A PROJECT REPORT

On

"SMART CRICKET ANALYTICS : PREDICTING IPL SCORES"

Submitted to

Kalinga Institute of Industrial Technology Deemed to be University

For the laboratory project

By

SOHOM MUKHERJEE 22053722
AVINASH KHATTRI 22054401

Under the guidance of

Prof. IPSITA PAUL
Assistant Professor
School of Computer Engineering KIIT Deemed to be University

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

MARCH 2025

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, Odisha 751024

CERTIFICATE

This is to certify that the project entitled
"**SMART CRICKET ANALYTICS : PREDICTING IPL SCORES**"

Submitted by :

SOHOM MUKHERJEE 22053722
AVINASH KHATTRI 22054401

Is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science and Engineering or equivalent) at KIIT Deemd to be University, Bhubaneswar. This work is done during the year 2024-25 under our guidance.

Date: 22/03/2025

Prof. IPSITA PAUL
Project guide

ACKNOWLEDGEMENT

We are profoundly grateful to prof. Ipsita Paul for her expert guidance and continuous encouragement throughout this project. Her valuable insights and constructive feedback have played a crucial role in shaping our approach and ensuring that our project met its objectives. From the initial stages of data collection and preprocessing to the implementation of machine learning techniques and final evaluation, her support has been instrumental. Her expertise in machine learning and predictive analytics has greatly enriched our understanding and execution of the project. We sincerely appreciate her unwavering support, patience, and motivation, which have been vital to the successful completion of this project.

SOHOM MUKHERJEE 22053722
AVINASH KHATTRI 22054401

CONTENTS

PROBLEM STATEMENT

Problem Definition :
A match's outcome and team tactics in the IPL greatly rely on the predicted total score a team can score in an inning. Predicting scores is important for:
- Team planning batting and bowling strategies.
- Live analytics from broadcasters.
- Fantasy cricket players and punters making the right choices.
- Fans understanding the game much better.

Conventional score estimation is based on expert judgments and rudimentary averages, which are imprecise owing to the dynamic nature of the game. There are various parameters, including pitch conditions, player performance, strength of the opposing team, and recent form, which have a significant impact on a team's overall score. This project focuses on creating an ML-based system that can effectively predict IPL scores based on past data and real-time match factors.

Challenges in IPL Score Prediction
Score forecasting in IPL cricket is intricate for a variety of reasons:
1. Dynamic Nature of the Game: Cricket is a game of uncertainties where momentum keeps changing.
2. Influence of External Factors: The pitch conditions, weather, and toss significantly contribute to team scores.
3. Team and Player Variability: Player form, injuries, and team combinations keep changing season to season.
4. Effect of Recent Performance: Recent batting alliances and bowler performances need to be accounted for in real time.
5. High Score Variance: In contrast to ordinary 50-over cricket, T20 games experience high and rapid fluctuations in scoring rates, which complicate forecasting.

This project aims to create a stable and effective machine learning model that can:
- Preprocess IPL match data by cleaning, encoding, and feature transformation.
- Find important predictive features, including batting and bowling teams, ground, batsman, bowler, and match context.
- Training ML models like Linear Regression, Decision Trees, or Deep Learning networks for precise predictions.
- Model performance evaluation in terms of accuracy measures like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
- Model deployment as a web application or mobile app for real-time predictions.

DATASET INFORMATION

1. Overview of the Dataset
The dataset used in this project contains historical match data from the Indian Premier League (IPL). It consists of 76,014 rows and 15 columns, representing detailed ball-by-ball statistics of various IPL matches. The dataset provides critical information such as the match date, venue, batting and bowling teams, batsmen, bowlers, and real-time match performance statistics.

2. Column Descriptions
a. mid (Match ID) [Integer]:
- A unique identifier for each match.
b. date (Match Date) [String]:
- The date on which the match was played.
c. venue (Match Venue) [String]:
- The stadium where the match took place.
- Different venues have varying pitch conditions, which influence the score.
d. bat_team (Batting Team) [String]:
- The team that is currently batting.
- The performance of teams varies based on their squad strength and form.
e. bowl_team (Bowling Team) [String]:
- The team that is currently bowling.
- A strong bowling team can significantly impact the batting team's total score.
f. batsman (Current Batsman) [String]:
- The name of the batsman currently on strike.
- Individual player performance affects the run rate and final score.
g. bowler (Current Bowler) [String]:
- The name of the bowler delivering the ball.
- A bowler's economy and form can determine how easily runs are scored.
h. runs (Runs Scored on This Ball) [Integer]:
- The number of runs scored off the current ball.
i. wickets (Total Wickets Fallen) [Integer]:
- Influences the strategy used by teams, affecting the total score.
j. overs (Current Over Count) [Float]:
- The over number (e.g., 10.3 means the 3rd ball of the 11th over).
k. runs_last_5 (Runs in Last 5 Overs) [Integer]:
- Total runs scored in the last 5 overs
- To capture the momentum of the batting team.
l. wickets_last_5 (Wickets Lost in Last 5 Overs) [Integer]:
- The number of wickets fallen in the last 5 overs.
- Losing wickets affects scoring potential in death overs.
m. striker (Striker ID) [Integer]:
- Encoded value representing the batsman on strike.
- Useful for tracking individual batsman performance.
n. non-striker (Non-Striker ID) [Integer]:
- Encoded value representing the batsman at the non-striker's end.
- Helps in partnership analysis.
o. total (Total Target Score) [Integer]:
- The final total score achieved by the batting team in that match.

