PROJECT REPORT FORMAT

A
PROJECT REPORT

On

Prediction of IPL Match Score and Winner Using Machine Learning Algorithms

Submitted In Partial Fulfillment of the Requirements [14 pts]for the Degree of

Bachelor of Technology

In

Artificial Intelligence & Machine Learning

By

Sachin Kumar Ray (00215611621)

Under the Supervision of

Dr. Preety Shoran

Department of Artificial Intelligence & Machine Learning

Dr. AKHILESH DAS GUPTA INSTITUTE OF TECHNOLOGY & MANAGEMENT

(A Unit of BBD Group)

Approved by AICTE and Affiliated with GGSIP University

FC-26, Shastri Park, New Delhi-110053

DECLARATION

I Sachin Kumar Ray (00215611621) of 4th semester B.Tech., in the department of Artificial Intelligence and Machine Learning from ADGITM, New Delhi, hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previouslypublished or written by another person nor material which to a substantial extent has been accepted for the award of any other degree of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: Sachin Kumar Ray

Roll No.: 00215611621

Date:

CERTIFICATE

This is to certify that Project Report entitled “Prediction of IPL Match Score And Winner Using Machine Learning’ which is submitted by Sachin Kumar Ray in partial fulfillment of the requirement for the award of degree B. Tech. in the Department of Artificial Intelligence and Machine Learning of Dr. Akhilesh Das Gupta Institute of Technology & Management (ADGITM) formerly known as Northern India Engineering College (NIEC), New Delhi, is a record of the candidate’s own work carried out by him under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of anyother degree**.**

Date: **Supervisor**

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B.Tech 2nd Year. I owe a special debt of gratitude to Dr. Preety Shoran for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness, and perseverance have been a constant source of inspiration for me. It is only her cognizant efforts that our endeavors have seenlight of the day.

I also take the opportunity to acknowledge the contribution of Dr. Preety Shoran for his/her full support andassistance during the development of the project.

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development ofour project. Last but not the least, I acknowledge our friends for their contribution in the completion of the project.

Signature:
Name: Sachin Kumar ray
Roll No. 00215611621
Date:

ABSTRACT

This study explains the application of machine learning to sports. According to recent changes in data science and sports, the use of sports-based machine learning and data mining shows the importance of process in outcome performance and prediction. The purpose of this paper is to evaluate current measurements used in the literature to understand the estimation methods used to model and analyse data and characterize the variables that govern performance. Finally, this article will present a reliable tool for data analysis using machine learning.

In today's world, sports produce enough statistical information about every player, team, match, and season. The first sports researchers were thought to be experts, coaches, team managers and analysts. Sports organizations have recently realized that there is research in their data and want to do research using different data mining methods. Sports data helps coaches and managers in many ways, such as predicting results, analysing player performance, skills, and evaluating strategies. Forecasts help managers and organizations make decisions to win teams and competitions. The present study shows that preliminary studies of data mining systems can predict outcomes and evaluate the strengths and weaknesses of each system. Predictions are made for each match. Although in many respects this application is very limited. It is very important to examine the use of machine learning in these situations to see if the application can provide better results in analysis.

This research aims to provide solutions that will help make predictions more accurate and precise than previous methods, using more accurate data and machine learning.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

[x]	Integer value of x.
≠	Not Equal
χ	Belongs to
€	Euro- A Currency
_	Optical distance
_o	Optical thickness or optical half thickness

LIST OF ABBREVIATIONS [16 Pts]

AAM	Active Appearance Model**[12 pts]**
ICA	Independent Component Analysis
ISC	Increment Sign Correlation
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics

CHAPTER -1

Introduction

Cricket, as one of the most popular sports worldwide, has captured the attention and enthusiasm of millions of fans across the globe. The Indian Premier League (IPL), in particular, has emerged as a premier professional Twenty20 cricket league, showcasing thrilling matches and attracting a massive fan base. With its high-scoring nature and unpredictable outcomes, the IPL offers an ideal platform for exploring the application of machine learning techniques to predict match scores.

The ability to predict the outcome of a cricket match accurately has long been a subject of interest and fascination among fans and analysts. Traditional methods of prediction have relied on statistical analysis, historical data, and expert opinions. However, the emergence of machine learning algorithms and advancements in data analytics have opened up new avenues for enhancing the accuracy and precision of predictions.

In this project, we delve into the exciting domain of IPL score prediction using machine learning techniques. The primary objective is to develop a predictive model capable of estimating the total score of an IPL match based on various factors such as team performance, player statistics, venue conditions, and match-specific attributes. By harnessing the power of machine learning, we aim to create a reliable tool that can assist cricket enthusiasts, team managers, and betting analysts in making informed decisions.

The IPL score prediction project presents an opportunity to explore and understand the intricacies of machine learning algorithms and their application to real-world scenarios. By analyzing and processing vast amounts of historical IPL data, we can identify patterns, correlations, and dependencies that contribute to the final score. Through feature engineering and model training, we aim to develop an accurate and robust predictor that can adapt to the dynamic nature of the IPL.

The outcome of this project has significant implications for various stakeholders involved in the IPL ecosystem. Team strategists can leverage the predictive model to formulate effective game plans, optimize batting lineups, and make data-driven decisions during matches. Broadcasters and media houses can utilize the predictions to enhance viewer engagement, generate insightful analysis, and provide an immersive experience to the fans. Additionally, fans and betting enthusiasts can benefit from the model's predictions to enhance their understanding, engage in informed discussions, and make more accurate predictions themselves.

In summary, this project aims to combine the excitement of the IPL with the power of machine learning to predict match scores accurately. By exploring the vast IPL dataset and developing an advanced predictive model, we aspire to contribute to the growing field of sports analytics and provide valuable insights into the dynamics of T20 cricket. The subsequent sections will delve into the methodology, data collection and preprocessing, feature engineering, model development, evaluation, and the results obtained, followed by a comprehensive analysis and conclusion.

1.1 Introduction of the Problem

The Indian Premier League (IPL), a highly popular and fiercely competitive Twenty20 cricket league, has captivated fans worldwide with its fast-paced matches and thrilling moments. As the IPL continues to grow in popularity, the desire to accurately predict match scores has become a topic of great interest and intrigue. Predicting the final score of an IPL match is a complex task due to the dynamic nature of the game, the diverse skill sets of the players, and the impact of various external factors such as pitch conditions, weather, and team dynamics.

The problem of IPL score prediction poses several challenges. Traditional methods of prediction, relying solely on statistical analysis and expert opinions, often fall short in capturing the nuanced factors that influence match outcomes. These methods struggle to account for the intricate interplay between player performance, team strategy, and situational factors that ultimately determine the final score. Therefore, there is a growing need to leverage advanced machine learning techniques to unravel the patterns and trends hidden within vast amounts of IPL data and make accurate predictions.

By employing machine learning algorithms, we aim to address this problem by developing a predictive model capable of estimating the total score of an IPL match. This model will consider various factors such as historical team performance, player statistics, venue characteristics, and match-specific attributes to generate reliable predictions. By training the model on a comprehensive dataset, we can uncover the underlying relationships between these factors and the final score, enabling us to make informed predictions for future matches.

The successful development of an IPL score prediction model using machine learning has far-reaching implications. Cricket teams can benefit from such a model by gaining insights into optimal batting strategies, player selections, and game plans. Media houses and broadcasters can enhance their coverage by providing real-time predictions and analysis to engage viewers and add depth to their commentary. Furthermore, fans and betting enthusiasts can use the predictions as a valuable tool for informed discussions, fantasy leagues, and making more accurate predictions themselves.

In this project, we aim to contribute to the field of sports analytics by tackling the challenging problem of IPL score prediction using machine learning techniques. By harnessing the power of data and employing sophisticated algorithms, we aspire to unlock valuable insights that can enhance the understanding and enjoyment of IPL matches. The subsequent sections will outline the methodology, data collection and preprocessing, feature engineering, model development, evaluation, and the results obtained, followed by a comprehensive analysis and conclusion.

1.2 Summarize Previous Research

Previous research on IPL score prediction using machine learning has made significant strides in developing accurate models for predicting the total score of IPL matches. Researchers have explored various methodologies, including feature engineering, regression techniques, ensemble methods, neural networks, and dimensionality reduction.

Feature engineering has played a crucial role in identifying relevant predictors such as historical team performance, player statistics, venue characteristics, and contextual factors. Regression techniques, such as linear regression and polynomial regression, have been utilized to establish relationships between input features and the target variable. Ensemble methods, such as random forests and gradient boosting, have demonstrated their effectiveness in improving prediction accuracy by combining multiple models.

Deep learning approaches, particularly neural networks, have shown promise in capturing complex patterns and dependencies within IPL data. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) like LSTM networks have been employed to model temporal and spatial information. Feature selection and dimensionality reduction techniques have been used to improve model performance and reduce complexity.

Evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and accuracy have been commonly used to assess model performance. Cross-validation techniques, such as k-fold cross-validation, have been employed to ensure model robustness and generalizability.

Overall, previous research has demonstrated the potential of machine learning in IPL score prediction. The combination of feature engineering, regression techniques, ensemble methods, neural networks, and dimensionality reduction has led to improved accuracy and predictive capabilities. These findings lay the foundation for further advancements in developing robust and accurate models for IPL score prediction using machine learning.

1.3 Researching the Problem

Research of the problem of IPL score prediction using machine learning involves exploring existing literature, studies, and methodologies related to the topic. Here is an outline of the key areas to focus on during the research:

1. Literature Review: Conduct a comprehensive review of existing research papers, articles, and publications related to IPL score prediction, machine learning, and sports analytics. Identify the state-of-the-art techniques, methodologies, and findings in this field. Analyze the strengths and limitations of previous studies and identify research gaps.

2. Data Sources: Identify reliable and relevant sources of IPL data, such as official IPL websites, cricket statistics databases, or APIs. Explore the availability and accessibility of match data, player statistics, team records, and other relevant information required for IPL score prediction.

3. Feature Selection: Investigate the features used in previous studies and research to determine the most relevant predictors for IPL score prediction. Explore the significance of factors such as team performance, player statistics, venue conditions, match-specific attributes, and contextual factors like toss outcomes or weather conditions. Consider incorporating innovative features that capture unique aspects of IPL matches.

4. Machine Learning Algorithms: Explore various machine learning algorithms suitable for regression tasks and analyze their applicability to IPL score prediction. Investigate algorithms such as linear regression, decision trees, random forests, support vector machines, neural networks, and ensemble methods. Analyze their strengths, limitations, and suitability for the IPL score prediction problem.

5. Model Development: Design and develop predictive models based on the selected machine learning algorithms. Implement the models using appropriate programming languages or frameworks such as Python with libraries like scikit-learn or TensorFlow. Define the pipeline for data preprocessing, feature engineering, model training, and prediction.

6. Evaluation Metrics: Determine appropriate evaluation metrics for assessing the performance of the predictive models. Consider metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), accuracy, or precision-recall. Select the most suitable metrics based on the nature of the problem and the available data.

7. Experimental Setup: Define the experimental setup, including the training and testing datasets, cross-validation techniques, and hyperparameter tuning strategies. Implement robust experimental procedures to ensure unbiased evaluation and model generalizability.

8. Comparative Analysis: Compare the performance of different models and algorithms based on the evaluation metrics. Analyze the strengths and weaknesses of each approach and identify the most effective models for IPL score prediction. Consider factors such as accuracy, computational efficiency, interpretability, and scalability.

