attention. Thinakaran (2023) compared Random Forest with XGBoost for T20 innings score prediction and found that Random Forest (95.76%) consistently outperformed XGBoost (88.54%), confirming the reliability of bagging-based ensemble methods.[5]

Lamsal and Choudhary (2018) explored multivariate regression and neural networks for predicting IPL match outcomes, achieving ~71.7% accuracy, indicating that early models laid the groundwork but left scope for improvement.[6]

Ahmad et al. (2024) extended this work to ODIs using multiple models, with Random Forest again emerging superior (85.7% accuracy, $R^2 = 80.74$) compared to boosting and neural networks.[7]

Param et al. (2024) combined Random Forest, SVM, and Naïve Bayes with player consistency metrics to predict match outcomes, achieving 89.82% accuracy. Their work introduced player-specific features that enhanced prediction beyond traditional team and match-level statistics.[8]

Similarly, Mozar et al. (2022) used Random Forest and Lasso Regression for live score and winner prediction in IPL, demonstrating stable results and deployment potential through a GUI system.[9]

Table1: Literature Review Table

Author(s) & Year	Methodology Used	Key Findings	Limitations
Kevin et al. (2023)	Linear Regression, Decision Tree	Achieved reasonable accuracy for T20 score prediction	Limited to small dataset; lacked ensemble models
Satwani et al. (2024)	Random Forest, Gradient Boosting	Provided real-time predictions for runs and match outcome	Performance drops in rapidly changing match conditions
Sudhini et al. (2025)	Random Forest, XGBoost	High accuracy in ODI & T20 score prediction ($R^2 \approx 0.93$)	Did not address overfitting and dynamic factors like pitch
Chandru & Prasath (2024)	Weighted K-Means for normalization	Improved prediction stability by data normalization	Normalization alone not sufficient for model accuracy
Thinakaran (2023)	Random Forest vs XGBoost	XGBoost outperformed Random Forest in score prediction	Focused only on T20; lacks cross-format validation
Lamsal & Choudhary (2018)	Logistic Regression, Naïve Bayes	Good results for IPL outcome prediction	Older model, lacks advanced ensemble techniques
Fayyaz et al. (2024)	SVM, Neural Networks	Demonstrated strong performance for score estimation	Model complexity increases computation time
Dalal et al. (2024)	Ensemble Learning, Regression	Enhanced prediction accuracy with combined models	Limited feature set; lacks weather and pitch data
Mozar et al. (2022)	Decision Tree, Random Forest	Effective for predicting score and win probability	Not optimized for real-time dynamic scenarios

Overall, the literature confirms the growing role of machine learning in cricket analytics. While early models demonstrated feasibility, recent advancements using ensemble learning and context-aware features have significantly improved prediction reliability. However, opportunities remain to integrate deep learning architectures

and real-time streaming data for even more precise and adaptive systems.

III. Algorithms

A. Linear Regression

Linear Regression is the simplest and most used algorithms in machine learning. It works on linear relationship, meaning a target can be predicted as a weighted sum of the input features plus a constant bias term. It finds the best straight fit line between independent variable and dependent variable. It can serve as a strong baseline model and provide fast predictions.

In cricket score prediction, Linear Regression uses variables such as overs completed, runs scored, wickets lost, venue, runs scored in last five overs and wickets lost in last five overs to predict the final score.[1] Because of its simplicity it is easy to implement. Through this analyst, broadcasters and team coach can quickly understand the factors and easily get the estimated final score.

However, Linear Regression assumes that relation between its target and features is purely linear, which might not capture the complexity in cricket matches, where there can be sudden batting collapse or changes in pitch behavior. But still, we can use it for quick predictions. Overall, the efficiency, low cost and simplicity make Linear Regression a valuable starting point in score prediction.

B. Random Forest

Random Forest is an ensemble learning algorithm, it constructs many decision trees during training time and outputs the average prediction (for regression) from all trees or majority vote (pick the class which is most trees occurs) for classification. This is achieved using two separate randomness processes: first, each tree is trained on a random selection of features and second, it is trained on a bootstrapped subsample of the data. This technique helps in controlling overfitting and improves generalization.

Random Forest can take into consideration several match-level variables at the same time like number of overs completed, wickets lost, rate of runs scored in last 5 overs and various venue-specific statistics for predicting cricket scores. Each tree may capture distinct things about the game, and when combined they result in a strong, stable prediction.

Because Random Forest can learn in live match conditions, it was the methodology of choice to be used for real-time, contextual score estimates. But it can suffer slowdowns due to more complex models, and computational cost may become substantial on very large datasets.

