Predictive Modeling and Team Performance Analysis in IPL

Name: Rohit Moholkar

Student Id: 10613711

Applied Research Project submitted in partial fulfilment of the requirements for the degree of

Masters of Science in Data Analytics

at Dublin Business School

Supervisor: Mr. Vivek Kshirsagar

January 2024

Declaration

‘I declare that this Applied Research Project that I have submitted to Dublin Business School for the award of Masters of Science in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.’

Signed: Rohit Moholkar.

Student Number: 10613711

Date: 08/01/2024

Acknowledgment

I would like to extend my sincere appreciation to my supervisor, Mr. Vivek Kshirsagar, for his invaluable support that paved the way for the completion of this work. His guidance and sagacious advice were instrumental at every step of the process, spanning both the implementation and writing phases.

Moreover, I would like to express my appreciation to the Dublin Business School for granting me access to a wealth of resources and an environment conducive to academic excellence. These factors collectively nurtured my intellectual growth and facilitated a rich learning experience.

Special gratitude is due to my mother, father, sister, and my entire circle of friends for their unwavering support and understanding during the research and writing phases.

In addition, I want to express my heartfelt thanks to God for helping me navigate and overcome the challenges faced during this academic journey. I will continue to trust in Him for the path that lies ahead.

Abstract

This study employs predictive modeling and data analytics to analyze players and team performance in the Indian Premier League (IPL) from 2008 to 2022. The aim is to conduct a thesis on real-world data, emphasizing the significant impact of data analysis in cricket, particularly during player auctions and match outcomes. The research utilizes two primary datasets, IPL_Matches_2008_2022 and IPL_Ball_by_Ball_2008_2022, and conducts exploratory data analysis, batting, bowling, and fielding analysis. The datasets are merged to create a unified dataframe, focusing on second innings data for machine learning. Logistic regression is employed to predict match outcomes, achieving an 81% accuracy, and offering valuable winning probabilities during matches. The study contributes to the ongoing discourse on leveraging data science in sports analytics and offers a tool for estimating match outcomes.

Keywords:

Indian Premier League (IPL), Data Analysis, Machine Learning, Logistic Regression, Cricket Statistics, Data Cleaning, Data Preprocessing, Feature Engineering, Real-time Prediction, Match Outcomes, Sports Analytics

Table of Content

INTRODUCTION

The sport of cricket, which is so deeply enmeshed with Indian culture and whose popularity stretches well beyond geographic boundaries, has become immensely popular in India. The sport enjoys great appeal, Cricket reaches into every corner of India as it becomes entrenched in daily life here more deeply than anywhere else on earth. The Indian Premier League (IPL) has thereby become a cricketing extravaganza which brings together talent and strategy with an international audience in India. This project, with the title “Predictive Modeling and Team Performance Analysis in IPL” Exploring the team and player performances in IPL from 2008 to 2022 through data analytics and predictive modeling.

The Indian Premier League (IPL), also familiarly known as the "Carnival of Cricket" has teams that are based on franchises representing cities and regions playing in a Twenty20 format. Known for its all-action games, big-name players, team lineups, great approach to cricket broadcasting and groundbreaking presentation of cricket. With all this excitement, the ever-present issue of putting together winning teams is always a part of IPL player auctions. The choices team owners make have profound impacts on how well tournaments proceed and who wins them. As the IPL continues to grow in popularity and competitiveness, the quest for accurate predictions of match outcomes has become increasingly paramount. Fans, analysts, and stakeholders alike seek insights into the intricate dynamics that influence the results of these closely contested matches. This burgeoning interest has given rise to a field of research where data analytics and machine learning converge to unravel patterns, unveil trends, and ultimately predict the outcomes of IPL matches.