9. Interpretability and Explainability: Investigate methods for interpreting and explaining the predictions made by the models. Explore techniques such as feature importance analysis, model visualization, or post-hoc explanation methods to gain insights into the factors driving the predictions.

10. Limitations and Future Directions: Discuss the limitations and potential sources of bias in the research. Address any ethical considerations related to the use of machine learning for score prediction. Propose future directions for research, including potential enhancements to the models, incorporation of additional data sources, or exploration of advanced techniques.

CHAPTER -2

Literature Survey

The Indian Premier League (IPL) is a popular professional Twenty20 cricket league in India. Predicting the scores of IPL matches is a challenging task due to the complex nature of the game. Machine learning techniques have been widely applied to predict the outcomes and scores of IPL matches. This literature survey aims to provide an overview of the existing research and methodologies employed for IPL score prediction using machine learning.

- Sharma A, & Saini, R. [1] presented an approach for IPL score prediction using multiple machine learning algorithms such as decision trees, random forests, and support vector machines (SVM). The authors compared the performance of these algorithms and evaluate their accuracy in predicting the scores of IPL matches.

- Kulkarni, P., & Gokhale, A. [2] presented an ensemble learning approach that combines multiple machine learning algorithms for IPL score prediction. The authors experiment with various classifiers, including k-nearest neighbours (KNN), SVM, and logistic regression. They evaluate the performance of the ensemble model and compare it with individual algorithms.

- Singh, H., & Grover, A. [3] presented a research focuses on the combination of long short-term memory (LSTM) and random forest regression for IPL score prediction. The authors propose a hybrid model that leverages the sequence modeling capabilities of LSTM and the ensemble learning approach of random forests. The performance of the proposed model is evaluated and compared with other techniques.

- Jaiswal, A., & Singh, S. (2021). [4] explores the application of the XGBoost algorithm, a gradient boosting technique, for IPL score prediction. The authors describe the feature selection process and evaluate the performance of the XGBoost model. They also compare its results with other machine learning algorithms.

- Gupta, M., & Khandelwal, M. (2021). [5] investigates the use of deep learning techniques, specifically convolutional neural networks (CNNs), for IPL score prediction. The authors propose a CNN-based model that considers various features related to teams, players, and match conditions. They evaluate the performance of the model and discuss its potential for accurate score prediction.

IPL score prediction using machine learning techniques has gained significant attention from researchers. The literature survey showcases various approaches, including ensemble learning, LSTM, XG Boost, and

deep learning techniques, for predicting IPL scores. The performance of these models is evaluated and compared using different evaluation metrics. Further research in this domain can explore the incorporation of additional features, real-time data, and more advanced machine learning algorithms to enhance the accuracy of IPL score prediction.

CHAPTER -3

Methodology and Technology

3.1 Data Analysis and Preprocessing

a. Data Collection and exploration

Data gathering is the fundamental module and the first stage of the project. The main aim is to collect appropriate dataset that is suitable for our needs and Data exploration is the first step of data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns to dig into more. The goal of data exploration is to learn about characteristics and potential problems of a data set without the need to formulate assumptions about the data beforehand. We take the data set match.csv, deliveries.csv and ipl.csv from Kaggle.

b. Data pre-processing

Machine learning relies heavily on data pre-processing to get highly accurate and insightful outputs so that we can apply that data on our model. A data set contain a lot of data which is not of our use so we do data pre-processing and convert data into a systematic form as of our need. The more reliable the produced results are, the better the data quality is. Realworld datasets are characterized by incomplete, noisy, and inconsistent data. Data pre-processing improves data quality by filling in missing or partial data, reducing noise, and addressing discrepancies.

c. Data cleaning

The process of eliminating errors and replacing them with genuine values is known as data cleaning. The data sets gathered contain noisy data, such as null values and inappropriate values, which must be cleaned. As a result, the data is cleaned by replacing null values with zeros, and the data is organized into correct columns so that we can properly analyse it.

d. Data visualization

The data that has been gathered is used to visualize the information for better comprehension. The Matplotlib and seaborn Library were used to visualize the graphs for team wins in various cities based on their venues and player strike rate.

e. Splitting into train and test

Train test split is a technique of splitting dataset into two parts that is used to estimate the performance of machine learning model by feeding the train data to the model and predict the data on test data and new data. A train test is the way of structuring your machine learning project so that you can test your hypothesis quickly and inexpensively. Basically, it's a way to divide the training data so that you can try your algorithm to one half and evaluate the result on the other half.

A train test split is when you split your data into a training set and a testing set. The training set is used for training the model, and the testing set is used to test your model. This allows us to train our models on the training data set, and then test their accuracy on the unseen testing set. There are a few different ways to do a train test split, but the most common is to simply split your data into two sets. For example, 80% for training and 20% for testing.

3.2 Machine Learning Model Used

f. Machine Learning Algorithms

Various machine learning algorithms can be applied to train predictive models. Some commonly used algorithms for prediction tasks include linear regression, logistic regression, decision trees, random forests, support vector machines (SVM), and gradient boosting algorithms like XGBoost or LightGBM. The choice of algorithm depends on the specific requirements and performance metrics

Algorithms used

1. Linear Regression

Linear regression is a statistical modelling technique used to establish a linear relationship between a dependent variable and one or more independent variables. It aims to fit a straight line to the data points in such a way that it minimizes the difference between the predicted values and the actual values.

The general form of a linear regression model is represented by the equation:

$y = b0 + b1x1 + b2x2 + ... + bn*xn$

where:

y is the dependent variable or the target variable that we want to predict.

b0 is the intercept or the constant term.

b1, b2, ..., bn are the coefficients or slopes associated with the independent variables x1, x2, ..., xn.

x1, x2, ..., xn are the independent variables.

It aims to establish a linear relationship between independent variables (predictors) and a dependent variable (target) to make predictions. By analysing historical data and training a linear regression model, one can make predictions about the scores of future IPL matches, providing valuable insights for teams, analysts, and cricket enthusiasts.

2. Logistic Regression

Logistic regression is one of the most useful machine learning algorithms based a statistical modelling technique that predict the dependent data variable by analysing the relationship between one or more independent variable. It is commonly used for binary classification problems such as yes or no, 0 or 1, True or False, while it is not typically used directly for

predicting scores in IPL matches, it can be employed for predicting the likelihood of a team winning or losing a match based on various factors. In this context, logistic regression can be utilized to estimate the probability of a team winning an IPL match. Logistic regression can provide insights into the likelihood of winning an IPL match based on historical data and relevant features.

3. **Random Forest Classifier**

   Random Forest Classifier is a machine learning algorithm that can be utilized for predicting the winning outcome of IPL matches. It is an ensemble learning method that combines multiple decision trees to make predictions. Random Forest Classifier is advantageous in IPL winning prediction because it can handle non-linear relationships and interactions among features, handle high-dimensional datasets, and mitigate overfitting issues. It also provides insights into feature importance, allowing for a better understanding of the factors influencing match outcomes. It's important to note that the performance of the Random Forest Classifier can be further optimized by tuning hyperparameters such as the number of decision trees, the depth of the trees, and the number of features considered at each split. It gives the one-sided prediction at every point that is predicted on data but for real situation this type of prediction is not of our use.

4. **Lasso Regression**

Lasso Regression, also known as L1 regularization, is a linear regression technique that incorporates a penalty term to encourage sparse solutions by shrinking the coefficients of less important features to zero. While Lasso Regression is not commonly used for direct IPL score prediction, it can be applied to select important features that influence the score and improve the model's performance. Lasso Regression helps in feature selection by highlighting the most influential features in predicting the IPL scores. By reducing the coefficients of less important features to zero, it provides a sparse solution and simplifies the model.

5. **Ridge Regression**

Ridge Regression is a linear regression technique that includes a regularization term called L2 regularization. It is commonly used for predicting IPL scores by minimizing the sum of squared errors while also shrinking the regression coefficients. Ridge Regression is a useful technique for IPL score prediction by incorporating regularization to prevent overfitting and shrink the coefficients.

g. Model Training and Evaluation:

The collected and preprocessed data is divided into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate its performance. Evaluation metrics such as mean squared error (MSE), accuracy, precision, recall, or F1 score can be used to

assess the model's performance.

h. Model Deployment

Once the predictive model is trained and evaluated, it can be deployed as a service or integrated into a web or mobile application. This allows users to input relevant information about an upcoming IPL match, and the model can provide a predicted score based on the input data and trained parameters.

CHAPTER -4

Result Analysis and Discussion

4.1 Result Analysis

Predicting sports scores, such as IPL cricket scores, using machine learning models can be an interesting and challenging task. Here is a general outline of the steps involved in building a machine learning model for IPL sports score prediction:

1. Data Collection: Gather historical match data, including features such as team performance indicators, player statistics, venue, weather conditions, and other relevant factors. Ensure the dataset encompasses a significant number of matches for robust analysis.
2. Data Pre-processing: Clean and pre-process the collected data, handling missing values, outliers, and categorical variables as necessary. Normalize or scale the features to ensure they are on a similar scale for effective modelling.
3. Data Partitioning: Split the dataset into training, validation, and testing sets. The training set will be used to train the model, the validation set will be used for hyperparameter tuning and model selection, and the testing set will be used to evaluate the final performance.

4. Model Selection: Choose an appropriate machine learning algorithm for sports score prediction. Depending on the nature of the problem and the characteristics of the data, algorithms such as regression models such as linear regression, random forest regression, logistic regression, lasso regression and ridge regression can be considered.

5. Model Evaluation: Evaluate the trained model using the validation set. Measure its performance using evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or other relevant metrics specific to sports prediction. Assess how well the model predicts the sports scores compared to the actual scores.

Model	MSE	RMSE	MAE	Accuracy
Linear Regression	257.50	16.04	12.31	72.16
Ridge Regression	257.11	16.03	12.29	72.20
Lasso Regression	263.73	16.24	12.34	71.49

TABLE: 1 Model Evaluation for IPL FIRST INNINGS SCORE PREDICTION

Model	Accuracy
Logistic Regression	83.10

TABLE: 2 Model Evaluation for 2nd INNINGS WIN PREDICTION

6. Final Model Evaluation: Once the model is optimized, evaluate its performance using the testing set, which represents unseen data. Calculate the performance metrics to assess the model's predictive ability and determine its effectiveness in predicting sports scores.

7. Deployment and Prediction: Deploy the final model to make predictions on new, unseen sports match data. Provide the relevant features for each match as input to the model, and it will predict the scores based on the learned patterns and relationships from the training process.