C. XGBoost

XGBoost is a powerful machine learning algorithm on the Gradient boosting framework. It sequentially trains trees, and each new tree traces the errors made by previous ones. Its speed, scalability and accuracy come to a large extent from some of the advanced regularization techniques used in XGBoost.

For example, if trying to predict cricket scores XGBoost would be able to capture complex non-linear relationships between features (e.g. interaction of pitch conditions, overs remaining, wickets) efficiently. Unlike Linear Regression, it does not assume a straight-line relationship, allowing it to adapt to sudden momentum changes in matches.

The algorithm minimises a loss function over multiple trees created following the residual errors left by the prior trees. In addition, XGBoost provides the flexibility to handle many data formats as well as missing data, parallel processing, and optimization for loss functions. Because of its strength in handling large amounts of historical data, and being able to detect tiny patterns while adjusting to match-by-match conditions, XGBoost is one the best algorithms for predictive tasks in sports analytics.

IV. Dataset Description

A. Source of Data:

The ODI dataset was collected from a cricket data repository (most likely Kaggle), containing ball-by-ball information for One Day International (ODI) matches.

B. Size and Time Span:

The dataset consists of **350,899 ball-by-ball records** from approximately **2006 to 2017**. It includes **15 features** related to match details, batting, bowling, and cumulative statistics.

1) Number of Rows: 350,899
2) Number of Columns: 15
3) Time Span: 3 January 2003 to 10 July 2017

C. Key Features:

Table2: Dataset Key attributes

Feature	Description	Example
mid	Unique Match ID	1
date	Date of match	2006-06-13
venue	Match stadium	Civil Service Cricket Club
bat_team	Current batting team	England
bowl_team	Current bowling team	Ireland
batsman	Striker facing the ball	ME Trescothick
bowler	Bowler delivering the ball	DT Johnston
runs	Runs scored on delivery	4
wickets	Wickets fallen so far	0
overs	Current over with balls	34.3
runs_last_5	Runs in last 5 overs	42
wickets_last_5	Wickets in last 5 overs	1
striker	Runs by striker	12
non-striker	Runs by non-striker	8
total	Total team score	301

V. METHODOLOGY

The methodology is divided into six stages: Dataset Preparation, Feature Encoding, Train-Test Split, Model Development, Model Evaluation, and Comparative Analysis. Each stage plays a vital role in building a systematic framework for predicting ODI scores. The inclusion of mathematical formulas ensures clarity on how models and evaluation metrics were implemented.

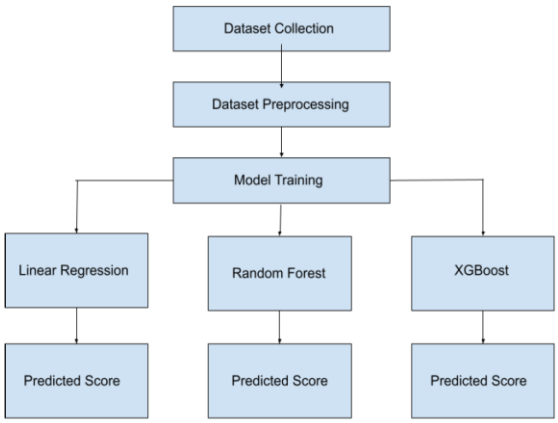

Fig1: Data flow diagram

A. Data Preparation

The dataset is important for this study, consisting of ODI match records with features such as runs scored, wickets lost, overs completed, venue, batting team, bowling team, and statistics from the last five overs. The target variable is the final total score of the innings.

Formally, the dataset can be represented as:

$$D= \{(X1, y1), (X2, y2), \ldots, (Xn, yn)\}$$

where Xi represents the feature vector and yi the corresponding total score.

Data preprocessing involved handling missing values, irrelevant columns. Features directly influencing the score were retained, such as wickets lost and runs scored, while redundant attributes were removed. This ensures a clean and structured dataset. By refining the data, the methodology created a reliable foundation for accurate model training and testing.

B. Feature Encoding

Cricket datasets have both numeric and categorical features. Numerical features like Runs scored, Wickets lost (0–10 wickets), and overs are linear in nature, which can be directly fed to machine learning model but Categorical Features such as Venue, Batting Team and Bowling Team needs some sort of transformation (one-hot encoding). They are pivotal because context is provided as a result of these attributes; one can be assured that certain venues do have batting paradises and some teams tend to play better (or worse) based on opposition.