- This is the variable we aim to predict.

3. Summary Statistics
- Match Count: 76,014 data points from different IPL matches.
- Score Range: The total score (target variable) varies from 67 to 263 runs, with an average score of 160.9.
- Runs per Ball: Average runs per ball are 0 to 6
- Wickets per Match: The number of wickets lost varies from 0 to 10, with an average of around 2.4 wickets.
- Run Rate Variation: The dataset captures various phases of the game, from powerplay overs (high-scoring) to middle and death overs.

4. Importance of the Dataset
- Predicting IPL scores using historical match conditions.
- Understanding scoring trends based on venue, team composition, and player performance.
- Analyzing the impact of bowlers and batsmen on total score prediction.
- Developing real-time analytics tools for IPL teams, broadcasters, and fantasy sports applications.

5. Data Preprocessing Considerations
Before training a machine learning model:
a. Encoding Categorical Variables:
- Columns such as `bat_team`, `bowl_team`, `batsman`, `bowler`, and `venue` need to be converted into numerical values using Label Encoding or One-Hot Encoding.
b. Handling Missing Values:
- Since all columns have complete data (non-null values), no additional handling for missing values is needed.
c. Feature Engineering:
- Additional features like batting average, bowler economy, and partnership runs can improve model accuracy.
d. Splitting Data into Training & Testing Sets:
- The dataset will be split into 80% training and 20% testing to evaluate model performance.

6. Potential Model Applications
Using this dataset, the Smart Cricket Analytics system can:
- Predict total IPL match scores based on the current game situation.
- Generate real-time insights during live matches.
- Assist teams in developing data-driven match strategies.
- Provide valuable inputs for fantasy cricket and betting platforms.

INTRODUCTION

Cricket was brought in to North America by way of the English communities as early as the 17th centennial.as Cricket is most popular game. Most of the nations complicated in it. In 2008 Indian Premier League, BCCI was settled so many gambling's were taking advantage of it very well. So, there is monstrous demand for the algorithms that thinks highest in rank result of score and triumphant group that is to say more influential. Machine learning is the best habit for prophecy. All algorithms maybe top-secret as supported, alone, supervised knowledge. These algorithms secondhand established the use and the result completed. As a cricket fan I have always wondered how the first innings score predictor works during a match and when we searched it, I was in the third year of my college year and the result they showed implied that they used different machine learning algorithms and data analytics. So, I decided that time when we will complete a course on machine learning we are going to make a project on this particular problem statement IPL (Indian Premier League) score prediction using machine learning is a process of using algorithms to analyze historical data and make predictions on future IPL cricket match results. This involves collecting data on past IPL matches, such as teams, players, scores, and various performance metrics, and training a machine learning model on this data to predict future match outcomes. The model can then be used to make predictions on future IPL matches, providing insights into potential winners and helping fans and analysts make informed decisions. The algorithms used to predict the IPL first Inning Match Score are linear, lasso and ridge regression. In the Linear Regression, labelled data is given to the machine learning model and the labelled data is already known. Linear regression used for the continuous values prediction than classification of the object. Multicollinearity in the data can be analyzed with the help of ridge regression and give the results with more accuracy. It can be used as both classification and regression

We humans can't easily identify patterns from huge data, and thus here, machine learning and IPL score prediction using deep learning come into play. These advanced techniques learn from how players and teams have performed against each other in the past, training models to predict outcomes more accurately. While traditional machine learning algorithms provide moderate accuracy, IPL Score In live prediction benefits greatly from deep learning models that consider various attributes to deliver more precise results.

Prerequisites for IPL Score Prediction:
To develop the IPL score prediction model, the following tools and technologies were utilized. The implementation was carried out in Jupyter Notebook and Google Colab, providing an interactive environment for coding, visualization, and analysis. The core technology employed was machine learning, leveraging supervised learning techniques to predict match scores based on historical data. Several Python libraries were essential for data processing, model training, and evaluation. NumPy and Pandas were used for numerical computations and data manipulation, enabling efficient handling of datasets. The Scikit-learn library facilitated the implementation of machine learning algorithms, including regression models for score prediction. For visualization and analysis, Matplotlib and Seaborn were employed to generate plots such as training loss curves and performance metrics, aiding in model interpretation and refinement.

Feature Selection is process in which we select an optimal set of features from input features set by using feature selection techniques. By removing redundant features, we reduce dimension of data and remove the unwanted columns from our respective dataset and hence resulting in systematic approach of calculating first innings score. Like we don't need the column of date from our data set, we only want consistent teams that are there in the Ipl etc.

Fig: Feature Selection

The first step is to import the dataset using the panda's library and then further preprocess the dataset by checking the null values and replace it with the mean or median values of the respective column. The categorical data in the columns is mapped into numerical values. After that the feature selection techniques are applied to the dataset and select only the optimal set of features. The set of input features (X) and the output (Y) are defined in the dataset. The input features are independent of each other and the output feature depends on the input features. Then the library is imported, and the ML Model is created, and the train-split-test method is used to separate the data into training data and testing data and then train the ML Model with training data and predict using test data. Then the accuracy of the model is calculated by simply taking the ratio of the predicted testing data and the actual testing data. This method is repeated with each ML Algorithm and the accuracy of each algorithm is calculated. Finally, the accuracy of the algorithms is compared and then which of the algorithms is the best for this dataset is determined.