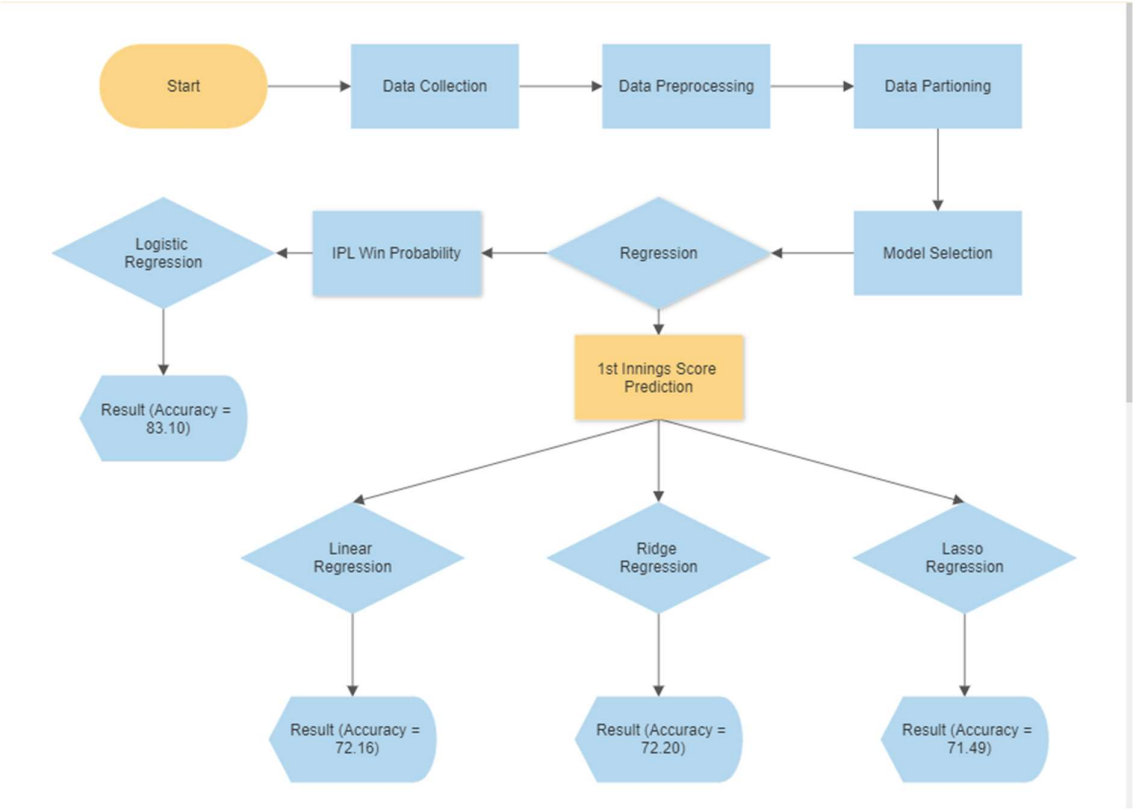


Fig: 1. Flow Chart of the Model

4.2 Discussion

IPL score prediction using machine learning is an intriguing research area with the potential to enhance our understanding of the game and aid in decision-making processes. By employing machine learning techniques, researchers aim to develop models that can accurately predict the total score of IPL matches. Here, we can discuss the broader implications, challenges, and potential advancements in this field.

1. Accuracy and Performance:
    - One of the primary goals of IPL score prediction is to achieve high accuracy in the predictions. Discuss the performance metrics used to evaluate the accuracy of predictive models, such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or accuracy.

- Analyze the performance of existing models in the literature and highlight their achievements in terms of accuracy. Compare their results with traditional statistical approaches or human expert predictions to showcase the advancements made by machine learning methods.

2. Feature Selection and Engineering:
    - Feature selection plays a crucial role in IPL score prediction. Discuss the relevant features used in predictive models, such as historical team performance, player statistics, venue characteristics, match-specific attributes, or contextual factors like toss outcomes or weather conditions.
    - Highlight the importance of feature engineering techniques in capturing valuable insights from the available data. Explore how innovative features or combinations of features can contribute to better predictions.
3. Choice of Machine Learning Algorithms:
    - Discuss the various machine learning algorithms employed in IPL score prediction. Consider approaches such as linear regression, decision trees, random forests, support vector machines, neural networks, or ensemble methods.
    - Analyze the strengths and weaknesses of different algorithms and their suitability for IPL score prediction. Highlight the algorithms that have shown promising results in terms of accuracy, interpretability, and scalability.
4. Interpretability and Explainability:
    - Interpretability and explainability are essential aspects of IPL score prediction models. Discuss techniques used to interpret and explain the predictions made by the models, such as feature importance analysis, model visualization, or post-hoc explanation methods.
    - Explore the implications of interpretability and explainability in gaining insights into the factors driving the predictions. Discuss how these insights can be used to inform decision-making processes in the context of IPL matches.
5. Challenges and Future Directions:
    - Address the challenges and limitations associated with IPL score prediction using machine learning. Discuss issues related to data quality, sample size, overfitting, concept drift, or the dynamic nature of the game.
    - Propose future directions for research and potential advancements in the field. Consider areas such as incorporating additional data sources (e.g., social media sentiment analysis), exploring advanced techniques (e.g., deep learning, reinforcement learning), or considering match-specific context (e.g., player injuries, team strategies).
6. Practical Applications:
    - Discuss the practical implications of IPL score prediction using machine learning. Explore how accurate predictions can be utilized in practice, such as enhancing betting odds, aiding team strategies, supporting media coverage, or engaging fans.

- Highlight the potential benefits and challenges of implementing predictive models in real-world scenarios and discuss the ethical considerations surrounding their use.

Overall, IPL score prediction using machine learning offers exciting possibilities for improving the understanding and analysis of cricket matches. By addressing the challenges and pushing the boundaries of research in this field, we can unlock new insights and develop robust models for accurate score predictions.

CHAPTER -5
Conclusions and Future Work

5.1 Conclusion

In conclusion, machine learning models have shown promise in predicting IPL scores and determining the winning team. By leveraging historical match data and relevant features such as player performance, team composition, pitch conditions, and venue, these models can capture patterns and relationships to make informed predictions.

However, it is important to acknowledge that predicting IPL scores and determining the winning team is a challenging task due to the inherent uncertainties and complexities of cricket matches. Factors such as player form, injuries, team strategies, weather conditions, and other unforeseen events can have a significant impact on the outcome.

Machine learning models provide a systematic approach to analyze and extract insights from the available data, but they have limitations. The accuracy of the predictions heavily relies on the quality and relevance of the data, the selection of appropriate features, and the choice of an effective model. Additionally, models need to be regularly updated and refined with new data to adapt to changing trends and dynamics in the sport.

While machine learning can improve prediction accuracy to some extent, it's important to consider other factors such as expert knowledge, match analysis, and contextual information when making IPL score and win predictions. Combining machine learning models with human expertise can lead to more robust and accurate predictions.

Overall, machine learning models serve as valuable tools in the domain of IPL score and win prediction, providing insights and aiding decision-making. However, they should be used in conjunction with other analytical approaches and expert judgment to account for the complex and unpredictable nature of cricket matches.

5.2 Future Work

The field of IPL score prediction using machine learning holds significant potential for future research and advancements. Here are some areas for future work in this domain:

1. Integration of Advanced Techniques: Explore the integration of advanced machine learning techniques such as deep learning, reinforcement learning, or ensemble methods. These approaches can capture complex patterns, temporal dependencies, and nonlinear relationships in the data, potentially leading to improved accuracy in score prediction.
2. Incorporation of Unstructured Data: Consider incorporating unstructured data sources such as live commentary, social media sentiment, or video analysis. These sources can provide valuable real-time information and insights that contribute to more accurate and timely predictions.

3. Contextual Factors: Investigate the inclusion of additional contextual factors that may impact match outcomes, such as player injuries, team strategies, pitch conditions, or player form. By considering these dynamic variables, the predictive models can capture a more comprehensive picture of the match scenario and make more precise predictions.
4. Transfer Learning: Explore the potential of transfer learning by leveraging pre-trained models from related domains or previous IPL seasons. By transferring knowledge from similar tasks or previous seasons, the models can adapt more quickly to new data and potentially improve their predictive performance.
5. Online Learning: Develop models that can continuously learn and adapt to new data as the IPL season progresses. Online learning techniques enable the models to update their predictions in real-time, incorporating the latest match results and player performances for enhanced accuracy.
6. Ensemble Methods: Investigate the use of ensemble methods, such as combining predictions from multiple models or incorporating the wisdom of crowds, to improve the robustness and accuracy of the predictions. Ensemble techniques can help mitigate the limitations of individual models and provide more reliable forecasts.
7. Model Explainability: Enhance the interpretability and explainability of the predictive models to provide insights into the factors driving the predictions. Develop techniques to explain the contributions of different features and model components, allowing stakeholders to understand the reasoning behind the predictions.
8. External Validation and Collaboration: Validate the predictive models on independent IPL seasons and collaborate with IPL teams, broadcasters, or betting platforms for real-world testing and feedback. Collaborative efforts can provide valuable insights, practical applications, and opportunities for model refinement and improvement.
9. Evaluation Metrics: Explore alternative evaluation metrics beyond traditional accuracy measures, such as calibration, uncertainty estimation, or profit-based metrics. These metrics can provide a more comprehensive assessment of the predictive models' performance in practical applications.
10. Ethical Considerations: Consider the ethical implications of IPL score prediction, particularly in the context of betting and gambling. Develop guidelines and frameworks to ensure responsible use of predictive models and address potential risks and concerns associated with their application.

By focusing on these future directions, researchers can further enhance the accuracy, reliability, and practical utility of IPL score prediction models. Advancements in these areas will not only benefit stakeholders within the IPL ecosystem but also contribute to the broader field of sports analytics and machine learning applications.

APPENDIX A

SURVEY DATA OR TYPICAL PART OF SOURCE CODE

A.1. GUI

Fig: 2. Homepage of GUI

Fig: 3. IPL FIRST INNINGS SCORE PREDICTION

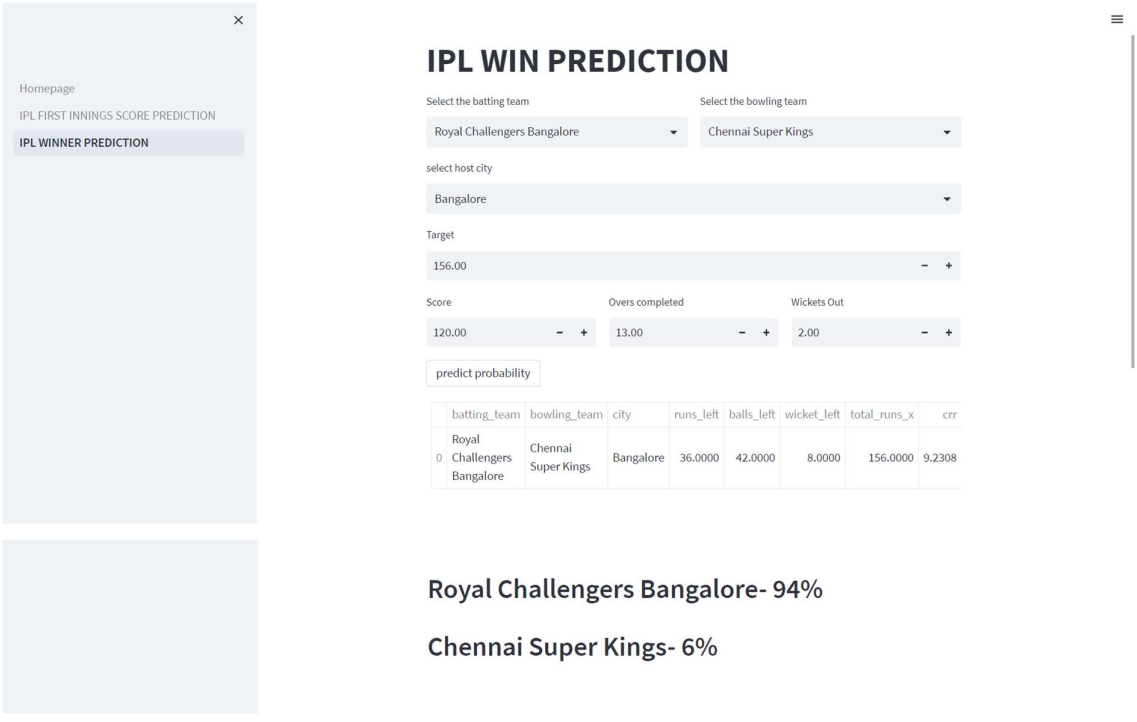


Fig:4: IPL WIN PREDICTION

A.2. SOURCE CODE

A.2.1. MODEL: 1 IPL FIRST INNINGS SCORE PREDICTION

Ipl First Inning run prediction

In [1]:
```
# Importing essential libraries
import pandas as pd
import pickle
import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns
```