C. Train-Test Split

Cross-Validation is one of the most important steps in any predictive modeling task where we train and build our model on one subset of the dataset and test it on yet another independent subset. In a general sense, this is called overfitting where model works very well in training data but fails miserably on unseen cases. The dataset was randomly partitioned into training (70%) and testing (30%) sets in this study.

Formally, the dataset is split as:

$$D=Dtrain \cup Dtest, Dtrain \cap Dtest= \emptyset$$

with proportions:

$$|Dtrain|=0.7n, |Dtest|=0.3n$$

where n represents the total number of match instances.

Here, Dtrain is used to learn patterns and train the model. While, Dtest evaluates how well the models works on unseen matches. The choice of having 70:30 is because it provides sufficiently large training set to capture diverse match scenario while keeping enough test data to reliably assess model performance. This split ensures that the model works perfectly and through testing it can give accurate results.

D. Model Development

The core of this methodology lies training three machine learning models- **Linear Regression**, **Random Forest** and **XGBoost** on the prepared dataset:

1) Linear Regression

Linear Regression works on linear relationship between independent variable and the target:

$$\hat{y}=\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_pX_p$$

where ŷ is the predicted score, xi are input features (runs, wickets, etc.), and βi are coefficients estimated using least squares. The model minimizes the **Mean Squared Error (MSE):**

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \quad \backslash(yi-\hat{y}i\backslash)2$$

2) Random Forest

Random Forest builds an **ensemble of decision trees** using bootstrap samples. Each tree makes a prediction, and the final output is their average:

$$\hat{y} = \sum_{j=1}^{k} Tj\ (X)$$

where Tj(X) is the prediction of the jth tree. Random Forest reduces overfitting.

3) XGBoost

XGBoost uses **gradient boosting**, adding trees sequentially to correct previous errors. At stage t:

$$\hat{Y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$$

where η is the learning rate and ft(xi) is the new tree.

By training these three models on identical datasets, the methodology ensures a fair comparison of their predictive capabilities in real-time ODI score prediction.

E. Model Evaluation

Model evaluation is important for the reliability and accuracy of predictions. In this study we used four metrics: **R^2 score, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Squared Error (MSE)**

R^2 Score:

$$R^2\ 1 - \frac{\sum_{i=1}^{n} \backslash(yi-\hat{y}i\backslash)2}{\sum_{i=1}^{n} \backslash(yi-\bar{y}i\backslash)2}$$

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \backslash(yi-\hat{y}i\backslash)2}$$

Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |yi-\hat{y}i|$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \backslash(yi-\hat{y}i\backslash)2$$

Here R^2 captures overall explanatory power and RMSE provides sense of error margin.

R^2 values range from 0 to 1 (0% to 100%). Values between 0 and 1 indicate the proportion of variance explained by the model. For example, an R^2 of 0.85 suggests that 85% of the variance in the dependent variable is explained by the model's independent variables.

A lower RMSE indicates a better model fit, meaning the model's predictions are closer to the actual values. A perfect model would have a RMSE value of 0.

MAE shows the average absolute error and difference between predicted and actual values, indicating how far the predicted values are from actual values. A lower MAE means better accuracy.

While MSE measures the average if squared differences between actual values and predicted values, giving more weight to larger errors. A lower MSE means better model fit and a perfect model if the value is closer to or is zero.

VI. Results

A. Model Performance Evaluation:

Among the evaluated models, Linear Regression performed the weakest due to its limited ability to capture non-linear data patterns, reflected in low accuracy and high errors. Random Forest improved predictive capability with stronger generalization and reduced errors. XGBoost achieved the best performance, delivering the highest accuracy and lowest error values, confirming its effectiveness for complex predictive tasks.

Table2: Model Evaluation

Model	R^2 Score	RMSE	MAE	MSE
Linear Regression	0.6191	38.51	29.11	1483.14
Random Forest	0.8230	26.26	17.24	689.38
XGBoost	0.8615	23.22	16.59	539.35