Based on two datasets, IPL_Matches_2008_2022 and IPL_Ball_by_Ball_2008_2022 which bring together over a decade's worth of information about the IPL matches, including every stroke hit or missed in each game. By looking at each step of every game in detail, from the way batsmen perform to how bowlers do and into how fielding responds, we combine all this information into one organized dataset. Our main focus is on the second innings an important part for our machine learning work. Beyond its immediate relevance to IPL franchises, this project aspires to contribute to the broader discourse on the integration of data science in sports analytics. Moreover, the project also is looking into ways that might allow further spread to local and grassroots competitions in other areas. Thus, expanding access rights among various regions of tournament cricket will be democratized too.

Through an exploration of the interplay between data and cricket, this project attempts to determine how strategic decision-making practices will evolve in an increasingly dynamic environment. The subsequent chapters of this thesis will delve into the intricacies of the methodology, present and interpret key findings, and discuss the implications of our research. Through this comprehensive exploration, the aim to shed light on the complexity of IPL match prediction, thereby enriching our understanding of the sport and paving the way for more informed decision-making in the dynamic realm of cricket.

BACKGROUND

Cricket, often characterized by intense excitement and an avid fan base, has increasingly become a field of dynamic play where sporting strategy intersects with data analysis. The advent of T20 formats such as leagues (like the IPL) has increased demand for detailed analysis on teams, players and matches. This is a new landscape for data science and cricket analysis. It offers excellent opportunities to understand patterns, formulate better strategies and predict results. This research endeavors to contribute to this domain by embarking on a comprehensive analysis of IPL matches and proposing a predictive model grounded in machine learning. The main point of our study is the IPL Complete Dataset, spanning from the inaugural season in 2008 to the year 2020, contains precious information about team compositions, player performances, and match outcomes. We refer to a number of inspiring and informative studies for reference in our attempt to establish the best predictive model. The research papers 'Cricket Team Prediction Using Machine Learning Techniques' and 'Sport analytics for cricket game results using machine learning' provide background information on player performance prediction, as well as describing the use of various machine learning algorithmns. (Petersen, 2008) analysis of Twenty/20 Cricket performance during the 2008 IPL season provided early glimpses into the multifaceted nature of cricketing statistics. Furthering the integration of technology, (Vistro, 2019) showcased the application of machine learning and data analytics in predicting cricket match winners.

The objective of this thesis "Predictive Modeling and Team Performance Analysis in IPL" is to be a logical extension from past work, continue creating new advances for the field of cricket. By performing thorough data cleaning, feature engineering and utilizing machine

learning techniques to predict match outcomes, aspire to construct a model that not only forecasts match outcomes but also contributes to the evolving field of sports analytics. The subsequent chapters will detail the methodology employed, present key findings, and delve into the implications of our research in enhancing our understanding of the intricate dynamics that govern the outcome of IPL matches. The evolution of cricket as a data-rich sport, coupled with the increasing accessibility of advanced analytical tools, opens avenues for a deeper understanding of the nuances that dictate success in cricket matches. The fusion of historical data, player statistics, and contextual variables forms the bedrock for constructing a predictive model that goes beyond mere speculation and harnesses the power of data-driven insights. In tandem with the aforementioned research, our study draws inspiration from diverse perspectives presented in research papers such as (Sudhamathy, 2020) exploration of IPL data using machine learning techniques in the R package. This study serves as a testament to the continual evolution of methodologies, as it exemplifies the adaptation of contemporary tools to the unique challenges posed by cricket match prediction. As we delve deeper into the multifaceted world of IPL analytics, the fusion of theoretical frameworks, practical insights, and advanced machine learning methodologies positions our project at the forefront of data-driven innovation within the realm of cricket analytics.

MOTIVATION

The roots of this project, "Predictive Modeling and Team Performance Analysis in IPL," link with a personal journey that spans the love for cricket, a deep connection with the Indian Premier League (IPL), and a professional path in the field of data analyst. As a big cricket fan since I was a kid, my connection with the cricket goes beyond just watching, since watching from a distance the first IPL season, my interest in the league has only increased as time goes by also one standout figure, Virat Kohli, has become a huge inspiration for life.