In [2]:
```
# Loading the dataset
df = pd.read_csv('ipl.csv')
```

In [3]:
```
df
```

In [4]:
```
# --- Data Cleaning ---
# Removing unwanted columns
columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']
df.drop(labels=columns_to_remove, axis=1, inplace=True)
```

In [5]:
```
df.head()
```

Out[5]:

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222

In [6]:
```
df['bat_team'] = df['bat_team'].str.replace('Delhi Daredevils', 'Delhi Capitals')
df['bowl_team'] = df['bowl_team'].str.replace('Delhi Daredevils', 'Delhi Capitals')

df['bat_team'] = df['bat_team'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')
df['bowl_team'] = df['bowl_team'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')

df['bat_team'] = df['bat_team'].str.replace('Gujarat Lions', 'Gujarat Titans')
df['bowl_team'] = df['bowl_team'].str.replace('Gujarat Lions', 'Gujarat Titans')

df['bat_team'] = df['bat_team'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')
df['bowl_team'] = df['bowl_team'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')

```

In [7]:
```
df['bat_team'].unique()
```

In [8]:
```
# Keeping only consistent teams
consistent_teams = [
    'Sunrisers Hyderabad',
    'Mumbai Indians',
    'Gujarat Titans',
    'Royal Challengers Bangalore',
    'Kolkata Knight Riders',
    'Kings XI Punjab',
    'Chennai Super Kings',
    'Rajasthan Royals',
    'Delhi Capitals',
    'Lucknow Super Giants'
]
```

In [9]:
```
df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_teams))]
```

In [11]:
```
# Removing the first 5 overs data in every match
df = df[df['overs']>=5.0]
```

In [12]:
```
df.head()
```

Out[12]:

	date	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
32	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	0	5.1	59	0	222
33	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.2	59	1	222
34	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.3	59	1	222
35	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.4	59	1	222
36	2008-04-18	Kolkata Knight Riders	Royal Challengers Bangalore	61	1	5.5	58	1	222

In [13]:
```
print(df['bat_team'].unique())
print(df['bowl_team'].unique())
```

```
['Kolkata Knight Riders' 'Chennai Super Kings' 'Rajasthan Royals'
 'Mumbai Indians' 'Kings XI Punjab' 'Royal Challengers Bangalore'
 'Delhi Capitals' 'Sunrisers Hyderabad' 'Lucknow Super Giants'
 'Gujarat Titans']
['Royal Challengers Bangalore' 'Kings XI Punjab' 'Delhi Capitals'
 'Rajasthan Royals' 'Mumbai Indians' 'Chennai Super Kings'
 'Kolkata Knight Riders' 'Sunrisers Hyderabad' 'Lucknow Super Giants'
 'Gujarat Titans']
```

In [14]:
```
# Converting the column 'date' from string into datetime object
from datetime import datetime
df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_72088\486738173.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
```

In [15]:
```
# --- Data Preprocessing ---
# Converting categorical features using OneHotEncoding method
encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])
```

In [16]:
```
encoded_df.head()
```

In [18]:
```
# Rearranging the columns
encoded_df = encoded_df[['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Capitals', 'bat_team_Gujarat Titans', 'bat_
             'bat_team_Kolkata Knight Riders', 'bat_team_Lucknow Super Giants', 'bat_team_Mumbai Indians', 'bat_team_Rajast
             'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
             'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Capitals', 'bowl_team_Gujarat Titans', 'bowl_team_Kings XI P
             'bowl_team_Kolkata Knight Riders', 'bowl_team_Lucknow Super Giants', 'bowl_team_Mumbai Indians', 'bowl_team_Ra
             'bowl_team_Royal Challengers Bangalore', 'bowl_team_Sunrisers Hyderabad',
             'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total']]
```

In [19]:
```
# Rename columns
encoded_df.columns = ['date', 'bat_team_Chennai_Super_Kings', 'bat_team_Delhi_Capitals', 'bat_team_Gujarat_Titans',
       'bat_team_Kings_XI_Punjab', 'bat_team_Kolkata_Knight_Riders', 'bat_team_Lucknow_Super_Giants',
       'bat_team_Mumbai_Indians', 'bat_team_Rajasthan_Royals',
       'bat_team_Royal_Challengers_Bangalore', 'bat_team_Sunrisers_Hyderabad',
       'bowl_team_Chennai_Super_Kings', 'bowl_team_Delhi_Capitals', 'bowl_team_Gujarat_Titans',
       'bowl_team_Kings_XI_Punjab', 'bowl_team_Kolkata_Knight_Riders', 'bowl_team_Lucknow_Super_Giants',
       'bowl_team_Mumbai_Indians', 'bowl_team_Rajasthan_Royals',
       'bowl_team_Royal_Challengers_Bangalore',
       'bowl_team_Sunrisers_Hyderabad', 'overs', 'runs', 'wickets',
       'runs_last_5', 'wickets_last_5', 'total']
```

In [20]:
```
encoded_df
```

In [21]:
```
# Splitting the data into train and test set
X_train = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year <= 2016]
X_test = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year >= 2017]
```

In [22]:
```
y_train = encoded_df[encoded_df['date'].dt.year <= 2016]['total'].values
y_test = encoded_df[encoded_df['date'].dt.year >= 2017]['total'].values
```

In [23]:
```
# Removing the 'date' column
X_train.drop(labels='date', axis=True, inplace=True)
X_test.drop(labels='date', axis=True, inplace=True)
```

In [24]:
```
X_train
```

In [25]:
```
y_train
```

Out[25]:
```
array([222, 222, 222, ..., 208, 208, 208], dtype=int64)
```

In [26]:
```
y_test
```

Out[26]:
```
array([207, 207, 207, ..., 107, 107, 107], dtype=int64)
```

One-Hot-Encoding

In [27]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

Linear Regression Model

In [29]:
```
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

```

In [30]:
```
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
# pipe = Pipeline(steps = [
#     ('step1', trf),
#     ('step2', LinearRegression())
# ])
```

In [32]:
```
# pipe.fit(X_train,y_train)
```

In [33]:
```
pred = regressor.predict(X_test)
```

In [34]:
```
pred
```

Out[34]:
```
array([172.25345288, 175.42084361, 174.78783414, ..., 100.8090797 ,
       100.16359499,  93.60084579])
```

Convert pred into data frame

In [35]:
```
actual_value = y_test
```

In [36]:
```
actual_value
```

Out[36]:
```
array([207, 207, 207, ..., 107, 107, 107], dtype=int64)
```

In [37]:
```
prediction1 = pd.DataFrame(pred)
actual_value = pd.DataFrame(actual_value)
```

In [38]:
```
prediction1
```

Concatenate actual_value and prediction1

In [40]:
```
frame1 = [actual_value, prediction1]
```

In [41]:
```
result1 = pd.concat(frame1, axis=1, join='inner')
```

In [42]:
```
result1.columns = ['actual_value', 'prediction1']
```

In [43]:
```
result1
```

In [44]:
```
regressor.predict(X_test)[250]
```

Out[44]:
```
174.18118748713826
```

In [45]:
```
from sklearn import metrics
import numpy as np
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE: 12.312732680652855
MSE: 257.5067665759122
RMSE: 16.04701737320404
```

Accuracy of linear regression

In [53]:
```
from sklearn.metrics import r2_score
score1 = r2_score(result1["actual_value"], result1["prediction1"])
print("The accuracy of our Linear Regression model is {}%".format(round(score1, 4) *100))
```

```
The accuracy of our Linear Regression model is 72.16%
```

In [54]:
```
score1
```

Out[54]:
```
0.7216047508395314
```

In [55]:
```
# Creating a pickle file for the classifier
filename = 'first-innings-score-lr-model.pkl'
pickle.dump(regressor, open(filename, 'wb'))
```

Ridge Refression

In [56]:
```
## Ridge Regression
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

In [57]:
```
ridge=Ridge()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regressor.fit(X_train,y_train)
```

```
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:157: LinAlgWarning: Ill-conditioned matrix
(rcond=1.43483e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:157: LinAlgWarning: Ill-conditioned matrix
(rcond=1.4416e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:157: LinAlgWarning: Ill-conditioned matrix
(rcond=1.34346e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:157: LinAlgWarning: Ill-conditioned matrix
(rcond=1.45125e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:157: LinAlgWarning: Ill-conditioned matrix
(rcond=1.34628e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True, overwrite_a=True).T
```

Out[57]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
                                   20, 30, 35, 40]},
             scoring='neg_mean_squared_error')
```

In [58]:
```
print(ridge_regressor.best_params_)
print(ridge_regressor.best_score_)
```

```
{'alpha': 40}
-328.65931292513926
```

In [59]:
```
prediction2=ridge_regressor.predict(X_test)
```

In [60]:
```
prediction2
```

Out[60]:
```
array([172.21538629, 175.38525289, 174.75053841, ..., 100.85862453,
       100.20958678,  93.64881386])
```

In [61]:
```
prediction2 = pd.DataFrame(prediction2)
```

In [62]:
```
prediction2
```

Concatenate actual_value and prediction2

In [63]:
```
frame2 = [actual_value, prediction2]
```

In [64]:
```
result2 = pd.concat(frame2, axis=1, join='inner')
```

In [65]:
```
result2.columns = ['actual_value', 'prediction2']
```

In [67]:
```
ridge_regressor.predict(X_test)[69]
```

Out[67]:
```
200.2563136470916
```

In [68]:
```
# import seaborn as sns
# sns.distplot(y_test-prediction2)
```

In [69]:
```
from sklearn import metrics
import numpy as np
print('MAE:', metrics.mean_absolute_error(y_test, prediction2))
print('MSE:', metrics.mean_squared_error(y_test, prediction2))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction2)))
```

```
MAE: 12.298881515325943
MSE: 257.11140659583253
RMSE: 16.034693841661976
```

Accuracy of Ridge Regression

In [70]:
```
from sklearn.metrics import r2_score
score2 = r2_score(result2["actual_value"], result2["prediction2"])
print("The accuracy of our Ridge Regression model is {}%".format(round(score2, 4) *100))
```

```
The accuracy of our Ridge Regression model is 72.2%
```

In [71]:
```
score2
```

Out[71]:
```
0.722032181705236
```

Lasso Regression

In [72]:
```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
```

In [73]:
```
lasso=Lasso()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40]}
lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)

lasso_regressor.fit(X_train,y_train)
print(lasso_regressor.best_params_)
print(lasso_regressor.best_score_)
```

```
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 4.788e+06, tolerance: 2.647e+03
  model = cd_fast.enet_coordinate_descent(
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 4.632e+06, tolerance: 2.608e+03
  model = cd_fast.enet_coordinate_descent(
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 4.875e+06, tolerance: 2.842e+03
  model = cd_fast.enet_coordinate_descent(
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 4.780e+06, tolerance: 2.898e+03
  model = cd_fast.enet_coordinate_descent(
C:\Users\sachin kumar ray\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 4.646e+06, tolerance: 2.742e+03
```