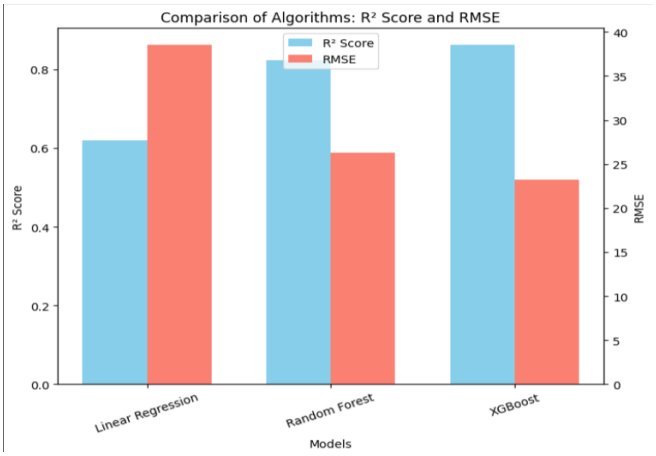

Fig2: Comparison of Algorithms- R^2 score and RMSE

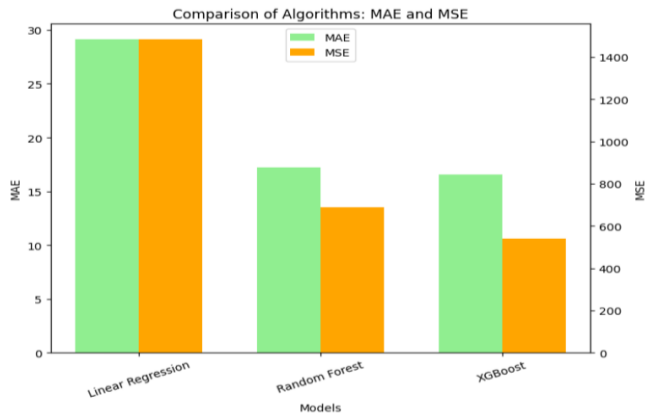

Fig3: Comparison of Algorithms- MAE and MSE

B. Analysis of Results

1) Linear Regression:

Linear Regression demonstrated the weakest predictive capability, with an R^2 score of 0.6191 and the highest associated error values. These results suggest that the model is inadequate for capturing complex and non-linear patterns present in the dataset. Its reliance on linear assumptions limits its applicability for real-world predictive tasks involving dynamic and heterogeneous data.

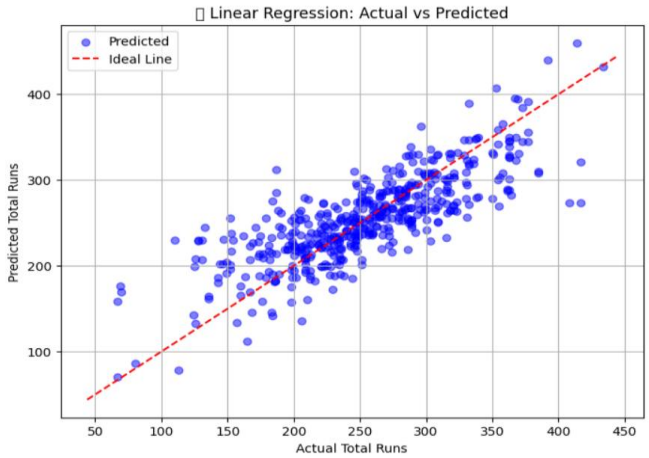

Fig4: Graphical analysis of Linear Regression (Actual vs Predicted Score)

2) Random Forest:

Random Forest achieved a stronger performance with an R^2 score of 0.8230 and substantially lower error measures. The ensemble nature of the algorithm enabled it to effectively address non-linearities and interactions within the dataset. These outcomes highlight Random Forest's suitability for moderately complex data, offering improved accuracy and robustness compared to traditional regression techniques.

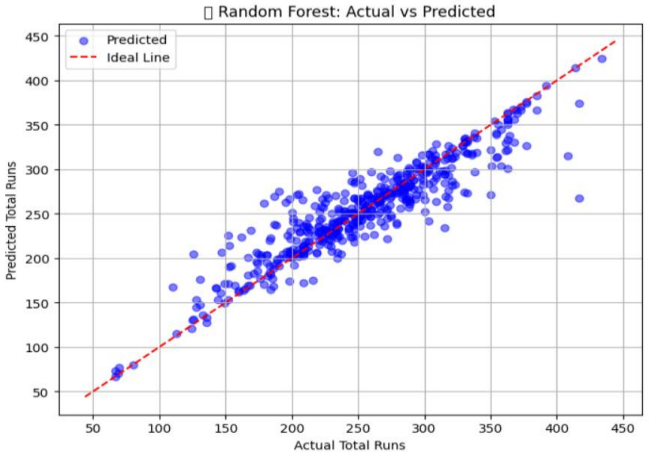

Fig5: Graphical analysis of Random Fores (Actual vs Predicted Score)

3) XGBoost:

XGBoost yielded the most accurate predictions, attaining the highest R^2 score of 0.8615 and the lowest error values across all evaluation metrics. Its gradient boosting framework, which iteratively minimizes residual errors, proved highly effective in handling data complexity. These findings confirm XGBoost's superior capability for real-world predictive modeling, particularly in scenarios demanding high accuracy and reliability.