This project is motivated by the thought that there's a lot of potential between cricket and data analysis. Every IPL match is a different story so analysis takes more than merely the respecting of statistics. In addition, seeing the growing integration of data scientists and analysts in numerous cricket committees and teams made it clear that decisions based on data are not only a trend but an indispensable necessity. The desire that IPL franchises will have an excellent player analysis, so as to offer guidance when making bids for players in the auction. Inspired by hopes for the democratization of cricket statistics, this project aims to introduce data-driven analysis into local tournaments. Picture the day when players in small towns can have access to information on a comparable scale with that available at IPL franchises. This would truly be democratizing cricket analysis (and ownership of knowledge) beyond anything the glitzy IPL could provide.

As the IPL continues to captivate fans worldwide, the thirst for actionable insights into match outcomes intensifies. The unpredictability and multifaceted nature of cricket matches, coupled with the strategic intricacies employed by teams, beckon for a sophisticated analytical

approach to unravel patterns and trends that may influence the destiny of a match. Cricket, traditionally reliant on the intuitions and experiences of players and experts, has seen a paradigm shift with the advent of data-driven methodologies. The potential implications extend beyond the boundary, impacting the cricketing ecosystem by offering insights into player performances, team dynamics, and the nuanced factors that tip the scales in favor of one team over another.

This research is motivated by the aspiration to bridge the gap between traditional cricketing knowledge and modern analytical methodologies. By leveraging the wealth of data encapsulated in the IPL Complete Dataset, the aim to contribute not only to the predictive modeling of match outcomes but also to the broader discourse on the evolving nature of cricket as a data-rich sport. Through the lens of machine learning and advanced analytics, this study seeks to unravel the underlying patterns in IPL match results, offering a data-driven perspective that augments the traditional cricketing narrative. As embark on this journey, the motivation lies in the prospect of uncovering insights that transcend the boundaries of conventional wisdom and pave the way for a more informed and nuanced understanding of the beautiful game within the electrifying context of the IPL. Basically, this is born out of the love for cricket and field in data analyst and science together, hoping to change people's viewpoint on thinking about cricketing.

LITERATURE REVIEW

Cricket, with its unpredictable nature and ever-evolving dynamics, has become a fascinating subject for researchers exploring the applications of data mining, machine learning, and analytics. The Indian Premier League (IPL), as a pinnacle of modern cricket, has been a focal point for numerous studies aiming to unravel the intricate patterns and factors influencing match outcomes.

Authors (Nimmagadda, n.d.) set the stage for predictive analytics in cricket by employing data mining techniques to predict cricket scores and outcomes. Their study highlighted the significance of statistical analyses in understanding the complex web of variables that contribute to match results. This early exploration paved the way for more nuanced and sophisticated approaches to cricket prediction. (Petersen, 2008) extended the discourse by delving into the performance dynamics of Twenty/20 Cricket during the 2008 IPL season. In a format known for its fast-paced nature and emphasis on aggressive play, their analysis provided early insights into trends and factors influencing team and player performances. This period marked a crucial juncture in the evolution of T20 cricket, and the study offered valuable perspectives on the strategic nuances that emerged. The integration of machine learning and data analytics into cricket prediction gained prominence through the work of (Vistro, 2019). Their study exemplified the potential of advanced techniques in capturing intricate patterns and contributing substantively to the field of cricket analytics. By leveraging machine learning algorithms, the research aimed not only to predict match winners but also to uncover underlying trends that traditional analyses might overlook.