In [74]:
```
prediction3=lasso_regressor.predict(X_test)
```

In [75]:
```
prediction3
```

Out[75]:
```
array([170.87637489, 174.29122344, 173.47838988, ..., 106.50735785,
       105.44371993,  99.57836569])
```

In [76]:
```
prediction3 = pd.DataFrame(prediction3)
```

In [77]:
```
prediction3
```

Concatenate actual_value and prediction3

In [78]:
```
frame3 = [actual_value, prediction3]
```

In [79]:
```
result3 = pd.concat(frame3, axis=1, join='inner')
```

In [80]:
```
result3.columns = ['actual_value', 'prediction3']
```

In [81]:
```
result3
```

In [82]:
```
lasso_regressor.predict(X_test)[250]
```

Out[82]:
```
170.64262687974497
```

In [83]:
```
y_test
```

Out[83]:
```
array([207, 207, 207, ..., 107, 107, 107], dtype=int64)
```

In [84]:
```
# import seaborn as sns
# sns.distplot(y_test-prediction3)
```

In [85]:
```
from sklearn import metrics
import numpy as np
print('MAE:', metrics.mean_absolute_error(y_test, prediction3))
print('MSE:', metrics.mean_squared_error(y_test, prediction3))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction3)))
```

```
MAE: 12.346651664171699
MSE: 263.73897753355544
RMSE: 16.240042411692016
```

Accuracy of Lasso Regression

In [86]:
```
from sklearn.metrics import r2_score
score3 = r2_score(result3["actual_value"], result3["prediction3"])
print("The accuracy of our Lasso Regression model is {}%".format(round(score3, 4) *100))
```

```
The accuracy of our Lasso Regression model is 71.49%
```

A.2.2. MODEL: 2 IPL WIN PREDICTION

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
match = pd.read_csv('matches.csv')
delivery = pd.read_csv('deliveries.csv')
```

In [3]:
```
match.head()
```

Out[3]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_m
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj S
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD S
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Ma
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Ja

In [4]:
```
match['team1'].unique()
```

Out[4]:
```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Daredevils', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Rising Pune Supergiants',
       'Delhi Capitals'], dtype=object)
```

In [5]:
```
match['team1'] = match['team1'].str.replace('Delhi Daredevils', 'Delhi Capitals')
match['team2'] = match['team2'].str.replace('Delhi Daredevils', 'Delhi Capitals')

match['team1'] = match['team1'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')
match['team2'] = match['team2'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')

match['team1'] = match['team1'].str.replace('Gujarat Lions', 'Gujarat Titans')
match['team2'] = match['team2'].str.replace('Gujarat Lions', 'Gujarat Titans')

match['team1'] = match['team1'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')
match['team2'] = match['team2'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')
```

In [6]:
```
delivery.head()
```

Out[6]:

In [7]:
```
delivery['batting_team'].unique()
```

Out[7]:
```
array(['Sunrisers Hyderabad', 'Royal Challengers Bangalore',
       'Mumbai Indians', 'Rising Pune Supergiant', 'Gujarat Lions',
       'Kolkata Knight Riders', 'Kings XI Punjab', 'Delhi Daredevils',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Rising Pune Supergiants',
       'Delhi Capitals'], dtype=object)
```

In [8]:
```
delivery['batting_team'] = delivery['batting_team'].str.replace('Delhi Daredevils', 'Delhi Capitals')
delivery['bowling_team'] = delivery['bowling_team'].str.replace('Delhi Daredevils', 'Delhi Capitals')

delivery['batting_team'] = delivery['batting_team'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')
delivery['bowling_team'] = delivery['bowling_team'].str.replace('Deccan Chargers', 'Sunrises Hyderabad')

delivery['batting_team'] = delivery['batting_team'].str.replace('Gujarat Lions', 'Gujarat Titans')
delivery['bowling_team'] = delivery['bowling_team'].str.replace('Gujarat Lions', 'Gujarat Titans')

delivery['batting_team'] = delivery['batting_team'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')
delivery['bowling_team'] = delivery['bowling_team'].str.replace('Rising Pune Supergiants', 'Lucknow Super Giants')
```

In [9]:
```
match.head()
```

Out[9]:

In [10]:
```
delivery.head()
```

Out[10]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_runs	legbye_runs	noball_runs	penalty_runs
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0

5 rows × 21 columns

In [11]:
```
match.shape
```

Out[11]:
```
(756, 18)
```

In [12]:
```
delivery.groupby(['match_id', 'inning',]).sum()['total_runs']
```

Out[12]:
```
match_id  inning
1         1         207
          2         172
2         1         184
          2         187
3         1         183
                   ...
11413     2         170
11414     1         155
          2         162
11415     1         152
          2         157
Name: total_runs, Length: 1528, dtype: int64
```

To convert this into data frame we use "reset_index()" method

In [13]:
```
total_score_df = delivery.groupby(['match_id', 'inning',]).sum()['total_runs'].reset_index()
```

WE add +1 because chased run is 1 more than the actual run and we are chasing

In [14]:
```
total_score_df['total_runs'] = total_score_df['total_runs'] + 1
```

In [15]:
```
total_score_df
```

Out[15]:

In [16]:
```
total_score_df = total_score_df[total_score_df['inning'] == 1]
```

In [17]:
```
total_score_df
```

Out[17]:

	match_id	inning	total_runs
0	1	1	208
2	2	1	185
4	3	1	184
6	4	1	164
8	5	1	158
...	...	...	...
1518	11347	1	144
1520	11412	1	137
1522	11413	1	172
1524	11414	1	156
1526	11415	1	153

756 rows × 3 columns

In [18]:
```
match_df = match.merge(total_score_df[['match_id', 'total_runs']], left_on = 'id', right_on = 'match_id')
```

In [19]:
```
match_df.head()
```

Out[19]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_m
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj S
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD S
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Titans	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Ma
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Capitals	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Ja

In [20]:
```
match_df['team1'].unique()
```

Out[20]:
```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Titans',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Capitals', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Sunrises Hyderabad',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Lucknow Super Giants'],
      dtype=object)
```

In [21]:
```
match_df['team2'].unique()
```

Out[21]:
```
array(['Royal Challengers Bangalore', 'Rising Pune Supergiant',
       'Kolkata Knight Riders', 'Kings XI Punjab', 'Delhi Capitals',
       'Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Titans',
       'Rajasthan Royals', 'Chennai Super Kings', 'Sunrises Hyderabad',
       'Pune Warriors', 'Kochi Tuskers Kerala', 'Lucknow Super Giants'],
      dtype=object)
```

In [22]:
```
teams = [
    'Sunrisers Hyderabad',
    'Mumbai Indians',
    'Gujarat Titans',
    'Royal Challengers Bangalore',
    'Kolkata Knight Riders',
    'Kings XI Punjab',
    'Chennai Super Kings',
    'Rajasthan Royals',
    'Delhi Capitals',
    'Lucknow Super Giants'
]
```

In [23]:
```
match_df.head()
```

In [24]:
```
match_df = match_df[match_df['team1'].isin(teams)]
match_df = match_df[match_df['team2'].isin(teams)]
```

In [25]:
```
match_df
```

In [26]:
```
match_df.shape
```

Out[26]:
```
(611, 20)
```

Remove that match in which dl_method is applied

In [27]:
```
match_df['dl_applied'].value_counts()
```

Out[27]:
```
0    593
1     18
Name: dl_applied, dtype: int64
```

In [28]:
```
match_df = match_df[match_df['dl_applied'] == 0]
```

In [29]:
```
match_df = match_df[['match_id', 'city', 'winner', 'total_runs']]
```

join it with delivery dataset

In [30]:
```
delivery_df = match_df.merge(delivery, on = 'match_id')
```

In [31]:
```
delivery_df
```

In [32]:
```
delivery_df['batting_team'].unique()
```

Out[32]:
```
array(['Sunrisers Hyderabad', 'Royal Challengers Bangalore',
       'Gujarat Titans', 'Kolkata Knight Riders', 'Delhi Capitals',
       'Mumbai Indians', 'Kings XI Punjab', 'Chennai Super Kings',
       'Rajasthan Royals', 'Lucknow Super Giants'], dtype=object)
```

In delivery data set, we need only that value where inning == 2

In [33]:
```
delivery_df = delivery_df[delivery_df['inning'] == 2]
```

In [34]:
```
delivery_df
```

In [36]:
```
delivery_df['current_score'] = delivery_df.groupby('match_id').cumsum()['total_runs_y']
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\3025607697.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['current_score'] = delivery_df.groupby('match_id').cumsum()['total_runs_y']
```

In [37]:
```
delivery_df['current_score']
```

Out[37]:
```
125         1
126         1
127         1
128         3
129         7
          ...
141509    152
141510    154
141511    155
141512    157
141513    157
Name: current_score, Length: 68415, dtype: int64
```

Now we add a new column to find run left after each delivery

Runs_left = total_runs - current_runs

In [38]:
```
delivery_df['runs_left'] = delivery_df['total_runs_x'] - delivery_df['current_score']
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\1162889266.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

In [40]:
```
delivery_df['balls_left'] = 120 - ((delivery_df['over']-1)*6 + delivery_df['ball'])
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\1633346659.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['balls_left'] = 120 - ((delivery_df['over']-1)*6 + delivery_df['ball'])
```

In [41]:
```
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna("0")
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].apply(lambda x:x if x == "0" else "1")
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\2400862059.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna("0")
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\2400862059.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['player_dismissed'] = delivery_df['player_dismissed'].apply(lambda x:x if x == "0" else "1")
```

In [42]:
```
delivery_df
```

In [43]:
```
delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype('int')
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\1121782170.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype('int')
```

In [44]:
```
wickets = delivery_df.groupby('match_id').cumsum()['player_dismissed'].values
```

In [45]:
```
delivery_df['wicket_left'] = 10 - wickets
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\2870077134.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['wicket_left'] = 10 - wickets
```

In [46]:
```
delivery_df
```

In [47]:
```
delivery_df['crr'] = (delivery_df['current_score']*6)/(120 - delivery_df['balls_left'])
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\2048085250.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['crr'] = (delivery_df['current_score']*6)/(120 - delivery_df['balls_left'])
```

To find required run_rate

rrr = runs left/ over left

In [48]:
```
delivery_df['rrr'] = (delivery_df['runs_left']*6)/delivery_df['balls_left']
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\2241449921.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['rrr'] = (delivery_df['runs_left']*6)/delivery_df['balls_left']
```

In [49]:
```
delivery_df
```

In [50]:
```
def result(row):
    return 1 if row['batting_team'] == row['winner'] else 0
```

In [51]:
```
delivery_df['result'] = delivery_df.apply(result, axis = 1)
```

```
C:\Users\sachin kumar ray\AppData\Local\Temp\ipykernel_57640\140305760.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  delivery_df['result'] = delivery_df.apply(result, axis = 1)
```

In [52]:
```
delivery_df
```

In [53]:
```
final_df = delivery_df[['batting_team', 'bowling_team', 'city', 'runs_left', 'balls_left', 'wicket_left', 'total_runs_x', 'c
```