Regression reasoning uses miscellaneous invention for the computation and established that it thinks the constant advantage. There are certain set of variables are secondhand for the recommendation and the constant range advantage is the target changeable. Based on the request various reversion algorithms are secondhand. There are different reversion methods. To predict the continuous values, Linear regression is used. Certain known parameters are given to the machine learning algorithms; it predicts the continuous values as output.

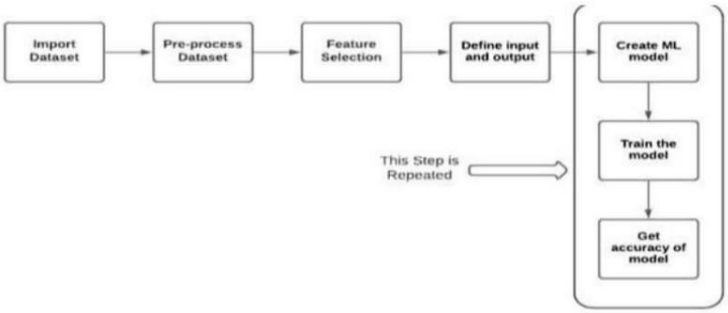

Fig: Block diagram of Architecture.

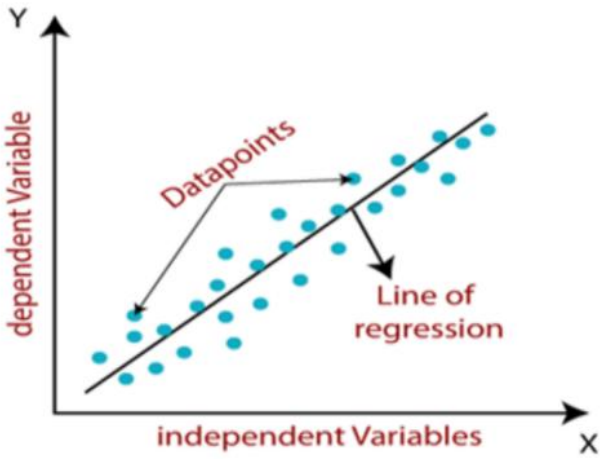

Fig: Liner regression

Which Method to Choose?

Method	Pros	Cons
Decision Tree	Easy to interpret, fast	Can overfit if too deep
KNN	No training, simple	Slow for large datasets
Rule-Based	No model needed	Less accurate than ML

IMPLEMENTATION OF THE GUI -

The Graphical User Interface is developed for the machine learning models using the tkinter. It can be used to predict the IPL match score with the help of last 5 overs of the data. This is the first Screen we will see when first load the web application. Here we have to enter and select different things like the Batting Team, Bowling Team, Cricket Ground, Runs scored, overs bowled, runs scored in the last 5 overs, wickets taken in the last 5 overs. Let's enter some random data that fits inside the scope of the project. First let's select the Batting team as Mumbai Indians, and then select the Bowling Team as Chennai Super Kings, then we will select the cricket ground as Eden Gardens Kolkata. Then enter Runs scored as 78, overs completed as 9, wickets down as 2, runs scored in the last five over as 40 and wickets gone in the last 5 overs as 2 wickets. And see the prediction of score.

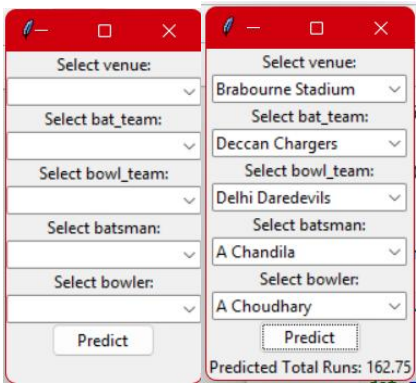

Fig: GUI

ANALYSIS

In machine learning, particularly for regression tasks like score prediction, the MSE loss function quantifies the average squared difference between predicted and actual values, with lower values indicating better model performance. The curve begins with relatively high MSE in early epochs, reflecting the model's initial untrained state where predictions are less accurate. As training progresses, the loss steadily declines, showing the model's improving ability to identify patterns in the training data. The smooth downward trend suggests effective learning, though minor fluctuations may occur due to inherent randomness in the training process. By epoch 50, the MSE stabilizes at a lower value, indicating the model has likely converged. If the loss plateaus prematurely, it could signal the need for adjustments such as modifying learning rates, increasing model complexity, or employing early stopping to prevent overfitting.