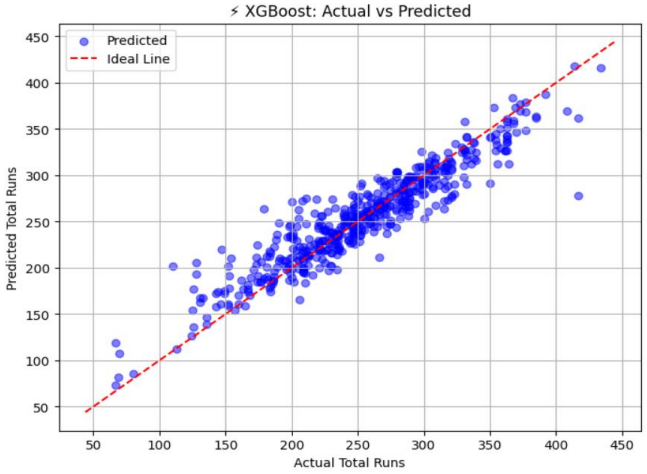

Fig6: Graphical analysis of XGBoost (Actual vs Predicted Score)

VII. Conclusion

This comparison study shows the importance of selecting models for predictive analysis when addressing huge datasets with complex and non-linear relationships. Linear Regression widely regarded for its simplicity, was found to be inadequate int this context, as it failed to capture deeper patterns in data and it also failed when there was sudden change in momentum or sudden batting collapse. Its limitation reaffirms the challenges of relying solely on regression techniques for real world applications where non linearity is inherent.

In contrast, Random Forest showed notable improvement compared to Linear Regression in predictive accuracy, reflecting the strength of ensemble learning by using the aggregation of multiple decision trees. Its capacity to generalize than linear model highlights its suitability for more complex datasets. Despite forming performing this well Random Forest was outperformed by a more advanced boosting technique.

XGBoost emerged as the most accurate and effective model in this comparison, achieving superior predictive accuracy and also maintaining lower error margins. Its iterative gradient boosting mechanism, which continuously refines predictions by correcting large errors and it proved highly effective in handling complex and large dataset. The results showed why XGBoost is best performing algorithm than others while maintaining its reputation as a state-of-the-art algorithm for predictive modeling.

From this evaluation, it is evident that while simpler models provide interpretability, advanced ensemble approaches such as XGBoost offer significantly greater reliability and accuracy. For real-world predictive tasks, particularly those involving complex and dynamic data, gradient boosting frameworks stand out as the most practical and impactful choice.

VIII. REFERENCES

[1] S. Kevin, B. Yadav, A. K. Pandey, and G. Rajbhar, "T20 Cricket Score Prediction Using Machine Learning," *IRE Journals*, vol. 7, no. 6, pp. 49–53, Dec. 2023.

[2] A. Satwani, K. Coutinho, N. John, and J. A. Arul Jothi, "Live Cricket Predictions for Runs and Win Using Machine Learning," in *Proc. IEEE Int. Symp.*, 2024.

[3] T. R. Sudhini, et al., "Analysis of Cricket Score Predictor using Machine Learning," *SSRN*, 2025.

[4] M. Chandru and S. Prasath, "Dataset Normalization in Cricket Score Prediction Using Weighted K-Means Clustering," *Int. J. of Innovative Studies in Applied Engineering (IJISAE)*, vol. 12, no. 21s, 2024.

[5] K. Thinakaran, "Predicting T20 Match Innings Score Using Novel Random Forest Compared With XGBoost," *Saveetha School of Engineering*, 2023.

[6] R. Lamsal and A. Choudhary, "Predicting Outcome of IPL Matches Using Machine Learning," *arXiv preprint*, 2018.

[7] A. Fayyaz, A. Zafar, M. Wasim, S. S. Abbas, A. Ahad, A. A. N. Godinho, P. J. Coelho, and I. M. Pires, "Cricket Score Prediction using Machine Learning Techniques," 2024.

[8] P. Dalal, H. Shah, T. Kanariya, and D. Joshi, "Cricket Match Analytics and Prediction using Machine Learning," 2024.

[9] O. Mozar, S. More, S. Nagare, and N. Pathak, "Cricket Score and Winning Prediction," 2022.