In [54]:
```
final_df
```

Out[54]:

	batting_team	bowling_team	city	runs_left	balls_left	wicket_left	total_runs_x	crr	rrr	result
125	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	119	10	208	6.000000	10.436975	0
126	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	118	10	208	3.000000	10.525424	0
127	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	117	10	208	2.000000	10.615385	0
128	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	205	116	10	208	4.500000	10.603448	0
129	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	201	115	10	208	8.400000	10.486957	0
...	...	...	...	...	...	...	...	...	...	...
141509	Chennai Super Kings	Mumbai Indians	Hyderabad	1	4	5	153	7.862069	1.500000	0
141510	Chennai Super Kings	Mumbai Indians	Hyderabad	-1	3	5	153	7.897436	-2.000000	0
141511	Chennai Super Kings	Mumbai Indians	Hyderabad	-2	2	4	153	7.881356	-6.000000	0
141512	Chennai Super Kings	Mumbai Indians	Hyderabad	-4	1	4	153	7.915966	-24.000000	0
141513	Chennai Super Kings	Mumbai Indians	Hyderabad	-4	0	3	153	7.850000	-inf	0

68415 rows × 10 columns

In [55]:
```
final_df['batting_team'].unique()
```

Out[55]:
```
array(['Royal Challengers Bangalore', 'Kolkata Knight Riders',
       'Delhi Capitals', 'Sunrisers Hyderabad', 'Mumbai Indians',
       'Kings XI Punjab', 'Gujarat Titans', 'Rajasthan Royals',
       'Chennai Super Kings', 'Lucknow Super Giants'], dtype=object)
```

In [55]:
```
final_df['batting_team'].unique()
```

Out[55]:
```
array(['Royal Challengers Bangalore', 'Kolkata Knight Riders',
       'Delhi Capitals', 'Sunrisers Hyderabad', 'Mumbai Indians',
       'Kings XI Punjab', 'Gujarat Titans', 'Rajasthan Royals',
       'Chennai Super Kings', 'Lucknow Super Giants'], dtype=object)
```

In [56]:
```
final_df['batting_team'].unique()
```

Out[56]:
```
array(['Royal Challengers Bangalore', 'Kolkata Knight Riders',
       'Delhi Capitals', 'Sunrisers Hyderabad', 'Mumbai Indians',
       'Kings XI Punjab', 'Gujarat Titans', 'Rajasthan Royals',
       'Chennai Super Kings', 'Lucknow Super Giants'], dtype=object)
```

Here we get our resultant dataset according to our use

Now shuffle the data so thast there is no biasesing in the model

In [57]:
```
final_df = final_df.sample(final_df.shape[0])
```

For any random match

In [58]:
```
final_df.sample()
```

In [61]:
```
final_df.dropna(inplace = True)
```

In [62]:
```
final_df = final_df[final_df['balls_left'] != 0]
```

In [63]:
```
final_df
```

Divide into independent and dependent variable

In [65]:
```
X = final_df.iloc[:, :-1]
Y = final_df.iloc[:, -1]
```

split into training set

In [66]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y, test_size = 0.2, random_state = 1)
```

In [71]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [72]:
```
trf = ColumnTransformer([
    ('trf', OneHotEncoder(sparse = False, drop = 'first'), ['batting_team', 'bowling_team', 'city'])
], remainder = 'passthrough')
```

Import Logistic regression and pipeline

In [73]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
```

Create a pipe object

In [74]:
```
pipe_ipl = Pipeline(steps = [
    ('step1', trf),
    ('step2', LogisticRegression(solver = 'liblinear'))
])
```

Feed the data

In [75]:
```
pipe_ipl.fit(X_train, Y_train)
```

Out[75]:
```
Pipeline(steps=[('step1',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('trf',
                                                  OneHotEncoder(drop='first',
                                                                sparse=False),
                                                  ['batting_team',
                                                   'bowling_team', 'city'])])),
                ('step2', LogisticRegression(solver='liblinear'))])
```

Predict the value

In [76]:
```
Y_pred = pipe_ipl.predict(X_test)
```

In [77]:
```
Y_pred
```

Out[77]:
```
array([0, 1, 0, ..., 0, 1, 1], dtype=int64)
```

Accuracy score

In [78]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, Y_pred)
```

Out[78]:
```
0.8267646403918949
```

Prediction

In [79]:
```
pipe_ipl.predict_proba(X_test)[250]
```

Out[79]:
```
array([0.66262169, 0.33737831])
```

we fit the model with a whole match scanerio, logistic regression gives bettrr probablity so we feed the 2nd inning data

This will give probabilty of winning and lossiing a match at the end of every over

In [80]:
```
def match_summary(row):
    print("Batting Team-" + row['batting_team'] + " | Bowling Team-" + row['bowling_team'] + " | Target- " + str(row['total_
```

In [81]:
```
def match_progression(x_df,match_id,pipe):
    match = x_df[x_df['match_id'] == match_id]
    match = match[(match['ball'] == 6)]
    temp_df = match[['batting_team','bowling_team','city','runs_left','balls_left','wicket_left','total_runs_x','crr','rrr']]
    temp_df = temp_df[temp_df['balls_left'] != 0]
    result = pipe.predict_proba(temp_df)
    temp_df['lose'] = np.round(result.T[0]*100,1)
    temp_df['win'] = np.round(result.T[1]*100,1)
    temp_df['end_of_over'] = range(1,temp_df.shape[0]+1)

    target = temp_df['total_runs_x'].values[0]
    runs = list(temp_df['runs_left'].values)
    new_runs = runs[:]
    runs.insert(0,target)
    temp_df['runs_after_over'] = np.array(runs)[:-1] - np.array(new_runs)
    wickets = list(temp_df['wicket_left'].values)
    new_wickets = wickets[:]
    new_wickets.insert(0,10)
    wickets.append(0)
    w = np.array(wickets)
    nw = np.array(new_wickets)
    temp_df['wickets_in_over'] = (nw - w)[0:temp_df.shape[0]]

    print("Target-",target)
    temp_df = temp_df[['end_of_over','runs_after_over','wickets_in_over','lose','win']]
    return temp_df,target

```

In [82]:
```
temp_df,target = match_progression(delivery_df,74,pipe_ipl)
temp_df
```

```
Target- 179
```

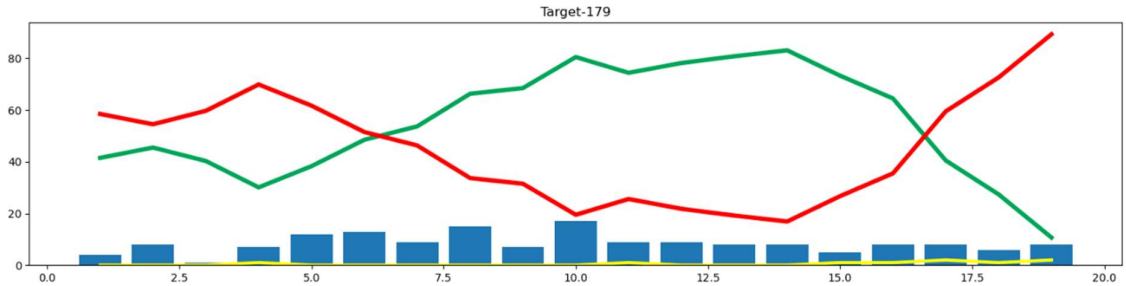
Out[82]:

	end_of_over	runs_after_over	wickets_in_over	lose	win
12399	1	4	0	59.7	40.3
12407	2	8	0	55.6	44.4
12413	3	1	0	60.6	39.4
12419	4	7	1	70.5	29.5
12425	5	12	0	62.4	37.6
12431	6	13	0	52.2	47.8
12437	7	9	0	47.1	52.9
12445	8	15	0	34.5	65.5
12451	9	7	0	32.3	67.7
12458	10	17	0	20.2	79.8
12464	11	9	1	26.2	73.8
12470	12	9	0	22.3	77.7
12476	13	8	0	19.7	80.3
12482	14	8	0	17.4	82.6
12488	15	5	1	27.1	72.9
12495	16	8	1	35.9	64.1
12501	17	8	2	59.5	40.5
12507	18	6	1	72.8	27.2
12513	19	8	2	89.6	10.4

In [83]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(18,8))
plt.plot(temp_df['end_of_over'],temp_df['wickets_in_over'],color='yellow',linewidth=3)
plt.plot(temp_df['end_of_over'],temp_df['win'],color='#00a65a',linewidth=4)
plt.plot(temp_df['end_of_over'],temp_df['lose'],color='red',linewidth=4)
plt.bar(temp_df['end_of_over'],temp_df['runs_after_over'])
plt.title('Target-' + str(target))
```

Out[83]: Text(0.5, 1.0, 'Target-179')

Out[83]: Text(0.5, 1.0, 'Target-179')



In [84]:
```
teams
```

Out[84]:
```
['Sunrisers Hyderabad',
 'Mumbai Indians',
 'Gujarat Titans',
 'Royal Challengers Bangalore',
 'Kolkata Knight Riders',
 'Kings XI Punjab',
 'Chennai Super Kings',
 'Rajasthan Royals',
 'Delhi Capitals',
 'Lucknow Super Giants']
```

In [85]:
```
delivery_df['city'].unique()
```

Out[85]:
```
array(['Hyderabad', 'Rajkot', 'Bangalore', 'Mumbai', 'Indore', 'Kolkata',
       'Delhi', 'Chandigarh', 'Kanpur', 'Jaipur', 'Chennai', 'Cape Town',
       'Port Elizabeth', 'Durban', 'Centurion', 'East London',
       'Johannesburg', 'Kimberley', 'Bloemfontein', 'Ahmedabad',
       'Dharamsala', 'Pune', 'Raipur', 'Ranchi', 'Abu Dhabi', 'Sharjah',
       nan, 'Cuttack', 'Visakhapatnam', 'Mohali', 'Bengaluru'],
      dtype=object)
```

In [86]:
```
import pickle
pickle.dump(pipe_ipl,open('pipe_ipl.pkl','wb'))
```

Published Paper **APPENDIX-B**

Prediction of IPL Match Score and Winner Using Machine Learning Algorithms

SACHIN KUMAR RAY

B.Tech. Student, Department of Artificial Intelligence & Machine Learning,
Dr. Akhilesh Das Gupta Institute of Technology & Management, New Delhi,
Delhi, India

ABSTRACT

This study explains the application of machine learning to sports. According to recent changes in data science and sports, the use of sports-based machine learning and data mining shows the importance of process in outcome performance and prediction. The purpose of this paper is to evaluate current measurements used in the literature to understand the estimation methods used to model and analyse data and characterize the variables that govern performance. Finally, this article will present a reliable tool for data analysis using machine learning.

In today's world, sports produce enough statistical information about every player, team, match, and season. The first sports researchers were thought to be experts, coaches, team managers and analysts. Sports organizations have recently realized that there is research in their data and want to do research using different data mining methods. Sports data helps coaches and managers in many ways, such as predicting results, analysing player performance, skills, and evaluating strategies. Forecasts help managers and organizations make decisions to win teams and competitions. The present study shows that preliminary studies of data mining systems can predict outcomes and evaluate the strengths and weaknesses of each system. Predictions are made for each match. Although in many respects this application is very limited. It is very important to examine the use of machine learning in these situations to see if the application can provide better results in analysis.

This research aims to provide solutions that will help make predictions more accurate and precise than previous methods, using more accurate data and machine learning.

Keywords:

"IPL score prediction," "machine learning," "data analysis," "predictive modelling," "cricket analytics," "Linear Regression," "Logistic Regression," "Ridge Regression," "Lasso Regression," "Random Forest Classifier," or "sports analytics."