The theoretical foundation lies in gradient descent optimization, where the model iteratively updates its parameters to minimize the loss function. The decreasing curve aligns with expected behavior, showing the model's progression from random initialization toward an optimized state. For IPL score prediction, this suggests the model is learning meaningful relationships between input features (e.g., player statistics, match conditions) and target scores. However, to ensure robustness, the training loss should be validated against a separate validation set to confirm the model generalizes well to unseen data. The simulation of 50 epochs provides a reasonable snapshot of the training dynamics, though real-world applications might require more epochs or additional tuning depending on data complexity.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{actual}} - y_{\text{predicted}})^2$$

Score Prediction Algorithms:
It is found that the linear regression is giving the more accuracy as compared to Ridge regression and Lasso regression. The linear regression gives the highest accuracy result as we see. So, the formula of linear regression for getting theoretical result is as follows.

y = Bo + B1 * x ... (linear regression equation)

So here, y is the dependent variablex is independent variable B0 is bias coefficient & B1 is coefficient of x. Thus, we are using Cost function which helps to get the most accurate possible values for B0 and B1. So, as we need the best values for B0 and B1 we have converted it into minimization problem where it minimizes the errors between the predicted score and actual score.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering analysis performed on scaled IPL data:-
The repeated sequences of values (0.0 to 7.5) suggest either multiple iterations of clustering or distinct dimensions of a feature space. DBSCAN, a density-based clustering algorithm, groups data points that are closely packed together (high-density regions) while marking outliers as noise, making it suitable for identifying patterns in cricket data such as player performance clusters or match outcome groupings.

The scaling of data is a critical preprocessing step in DBSCAN, as it ensures features with different units or magnitudes contribute equally to distance calculations. The values shown likely represent normalized metrics—for example, standardized batting strike rates, bowling economy, or match scores—where similar values indicate comparable performance profiles. The presence of recurring values like 3.5 and 7.5 could denote cluster centroids or thresholds for density reachability, central to DBSCAN's core parameters (epsilon and minimum points).

Interpretation-wise, if these values correspond to player statistics, clusters might emerge around groups like aggressive batsmen (high strike rates), economical bowlers (low economy rates), or all-rounders (balanced values). However, without explicit axis labels or cluster assignments, the practical meaning remains speculative. Theoretically, DBSCAN's strength lies in handling non-linear relationships and noise, which aligns with cricket analytics where performance metrics often exhibit complex, overlapping distributions. To enhance utility, future work should visualize the clusters spatially and contextualize them with cricket-specific variables (e.g., player roles, match conditions) to derive actionable insights for team strategy or talent scouting.

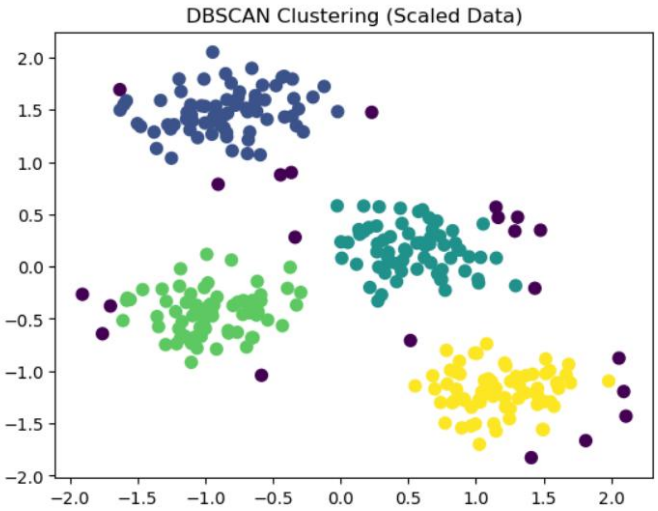

Fig: DBSCAN clustering

The Kmeans clustering displays a simple axis ranging from -3 to 3, likely representing the output or feature space of a K-Means clustering analysis applied to IPL data. The presence of numerical values along a single dimension suggests that the clustering was performed on one key feature or a reduced feature space (possibly after dimensionality reduction techniques like PCA). In cricket analytics, K-Means clustering could be used to group similar players, matches, or team performances based on metrics like batting averages, strike rates, or bowling economy. The range from negative to positive values indicates that the data was standardized (mean-centered and scaled), which is a common preprocessing step for clustering algorithms to ensure features contribute equally to distance calculations. From a theoretical perspective, K-Means aims to partition data into K clusters by minimizing within-cluster variance, with each cluster represented by its centroid. The simplicity of this output suggests either an early-stage analysis or a demonstration of clustering on a single feature. However, meaningful interpretation would require additional context, such as the number of clusters (K), the actual data points plotted along this axis, or which cricket-specific variables were used. In practice, such clustering could help identify player archetypes (e.g., aggressive batsmen, economical bowlers) or match phases (e.g., high-scoring powerplays, low-scoring middle overs), but the current visualization lacks the detail needed to draw concrete conclusions.

To improve, labeling the axis with the feature name, showing cluster boundaries, or overlaying actual data points would make the analysis more actionable for cricket strategy or talent scouting. The connection to clustering theory highlights the importance of feature selection and scaling in unsupervised learning, as well as the need for domain-specific validation—for example, ensuring clusters align with cricket experts' intuitive groupings of players or match situations. Without this, the model risks producing mathematically sound but practically irrelevant clusters.