(Sudhamathy, 2020) furthered the exploration by applying machine learning techniques within the R package, showcasing the adaptability of contemporary tools in cricket analytics. The study emphasized the need for dynamic methodologies that can evolve with the changing landscape of cricket. This adaptability is crucial in a sport where player strategies, team dynamics, and external variables can significantly influence match outcomes. Predicting the performance of bowlers in the IPL took center stage in the research by (Saikia, 2012). Their approach, utilizing artificial neural networks, demonstrated the efficacy of advanced computational methodologies in capturing the nuances of player-specific contributions to match outcomes. As cricket strategies evolved, the study illustrated the importance of incorporating individual player performances into predictive models. The multifaceted nature of factors influencing match outcomes was explored by (Jaipuria, 2022), who delved into the influence of toss results, toss decisions, and venue conditions on IPL matches. This comprehensive analysis highlighted the intricate interplay between external variables and team performances, providing valuable insights into the diverse factors shaping match results. In a sport where the toss is considered a crucial determinant, this research added depth to our understanding of the game's dynamics. The visualization and prediction of IPL matches using HBase was explored by (Singh, 2017). Their study ventured into the integration of technologies beyond traditional analytics tools, showcasing the potential of diverse methodologies in cricket analytics. By leveraging HBase, a distributed database, the research demonstrated the importance of technological innovation in handling vast datasets inherent in cricket analytics.

Machine learning algorithms, specifically Support Vector Machines (SVM), took the spotlight in the study by (Sinha, 2020), who proposed an IPL Win Prediction System. This research

showcased the versatility of SVM in handling the complexities of cricket data, offering a predictive model to enhance team performance. The adaptability of machine learning algorithms to the unique challenges posed by cricket analytics was evident, providing a promising avenue for predictive modeling. An in-depth analysis of toss results, toss decisions, and venue conditions in IPL matches was furthered by (Singhal, 2023). Their study contributed to the ongoing discourse on the influence of external variables on match outcomes. The nuanced exploration of these factors added depth to our understanding of the intricate dynamics that shape the destiny of IPL matches. In this rich tapestry of research, our study aims to contribute substantively by leveraging the IPL Complete Dataset and applying advanced machine learning algorithms. The collective insights from these studies lay a foundation for a comprehensive analysis of IPL match outcomes. Through this exploration, aspire not only to predict match winners accurately but also to offer a nuanced comprehension of the diverse factors influencing the outcomes of IPL matches.

The narrative of cricket analytics and prediction is a journey marked by innovation, adaptability, and a continual quest for understanding in a sport that thrives on uncertainty. As we delve deeper into this rich landscape, it becomes evident that each research endeavour contributes a unique perspective, shaping the collective narrative of cricket analytics. The work of (Nimmagadda, n.d.), which initially explored the predictive potential of data mining, opened doors to a realm where the statistical intricacies of cricket could be unravelled. Their emphasis on data-driven insights laid the groundwork for subsequent studies, forging a path toward a more systematic and informed approach to cricket prediction. (Petersen, 2008) stepped into this evolving arena during the nascent stages of the IPL, capturing the essence of a format that was fast becoming the epitome of excitement in cricket. The 2008 season, a watershed moment for

T20 cricket, provided fertile ground for analysis, offering insights into the performance dynamics that set the stage for the league's future. (Vistro, 2019) ushered in a new era by integrating machine learning and data analytics into cricket prediction. The study exemplified the potential of advanced techniques in capturing nuanced patterns that elude traditional analyses. In doing so, it not only aimed to predict match winners but also to decode the underlying trends that define the ebb and flow of IPL matches. (Sudhamathy, 2020) demonstrated the flexibility of methodologies by embracing machine learning techniques within the R package. Their work underscored the need for adaptable tools in cricket analytics, where the evolving nature of the sport demands approaches that can seamlessly navigate through diverse datasets and emerging trends. The study by (Saikia, 2012) delved into the realm of player-specific performance prediction, acknowledging the crucial role individual contributions play in shaping match outcomes. By leveraging artificial neural networks, the research signalled a shift towards a more granular understanding of player dynamics, a dimension that traditional analyses often struggle to encapsulate. (Jaipuria, 2022) explored the influence of external variables on match outcomes, adding depth to the narrative by examining toss results, toss decisions, and venue conditions. In a sport where conditions can be as influential as player performances, this study provided a comprehensive view of the contextual factors shaping the destiny of IPL matches.