INTRODUCTION

Machine learning is a branch of artificial intelligence in computer science that uses statistical techniques to allow computers to "learn" from data without being explicitly programmed.

Simultaneous advances in computer technology, big data and theoretical understanding, ML techniques redefined in the 21st century and ML ideas have become an important part of the technology industry, computer science, software that help to solve many of engineering and research studies difficult problem.

The main purpose of is to determine the importance of influencing the match results and to select the best machine learning model that fits the data and performs the best results. Some projects have been published in the area of predicting cricket matches. Due to use of few essential factor, the accuracy is lower. However, in other studies Machine learning model was wrong. Therefore, it is very important that the considers all the important factors that will affect the results of the match and the best model for the training and analyses the data. This will improve the accuracy of outcomes.

Many research papers have been published and completed over the years using supervised machine learning to predict the outcome of cricket matches. Algorithms such as linear regression, support vectors machine, logistic regression, Decision trees, Bayesian Networks, and random forests. Cricket is a sport played worldwide and there is a lot of fans following of IPL because overseas player also played IPL. As fans watching the IPL, people make their own predictions while watching a particular match, based on the data they have they make a call on who will win the match by using different statistics and records. So, there is a huge demand for the algorithms that predicts the best result of score and winning team that is more important. We will perform prediction for all the matches that have taken place in the IPL. This is done by using machine learning algorithms for performing the prediction of the results of the matches.

LITERATURE SURVEY

The Indian Premier League (IPL) is a popular professional Twenty20 cricket league in India. Predicting the scores of IPL matches is a challenging task due to the complex nature of the game. Machine learning techniques have been widely applied to predict the outcomes and scores of IPL matches. This literature survey aims to provide an overview of the existing research and methodologies employed for IPL score prediction using machine learning.

Sharma, A., & Saini, R. (2018). [1] presented an approach for IPL score prediction using multiple machine learning algorithms such as decision trees, random forests, and support vector machines (SVM). The authors compared the performance of these algorithms and evaluate their accuracy in predicting the scores of IPL matches.

Kulkarni, P., & Gokhale, A. (2019) [2] presented an ensemble learning approach that combines multiple machine learning algorithms for IPL score prediction. The authors experiment with various classifiers, including k-nearest neighbours (KNN), SVM, and logistic regression. They evaluate the performance of the ensemble model and compare it with individual algorithms.

Singh, H., & Grover, A. (2020) [3] presented a research focuses on the combination of long short-term

memory (LSTM) and random forest regression for IPL score prediction. The authors propose a hybrid model that leverages the sequence modeling capabilities of LSTM and the ensemble learning approach of random forests. The performance of the proposed model is evaluated and compared with other techniques.

Jaiswal, A., & Singh, S. (2021). [4] explores the application of the XGBoost algorithm, a gradient boosting technique, for IPL score prediction. The authors describe the feature selection process and evaluate the performance of the XGBoost model. They also compare its results with other machine learning algorithms.

Gupta, M., & Khandelwal, M. (2021). [5] investigates the use of deep learning techniques, specifically convolutional neural networks (CNNs), for IPL score prediction. The authors propose a CNN-based model that considers various features related to teams, players, and match conditions. They evaluate the performance of the model and discuss its potential for accurate score prediction.

IPL score prediction using machine learning techniques has gained significant attention from researchers. The literature survey showcases various approaches, including ensemble learning, LSTM, XG Boost, and deep learning techniques, for predicting IPL scores. The performance of these models is evaluated and compared using different evaluation metrics. Further research in this domain can explore the incorporation of additional features, real-time data, and more advanced machine learning algorithms to enhance the accuracy of IPL score prediction.

METHODOLOGIES

Data Collection and exploration

Data gathering is the fundamental module and the first stage of the project. The main aim is to collect appropriate dataset that is suitable for our needs and Data exploration is the first step of data analysis used to explore and visualize data to uncover insights from the start or identify areas or patterns to dig into more. The goal of data exploration is to learn about characteristics and potential problems of a data set without the need to formulate assumptions about the data beforehand. We take the data set match.csv, deliveries.csv and ipl.csv from Kaggle.

Data pre-processing

Machine learning relies heavily on data pre-processing to get highly accurate and insightful outputs so that we can apply that data on our model. A data set contain a lot of data which is not of our use so we do data pre-processing and convert data into a systematic form as of our need. The more reliable the produced results are, the better the data quality is. Realworld datasets are characterized by incomplete, noisy, and inconsistent data. Data pre-processing improves data quality by filling in missing or partial data, reducing noise, and addressing discrepancies.

Data cleaning

The process of eliminating errors and replacing them with genuine values is known as data cleaning. The data sets gathered contain noisy data, such as null values and inappropriate values, which must be cleaned.

As a result, the data is cleaned by replacing null values with zeros, and the data is organized into correct columns so that we can properly analyse it.

Data visualization

The data that has been gathered is used to visualize the information for better comprehension. The Matplotlib and seaborn Library were used to visualize the graphs for team wins in various cities based on their venues and player strike rate.

Splitting into train and test

Train test split is a technique of splitting dataset into two parts that is used to estimate the performance of machine learning model by feeding the train data to the model and predict the data on test data and new data. A train test is the way of structuring your machine learning project so that you can test your hypothesis quickly and inexpensively. Basically, it's a way to divide the training data so that you can try your algorithm to one half and evaluate the result on the other half. A train test split is when you split your data into a training set and a testing set. The training set is used for training the model, and the testing set is used to test your model. This allows us to train our models on the training data set, and then test their accuracy on the unseen testing set. There are a few different ways to do a train test split, but the most common is to simply split your data into two sets. For example, 80% for training and 20% for testing.

Algorithms used

Linear Regression

Linear regression is a statistical modelling technique used to establish a linear relationship between a dependent variable and one or more independent variables. It aims to fit a straight line to the data points in such a way that it minimizes the difference between the predicted values and the actual values.

The general form of a linear regression model is represented by the equation:

$y = b0 + b1x1 + b2x2 + ... + bn*xn$

where:

y is the dependent variable or the target variable that we want to predict.

b0 is the intercept or the constant term.

b1, b2, ..., bn are the coefficients or slopes associated with the independent variables x1, x2, ..., xn.

x1, x2, ..., xn are the independent variables.

It aims to establish a linear relationship between independent variables (predictors) and a dependent variable (target) to make predictions. By analysing historical data and training a linear regression model, one can make predictions about the scores of future IPL matches, providing valuable insights for teams, analysts, and cricket enthusiasts.[7].

Logistic Regression

Logistic regression is one of the most useful machine learning algorithms based a statistical modelling technique that predict the dependent data variable by analysing the relationship between one or more independent variable. It is commonly used for binary classification problems such as yes or no, 0 or 1, True

or False, while it is not typically used directly for predicting scores in IPL matches, it can be employed for predicting the likelihood of a team winning or losing a match based on various factors. In this context, logistic regression can be utilized to estimate the probability of a team winning an IPL match. Logistic regression can provide insights into the likelihood of winning an IPL match based on historical data and relevant features.[6].

Random Forest Classifier

Random Forest Classifier is a machine learning algorithm that can be utilized for predicting the winning outcome of IPL matches. It is an ensemble learning method that combines multiple decision trees to make predictions. Random Forest Classifier is advantageous in IPL winning prediction because it can handle non-linear relationships and interactions among features, handle high-dimensional datasets, and mitigate overfitting issues. It also provides insights into feature importance, allowing for a better understanding of the factors influencing match outcomes. It's important to note that the performance of the Random Forest Classifier can be further optimized by tuning hyperparameters such as the number of decision trees, the depth of the trees, and the number of features considered at each split. It gives the one-sided prediction at every point that is predicted on data but for real situation this type of prediction is not of our use.

Lasso Regression

Lasso Regression, also known as L1 regularization, is a linear regression technique that incorporates a penalty term to encourage sparse solutions by shrinking the coefficients of less important features to zero. While Lasso Regression is not commonly used for direct IPL score prediction, it can be applied to select important features that influence the score and improve the model's performance. Lasso Regression helps in feature selection by highlighting the most influential features in predicting the IPL scores. By reducing the coefficients of less important features to zero, it provides a sparse solution and simplifies the model.[9].

Ridge Regression

Ridge Regression is a linear regression technique that includes a regularization term called L2 regularization. It is commonly used for predicting IPL scores by minimizing the sum of squared errors while also shrinking the regression coefficients. Ridge Regression is a useful technique for IPL score prediction by incorporating regularization to prevent overfitting and shrink the coefficients.[8]

MODEL

Predicting sports scores, such as IPL cricket scores, using machine learning models can be an interesting and challenging task. Here is a general outline of the steps involved in building a machine learning model for IPL sports score prediction:

Data Collection: Gather historical match data, including features such as team performance indicators, player statistics, venue, weather conditions, and other relevant factors. Ensure the dataset encompasses a significant number of matches for robust analysis.

Data Pre-processing: Clean and pre-process the collected data, handling missing values, outliers, and categorical variables as necessary. Normalize or scale the features to ensure they are on a similar scale for

effective modelling.

Data Partitioning: Split the dataset into training, validation, and testing sets. The training set will be used to train the model, the validation set will be used for hyperparameter tuning and model selection, and the testing set will be used to evaluate the final performance.

Model Selection: Choose an appropriate machine learning algorithm for sports score prediction. Depending on the nature of the problem and the characteristics of the data, algorithms such as regression models such as linear regression, random forest regression, logistic regression, lasso regression and ridge regression can be considered.

Model Evaluation: Evaluate the trained model using the validation set. Measure its performance using evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), or other relevant metrics specific to sports prediction. Assess how well the model predicts the sports scores compared to the actual scores.

Model	MSE	RMSE	MAE	Accuracy
near Regression	257.50	16.04	12.31	72.16
dge Regression	257.11	16.03	12.29	72.20
sso regression	263.73	16.24	12.34	71.49

Table: 3. Model Evaluation for IPL FIRST INNINGS SCORE PREDICTION

Model	Accuracy
Logistic Regression	83.10

Table: 4 Model Evaluation for 2nd INNINGS WIN PREDICTION

Final Model Evaluation: Once the model is optimized, evaluate its performance using the testing set, which represents unseen data. Calculate the performance metrics to assess the model's predictive ability and determine its effectiveness in predicting sports scores.

Deployment and Prediction: Deploy the final model to make predictions on new, unseen sports match data. Provide the relevant features for each match as input to the model, and it will predict the scores based on the learned patterns and relationships from the training process.