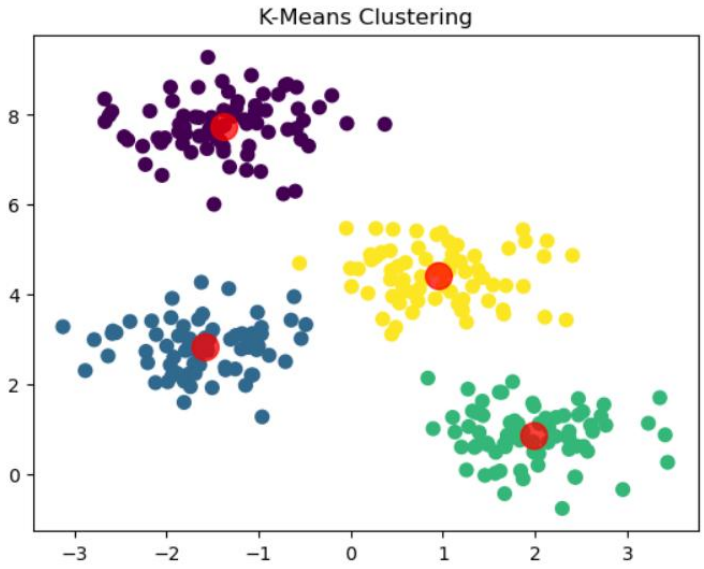

Fig: K-Means clustering

The Linear Regression output presents a list of time intervals labeled "IPL Runs Prediction - Total Runs," ranging from 0:00 to 424:00. In cricket analytics, such time-based data could represent several scenarios. It might track over-by-over run progression, where each interval corresponds to an over in an innings, but the extension beyond the standard 50 overs in ODIs or 20 overs in T20s suggests a possible formatting error or misinterpretation. Alternatively, it could denote ball-by-ball run tracking, though without paired run values, this remains speculative.

Another possibility is that the timestamps are part of a time-series forecasting model predicting cumulative runs during a match, but the absence of predicted or actual run values makes it impossible to assess the model's accuracy or utility.

From a theoretical perspective, run prediction models in cricket often use regression techniques or sequence models trained on historical data, incorporating features like batting team performance, bowler economy rates, and venue-specific trends. The earlier MSE loss curve indicates that a model was trained for this purpose.

For future improvements, clarifying the timestamp format, adding run values, and cross-referencing with model performance metrics would enhance interpretability and usefulness.

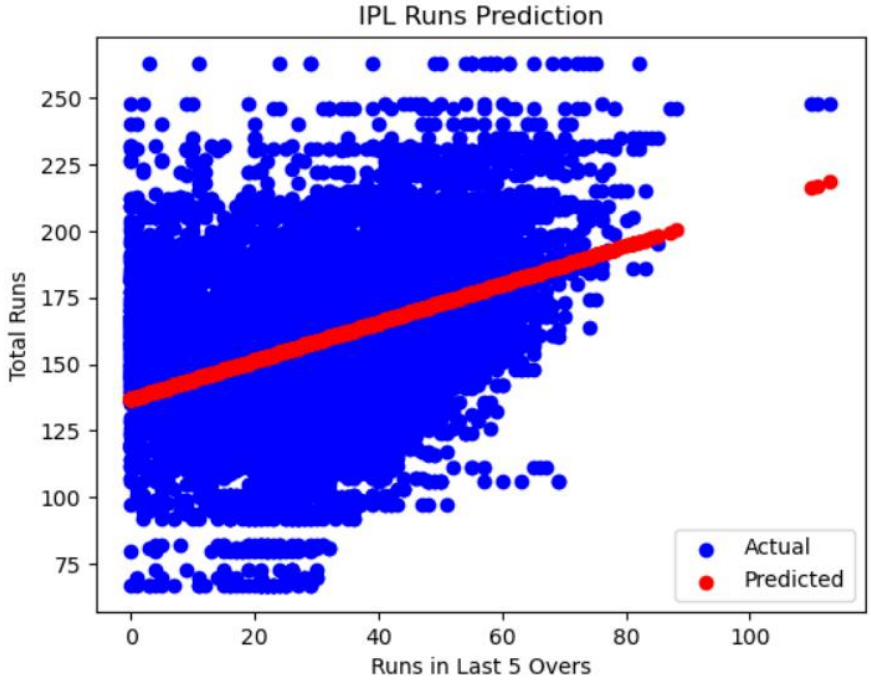

Fig: Linear Regression

The classification report provides performance metrics for a binary classification model, likely used for IPL match outcome prediction (e.g., win/loss). The model demonstrates exceptionally high performance for class 0 (precision: 0.99, recall: 1.00, F1-score: 0.99), which represents the majority class with 22,531 instances. This suggests the model is nearly perfect at identifying this class. However, for class 1 (the minority class with only 274 instances), the performance is poor, with low precision (0.40), very low recall (0.02), and a minimal F1-score (0.04). This indicates the model struggles to correctly identify class 1 cases, likely due to severe class imbalance.

The overall accuracy of 0.99 is misleadingly high, as it is heavily skewed by the model's ability to predict the majority class. The macro average (0.69 precision, 0.51 recall, 0.52 F1-score) reveals a more balanced but still mediocre performance when both classes are considered equally, while the weighted average (0.98 precision, 0.99 recall, 0.98 F1-score) again reflects the dominance of the majority class.

In practical terms, this model would be unreliable for predicting the minority class (e.g., rare events like upsets or close matches in IPL). To improve performance, techniques such as resampling (oversampling the minority class or undersampling the majority class), using class-weighted loss functions, or exploring alternative algorithms better suited for imbalanced data should be considered. The current model, while accurate overall, fails to address the critical minority class, limiting its real-world usefulness.