(Singh, 2017) ventured into the realm of visualization and prediction using HBase, a testament to the integration of technology into cricket analytics. Their work showcased the potential of innovative technologies in handling and interpreting the vast datasets inherent in cricket analytics, demonstrating the need for a multidimensional approach. The proposal of an IPL Win Prediction System by (Sinha, 2020), grounded in Support Vector Machines, highlighted

the adaptability of machine learning algorithms to the challenges of cricket analytics. The study showcased the potential of SVM in providing a predictive model that could enhance team performance, signalling a shift toward more sophisticated analytical tools in the realm of cricket prediction. The extensive analysis of toss results, toss decisions, and venue conditions in IPL matches by (Singhal, 2023) further enriched our understanding of the intricate dynamics that govern match outcomes. By exploring the multifaceted nature of these external factors, the study underscored the complexity inherent in predicting the results of high-stakes cricket matches.

(Thorat, 2021) highlighting the prevalence of research predominantly centered on the One Day International (ODI) format. However, the authors emphasize the dynamic nature of the Twenty20 (T20) format, advocating for increased attention in predicting match outcomes within the cricket domain. This underscores the need for a tailored approach to accommodate the unique parameters of T20 cricket. The research Cricket Team Prediction Using Machine Learning Techniques by (Patil, n.d.) underscores the crucial role of such estimates in aiding team management decisions, facilitating the selection of optimal players for each match. (Kapadia, 2019) proposed machine learning algorithms and feature selection methods in cricket analytics. Utilizing filter-based techniques like Correlation-based Feature Selection and Information Gain, the research employs machine learning models such as Naïve Bayes, Random Forest, K-Nearest Neighbour (KNN), and Model Trees. These findings showcase a comprehensive evaluation of methodologies to predict cricket match results.

These research endeavors collectively lay the foundation for this project, aiming to extend the discourse on predictive modeling and team performance analysis in IPL. Our research stands at the intersection of these diverse perspectives, drawing inspiration from the collective wisdom

of these studies. The IPL Complete Dataset serves as a treasure trove of information, a repository from which we extract insights to construct a predictive model that not only forecasts match winners but also contributes substantively to the broader field of cricket analytics.

METHODOLOGY

The methodology diagram in 'Figure 1' shows the important steps undertaken in this project. Starting with the collection of two datasets from Kaggle, the next step unfolds with methodical data analysis performed on each dataset. Subsequently, the datasets are properly merged, followed by a comprehensive phase of data cleaning and preprocessing to ensure data integrity. The feature engineering process is instrumental in creating new columns, enhancing the dataset's depth. The implementation of a logistic regression machine learning model stands as a vital step, providing a predictive framework for match outcomes. Finally, the methodology concludes with a real-time prediction phase, where users input specific parameters, and the model delivers instantaneous predictions based on the established logistic regression model. This systematic approach ensures a robust and insightful exploration of IPL cricket dynamics.