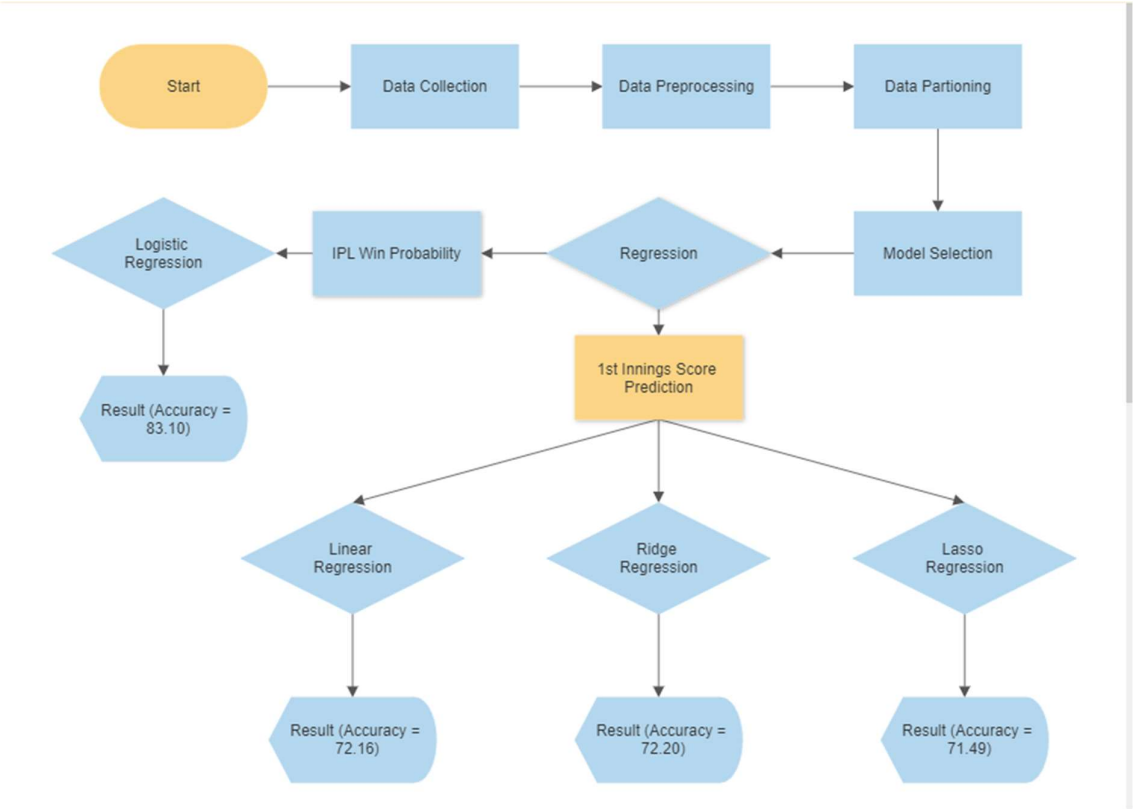


Fig: 5. Flow Chart of the Model

IMPLEMENTATION

The Graphical User Interface (GUI) has been developed for this machine learning models are using the Python Streamlit Library Framework.

Streamlit is an open-source Python library that allows you to create interactive web applications for data science and machine learning projects. It simplifies the process of building and sharing data-focused apps by providing an intuitive and easy-to-use interface.

Here's an overview of how to use Streamlit for your IPL score prediction project:

Installation: Install Streamlit using pip by running the following command in your terminal:

Create a Python Script: Create a new Python script (e.g., app.py) to write your Streamlit application code.

Import Dependencies: In the app.py script, import the necessary dependencies, including Streamlit and any other libraries required for data processing and prediction.

Define the App: Write the Streamlit code to define the structure and behavior of your app.

Run the App: In your terminal, navigate to the directory containing app.py and run the Streamlit app using the following command: Streamlit run app.py

Test and Interact with the App: Once the app is running, open your web browser and navigate to the local URL provided by Streamlit (e.g., http://localhost:8501). You can now interact with the app, inputting the relevant details and exploring the predicted scores.

Continuously Improve and Enhance: Iterate on your Streamlit app based on user feedback, add visualizations, improve the UI, or incorporate additional features to make it more robust and user-friendly.

Streamlit automatically reloads the app whenever you save changes to the app.py file, allowing for quick iterations and updates. It provides a simple yet powerful way to showcase and share your IPL score prediction model with others.

First, on our home page, we get intro about our model and on the side of page a navigation bar is present. There are options to select which type of prediction we want.

Fig: 6. Homepage of GUI

In our model, there are two type of prediction, first one is “IPL FIRST INNINGS SCORE PREDICTION” and second one is winning probability of the team who is chasing “IPL WIN PREDICTION”.

After selecting the “IPL FIRST INNINGS SCORE PREDICTION”, There is a page where user have to give the inputs like “Select the batting team”, “Select the bowling team”, “Overs Completed”, “Runs”, “Wicket Out”, “Run Scored in previous 5 overs”, “Wicket Taken in previous 5 overs” to predict the winner of match after clicking on “Predict Probability”.

Fig: 7. IPL FIRST INNINGS SCORE PREDICTION

After selecting the “IPL WINPREDICTION”, There is a page where user have to give the inputs like “Select the batting team”, “Select the bowling team”, “Select host city”, “Target”, “Score”, “Overs Completed”, “Wickets Out”, to predict the winner of match after clicking on “Predict Probability”.

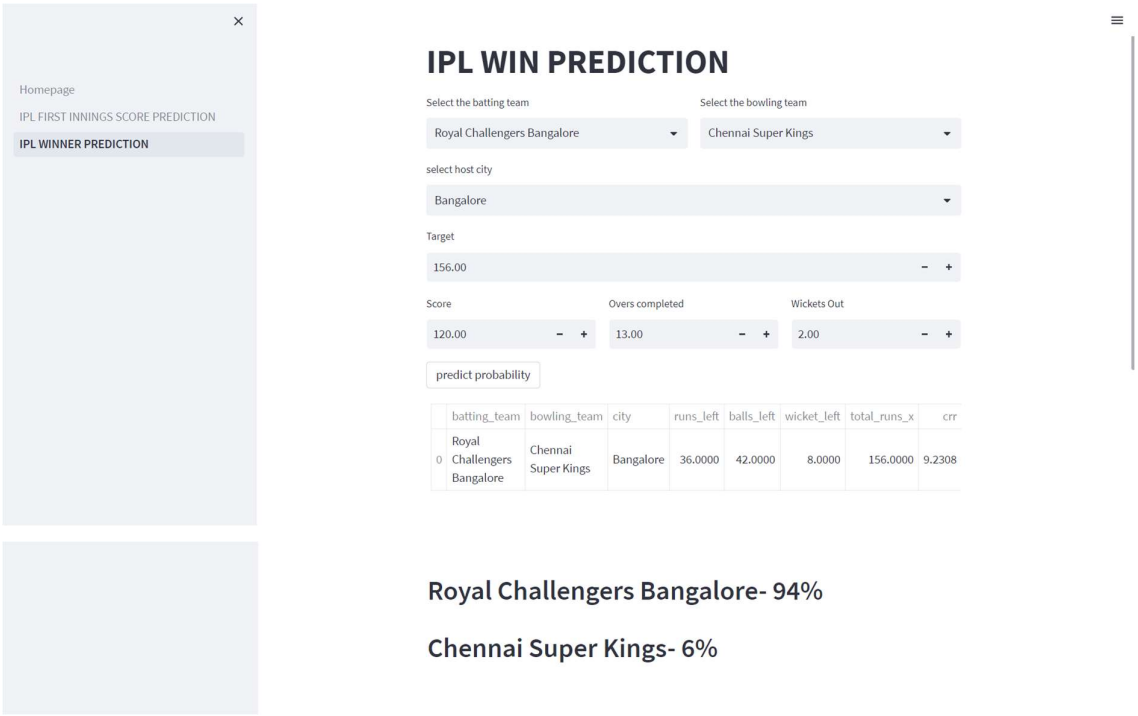


Fig: 8. IPL WIN PREDICTION

CONCLUSION

In conclusion, machine learning models have shown promise in predicting IPL scores and determining the winning team. By leveraging historical match data and relevant features such as player performance, team composition, pitch conditions, and venue, these models can capture patterns and relationships to make informed predictions.

However, it is important to acknowledge that predicting IPL scores and determining the winning team is a challenging task due to the inherent uncertainties and complexities of cricket matches. Factors such as player form, injuries, team strategies, weather conditions, and other unforeseen events can have a significant impact on the outcome.

Machine learning models provide a systematic approach to analyze and extract insights from the available data, but they have limitations. The accuracy of the predictions heavily relies on the quality and relevance of the data, the selection of appropriate features, and the choice of an effective model. Additionally, models need to be regularly updated and refined with new data to adapt to changing trends and dynamics in the sport.

While machine learning can improve prediction accuracy to some extent, it's important to consider other factors such as expert knowledge, match analysis, and contextual information when making IPL score and win predictions. Combining machine learning models with human expertise can lead to more robust and accurate predictions.

Overall, machine learning models serve as valuable tools in the domain of IPL score and win prediction, providing insights and aiding decision-making. However, they should be used in conjunction with other analytical approaches and expert judgment to account for the complex and unpredictable nature of matches.

References

1. Sharma, A., & Saini, R. (2018). IPL score prediction using machine learning techniques. International Journal of Computer Applications, 180(2), 13-17.
2. Kulkarni, P., & Gokhale, A. (2019). Cricket score prediction using ensemble machine learning techniques. International Journal of Advanced Research in Computer Science, 10(5), 220-225.
3. Singh, H., & Grover, A. (2020). Predicting IPL scores using LSTM and random forest regression. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 358-362). IEEE.
4. Jaiswal, A., & Singh, S. (2021). IPL score prediction using XGBoost algorithm. In 2021 International Conference on Information Technology (ICIT) (pp. 1-5). IEEE.
5. Gupta, M., & Khandelwal, M. (2021). IPL score prediction using deep learning techniques. In 2021 International Conference on Innovative Computing and Communication (ICICC) (pp. 1-5). IEEE.
6. Title: "Predicting the Outcome of IPL Matches Using Logistic Regression" Authors: Gupta, A., & Verma, S. Year: 2019 Journal/Conference: International Journal of Computer Sciences and Engineering
7. Title: "IPL Score Prediction Using Machine Learning Techniques" Authors: Singh, A., & Verma, V. Year: 2020 Journal/Conference: 2020 International Conference on Smart Electronics and Communication (ICOSEC)
8. Title: "IPL Score Prediction Using Ridge Regression" Authors: Sharma, S., & Rani, N. Year: 2018 Journal/Conference: 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)
9. Title: "IPL Score Prediction using Machine Learning Techniques: A Comparative Study" Authors: Yadav, N., et al. Year: 2020 Journal/Conference: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)
10. Title: "A Study on IPL Score Prediction Using Machine Learning Techniques" Authors: Sharma, S., et al. Year: 2019 Journal/Conference: 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)
11. Title: "Machine Learning-based Prediction of Indian Premier League (IPL) Match Outcomes" Authors: Kannan, S., et al. Year: 2020 Journal/Conference: 2020 IEEE Region 10 Symposium (TENSYMP)

12. Title: "IPL Score Prediction using Machine Learning Techniques" Authors: Raj, R., et al. Year: 2020 Journal/Conference: 2020 International Conference on Computing, Power and Communication Technologies (GUCON)
13. Rabindra Lamsal and Ayesha Choudhary, “Predicting Outcome of Indian Premier League (IPL) Matches Using Machine Learning”, arXiv:1809.09813 [stat.AP] (September 2018).
14. Singh, P., Agarwal, V., & Goel, P. (2020). IPL Match Result and Score Prediction: An Approach using Machine Learning. In 2020 International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC) (pp. 424-429). IEEE.
15. Kaggle IPL Dataset: Kaggle is a platform that hosts various datasets and machine learning competitions. The Kaggle IPL dataset contains historical data of IPL matches, including ball-by-ball details, player performances, and match outcomes. You can use this dataset to build and train your own IPL score prediction model. You can find the dataset here: https://www.kaggle.com/nowke9/ipldata
16. GitHub repositories: There are several open-source projects and code repositories on platforms like GitHub that provide IPL score prediction models. Searching for "IPL score prediction machine learning" on GitHub will yield multiple repositories with code and examples to guide you.