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	22531
1	0.40	0.02	0.04	274
accuracy			0.99	22805
macro avg	0.69	0.51	0.52	22805
weighted avg	0.98	0.99	0.98	22805

Fig: classification report

The confusion matrix provides insights into the performance of an IPL win prediction model by comparing actual match outcomes (win/loss) against the model's predictions. The matrix shows that the model correctly predicted 22,522 losses (true negatives) and 6 wins (true positives). However, it misclassified 268 actual wins as losses (false negatives) and 9 actual losses as wins (false positives). The high number of true negatives indicates the model is highly accurate in predicting losses, likely due to a significant class imbalance where losses dominate the dataset.

However, the large number of false negatives (268) reveals the model's poor performance in correctly identifying wins, which aligns with the earlier classification report showing low recall for the minority class (wins). This suggests the model is biased toward predicting losses, missing nearly all win cases. The very low false positives (9) mean the model rarely misclassifies losses as wins, but this comes at the cost of failing to detect most actual wins. In practical terms, while the model appears accurate overall (due to class imbalance), it is unreliable for predicting wins—a critical flaw for applications like betting or strategy planning.

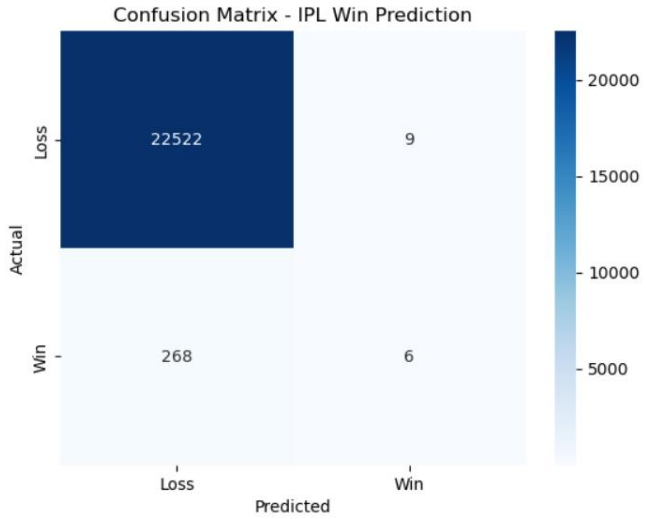

Fig: confusion matrix

The Training Loss Curve for an IPL Score Prediction Model, the Mean Squared Error (MSE) Loss against training epochs (0 to 50). The curve shows a rapid decline in loss during the initial epochs (0–10), indicating that the model is quickly learning meaningful patterns from the training data. As training progresses, the loss continues to decrease but at a slower rate, eventually stabilizing around 0.2 MSE by epoch 50. This suggests that the model has converged, meaning further training is unlikely to yield significant improvements. We can infer that the model is effectively learning the relationship between input features (e.g., player statistics, match conditions) and target scores.

Since this is only the training loss, we cannot fully assess generalization without a validation loss curve. If the validation loss follows a similar trend, the model is likely robust. If not, it may be overfitting (memorizing training data rather than learning generalizable patterns). A final MSE of 0.2 is promising, but its practical significance depends on the data scaling. If scores were normalized (e.g., divided by 100), the actual prediction error could be higher. To ensure reliability, the model should be tested on unseen match data and compared with baseline metrics like Mean Absolute Error (MAE) or R^2 score.

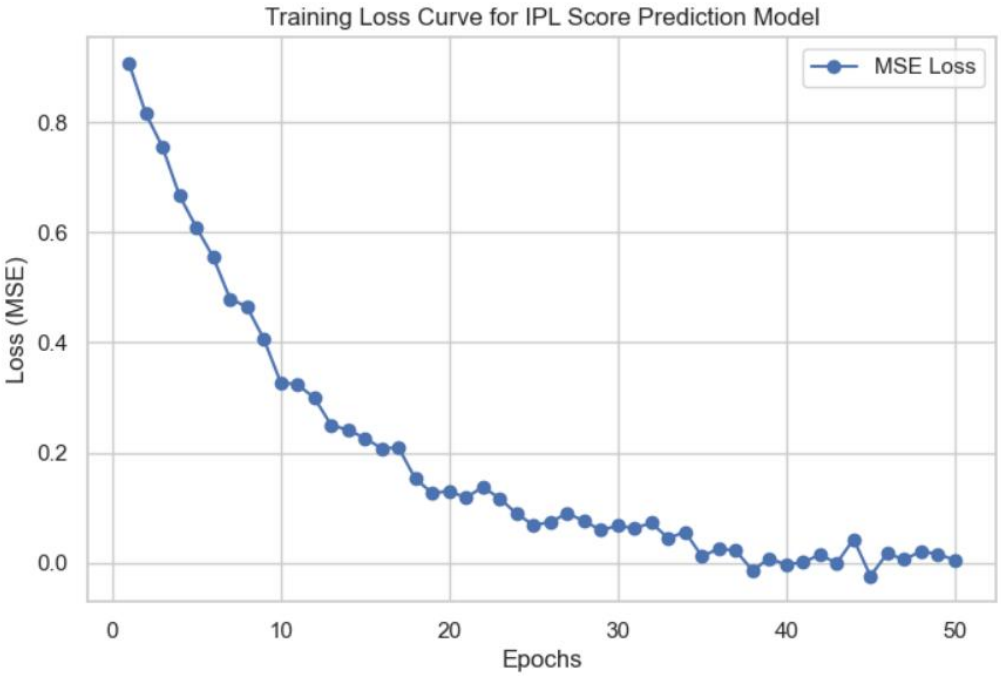

Fig: Training Loss Curve

Decision tree structure for predicting match outcomes (Win/Lose) in IPL cricket:-

With nodes representing splitting rules based on features like overs completed, wickets fallen, and runs scored. The tree uses Gini impurity (ranging from 0 to 0.5) to measure node purity, where lower values indicate clearer class separation. For instance, the root node (overs ≤ 19.55) shows a strong bias toward "Lose" predictions (52,613 samples vs. 596 "Win"), reflecting the imbalanced nature of cricket data where losses often dominate.