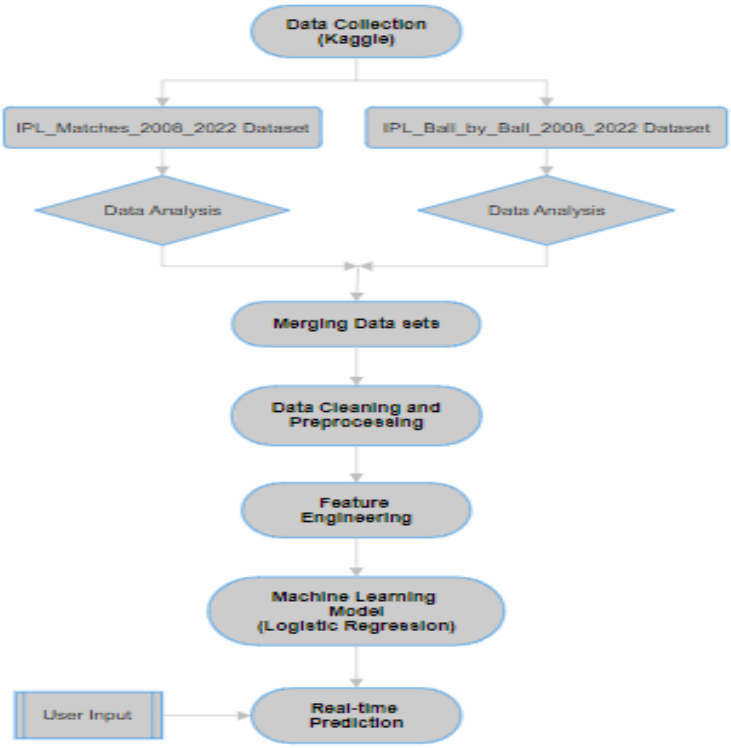

Figure 1: Architecture of the research

MAIN BODY

Dataset Overview:

The IPL datasets, collected from Kaggle, summarize a comprehensive journey of the Indian Premier League spanning from 2008 to 2022.

IPL_Matches_2008_2022 Dataset: The first dataset, contains information about each individual IPL match. Each row in this dataset corresponds to a single match, provides details of the matches played. Column names and detailed explanation of each column as follows:

ID: Unique identifier for each IPL match.

City: The city where the match was played.

Date: Date of the match.

Season: The IPL season (2008 to 2022)

MatchNumber: The specific number assigned to each match within a season.

Team1: One of the participating teams.

Team2: The other participating team.

Venue: Stadium where the match took place.

TossWinner: Team that won the toss.

TossDecision: Decision taken by the team after winning the toss (bat or bowl).

SuperOver: When a match ends in a draw, it is decided by playing one extra over, which is called a super over. Indicates if a Super Over was played.

WinningTeam: Team that won the match.

WonBy: Margin of victory which match won by runs or by wickets.

Margin: Additional details about the margin of victory, number of runs or count of wickets.

method: if match affected by rain or some other reason, then DLS method apply (Duckworth-Lewis).

Player_of_Match: Player who was awarded the "Man of the Match."

Team1Players: List of players in Team1.

Team2Players: List of players in Team2.

Umpire1: Name of the first umpire.

Umpire2: Name of the second umpire.

IPL_Ball_by_Ball_2008_2022 Dataset: The second dataset, provies detailed information about every ball bowled in the IPL. Each row in this dataset corresponds to an individual ball played throughout the tournament. Column names and detailed explanation of each column as follows:

ID: Unique identifier for each IPL match.

innings: Innings number of match (first or second).

overs: Number of overs bowled in the current innings.

ballnumber: Specific ball number within an over.

batter: Name of batsman facing the ball.

bowler: Name of bowler delivering the ball.

non-striker: Name of batsman at the non-striker's end.

extra_type: Type of extras (if any) conceded in the delivery.

batsman_run: Runs scored by the batsman off the delivery.

extras_run: Extra runs conceded (if any) off the delivery.

total_run: Total runs scored in the delivery (sum of batsman_run and extras_run).

non_boundary: Indicates if the delivery was a boundary.

isWicketDelivery: Indicates if the delivery resulted in a wicket.

player_out: Name of batsman who got out (if applicable).

kind: Type of dismissal (e.g., caught, bowled, runout).

fielders_involved: Name of fielder involved in the dismissal.

BattingTeam: Team batting in the current innings.

After performing proper analysis on both datasets, a merged dataset was constructed by integrating them on the basis of the ID column. That integration became the basis for complete studies of the batting, bowling and fielding ends as well as team analysis.

Implementation:

In the initial phase of the project, the foundation laid by importing essential libraries, including numpy for numerical operations, pandas for data manipulation, and matplotlib and seaborn for data visualization.

```
import numpy as np # to work with numeric data.
import pandas as pd # data import, data manipulation, statistics.
import matplotlib.pyplot as plt # data visualization.
import seaborn as sns # data visualization.
```

Datasets Analysis:

Analysis on dataset1:

Loading IPL_Matches_2008_2022 Dataset in df1, which contains 950 rows and 20 columns, summarizes essential details of IPL matches from 2008 to 2022. Getting basic understanding about dataset before move forward on data arrangement/cleaning and analysis.

```
# load IPL_Matches_2008_2022 dataset in df1.

df1 = pd.read_csv(r"C:\Users\rohit\OneDrive\Desktop\DOCUMENTATION_AND_IPL_DATASET\IPL_Matches_2008_2022.csv")
```

```
# Total number of rows and columns in the df1.

df1.shape
```

```
(950, 20)
```

Identified null values in 'City', 'Margin', and 'Method' columns. Null values presented in the 'City' column were addressed by replacing them with the respective cities where the matches took place, such as Dubai and Sharjah. The replaced name of cities based on a contextual

understanding of the dataset and a knowledge of IPL venues. Also, changed the datatype of date column from object to datetime.

```
# Replacing nan values in 'City' column with respective cities. (Dubai, Sharjah)

df1.loc[(df1['City'].isnull()) & (df1['Venue'] == 'Dubai International Cricket Stadium'), 'City'] = 'Dubai'
df1.loc[(df1['City'].isnull()) & (df1['Venue'] == 'Sharjah Cricket Stadium'), 'City'] = 'Sharjah'
```

```
# Check nan values.

df1['City'].isnull().sum()
```

```
0
```

The first type of analysis is univariate. Made some insightful discoveries after conducting a detailed univariate analysis on IPL_Matches_2008_2022 Dataset.

Explored how many matches took place in each city, giving insights into the distribution of IPL games in different locations. Explored the count of matches conducted in each stadium, giving an overview of how particular venues are favored and utilized in the context of the games. Delved into the number of matches won by each team represents 'Figure 2', providing a comprehensive overview of team performances.

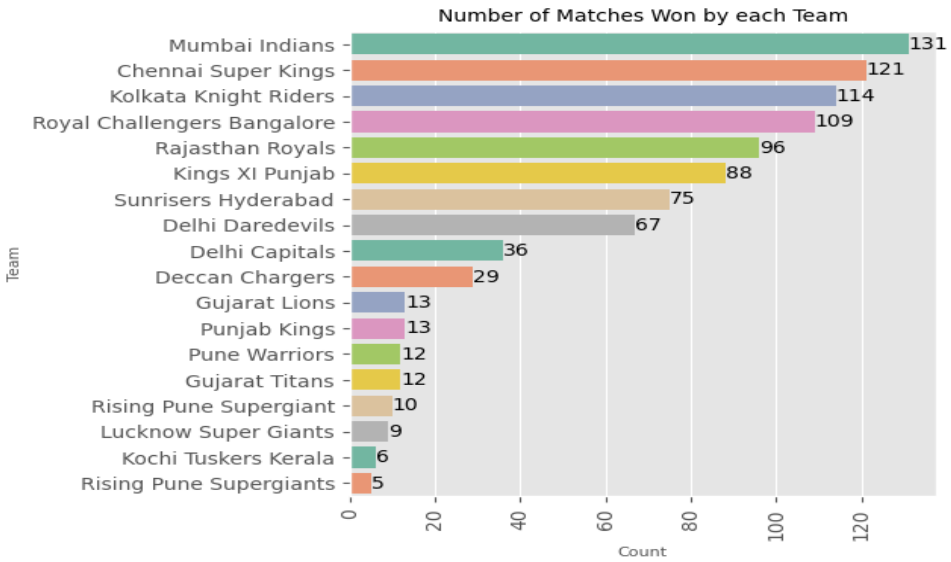

Figure 2: Match Won by Each Team

Identified the top ten players with the most Player of the Match awards which represented in 'Figure 3', recognizing individual player contributions. Analyzed toss decisions through a pie chart shown in 'Figure 4', showcasing the distribution of teams choosing to bat or field after winning the toss. Investigated the proportion of matches predicted according to DLS method which are only 0.02%, with a focus on how weather affects match results. Finding top five teams with most number of toss wins. Examined match results, revealing the percentage of matches decided with and without Super Overs which are only 1.47%, providing insights into the competitiveness of IPL matches.