The Gini value of 0.029 here suggests minimal impurity, meaning most matches follow this pattern. Deeper nodes reveal nuanced decision boundaries. A critical split occurs at "runs ≤ 103.5," where 382 of 383 samples are classified as "Win," indicating that very low target runs (likely in shortened matches) heavily favor the chasing team. Conversely, nodes with higher Gini values (e.g., 0.353 for 397 samples) denote uncertainty, possibly representing close matches where wickets or overs remaining complicate predictions. The terminal nodes (Gini = 0) are perfectly pure, such as 25 samples all predicting "Win," likely corresponding to extreme scenarios (e.g., very low targets with ample wickets).

Theoretically, decision trees recursively partition data to maximize homogeneity, making them interpretable for cricket analytics. However, the model's heavy reliance on imbalanced data (evident from the 52,612:189 "Lose" ratio in one node) risks overfitting to majority classes. This aligns with the "class imbalance problem" in machine learning, where models may ignore rare but critical events (e.g., close wins).

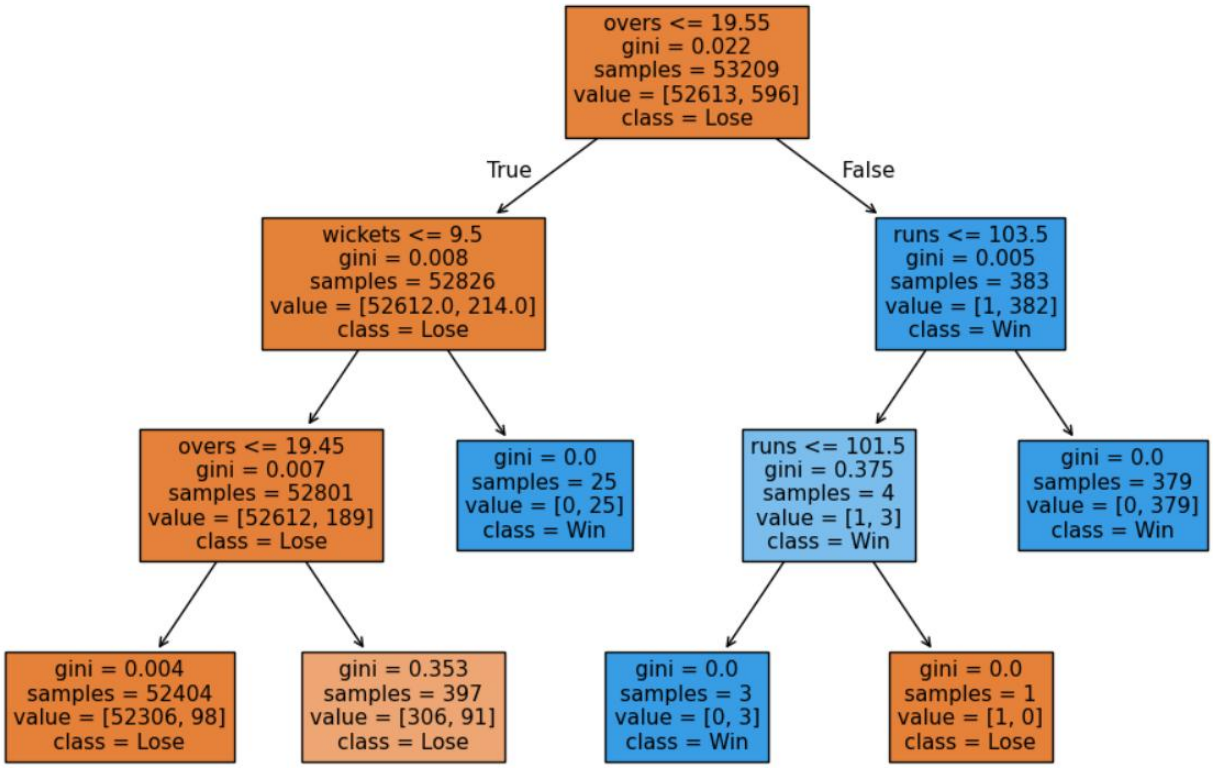

Fig: Decision Tree

Observations:-

Loss Trends - Both training loss and validation loss start high (~30) and decrease sharply. This indicates that the model is effectively learning patterns from the data. In Later Epochs (10–50),the losses plateau around 20–22 suggesting that further training yields minimal improvement. The convergence is stable, with no sudden spikes or drops. The validation loss closely follows the training loss, which is a positive sign.There is no significant divergence meaning the model is not overfitting (i.e., it generalizes well to unseen data). The model improves consistently without erratic behavior. No Overfitting-Training and validation losses remain close, indicating good generalization.

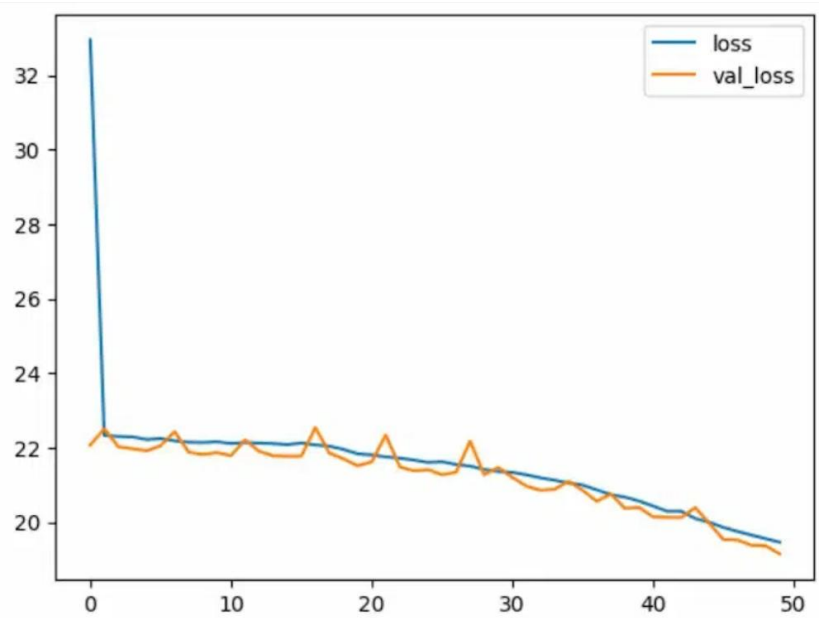

Fig: Epoch vs Loss & validation loss

CONCLUSION

The analysis of IPL data through machine learning techniques has provided valuable insights into predicting the total score of a match based on various parameters such as team composition, venue, and bowler performance. By leveraging data preprocessing techniques, including label encoding for categorical features and appropriate feature selection, the model was trained effectively. The application of Linear Regression as a predictive model demonstrated promising results, though some limitations were observed in the accuracy of predictions.

Key Findings:

(a)Data preprocessing significantly impacts the efficiency and accuracy of the model.
(b)The choice of features, such as batting and bowling teams, influences the prediction quality.
(c)Linear Regression performed reasonably well but could be enhanced by exploring advanced models like Random Forest, Gradient Boosting, or Deep Learning techniques.
(d)Evaluation metrics indicated that while the model captures general trends, further improvements are necessary to minimize prediction errors.

Future Scope and Recommendations:

(a)Advanced Model Implementation: Future iterations can explore complex machine learning models, such as Decision Trees, Random Forest, or Neural Networks, to enhance predictive performance.
(b)Feature Engineering: Additional variables, such as player statistics, pitch conditions, and weather data, could be incorporated to refine the model's accuracy.
(c)Hyperparameter Tuning: Optimizing hyperparameters using techniques like Grid Search or Bayesian Optimization may improve prediction accuracy.
(d)Ensemble Learning: Combining multiple models using ensemble techniques could lead to a more robust prediction system.
€Real-time Prediction: Deploying the model as a web-based application for real-time IPL score prediction can extend its practical applicability.

This project provides a strong foundation for predictive analytics in sports, showcasing how machine learning can be applied to enhance decision-making and strategic planning in cricket. Further refinements and additional data sources can contribute to making the model more comprehensive and accurate.

REFERENCE

1. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

3. Scikit-learn: Machine Learning in Python. Available at: https://scikit-learn.org/

4. IPL Official Statistics and Data. Available at: https://www.iplt20.com/stats

5. Kaggle IPL Dataset and Exploratory Analysis. Available at: https://www.kaggle.com/

6. McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter. O'Reilly Media.

7. “Ipl Cricket Score and Winning Prediction Using Machine Learning Techniques” Nikhil Dhonge Shraddha Dhole, Nikita Wavre, Mandar Pardakhe, Department of Electrical Engineering, MIT Academy of Engineering (Affiliated to SPPU), Pune, Maharashtra, India Issue: 5 May, 2021

8. "Prediction of IPL Match Score and Winner Using Machine Learning Algorithms", International Journal of Emerging Technologies and Innovative Research (www.jetir.org),ISSN:2349-5162, Vol.8, Issue 6, page no.c437-c444, June-2021

9. I. Anik, S. Yeaser, A. G. M. I. Hossain and A. Chakrabarty, "Player’s Performance Prediction in ODI Cricket Using Machine Learning Algorithms," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT).

10. “Prediction on Ipl Data Using Machine Learning Techniques in R Package” G. Sudhamathy and G. Raja Meenakshi Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, India CTACT JOURNAL ON SOFT COMPUTING.

11. "Prediction of IPL Match Score and Winner Using Machine Learning Algorithms", International Journal of Emerging Technologies and Innovative Research (www.jetir.org)

12. Lokhande, Chawan, Pramila, “Prediction of Live Cricket Score and Winning,” International Journal of Trend in Research and Development, Volume 5(1), (2018).