	Player	Player_of_Match_count
0	AB de Villiers	25
1	CH Gayle	22
2	DA Warner	18
3	RG Sharma	18
4	MS Dhoni	17
5	YK Pathan	16
6	SR Watson	16
7	KA Pollard	14
8	SK Raina	14
9	V Kohli	14

Figure 3: Player of the Match

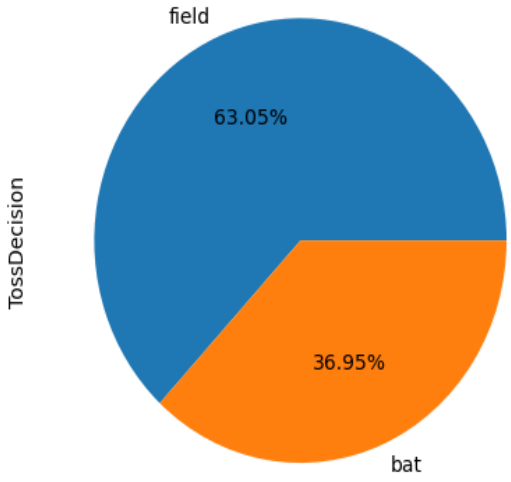

Figure 4: Toss Decision Percentage

Extended the analysis to bivariate and multivariate dimensions for a deeper understanding. Explored the number of matches played by each team using 'Team1' and 'Team2' columns, presenting a comparative view of team participation across seasons. Investigated match outcomes by margin shown in 'Figure 5', distinguishing wins by wickets and runs, visualization helps understanding this better and also, identified the top five wins by run margin (Where team

Mumbai wins over team Delhi by 146 runs which is the highest). Unearthed patterns related to umpire appearances, identifying the umpire with the most on-field appearances in matches.

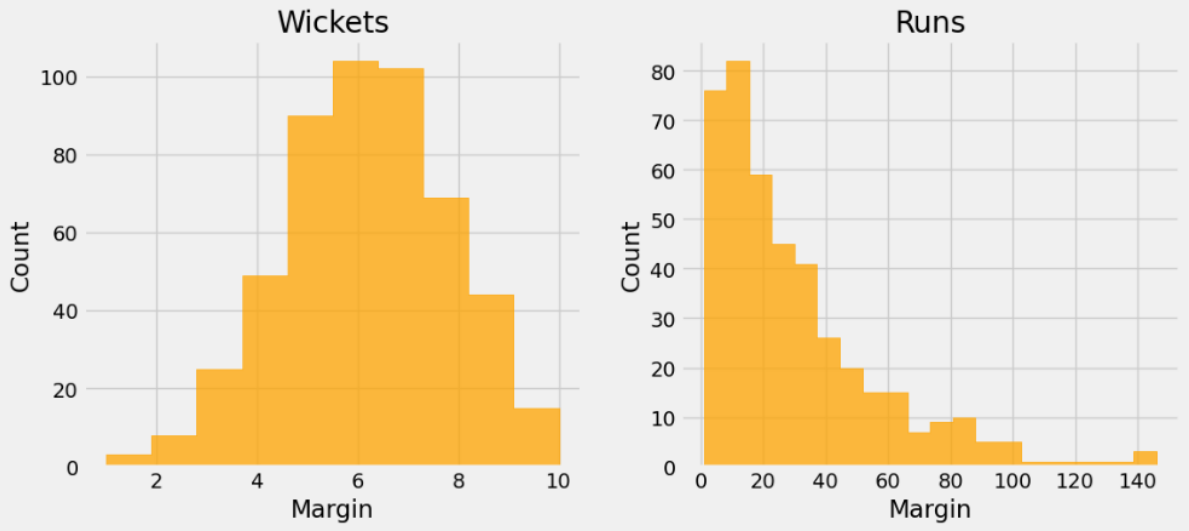

Figure 5: Win Margin Distribution

Analysis on dataset2:

The dataset, df2, comprises 225954 rows and 17 columns, capturing detailed information about each ball bowled in the IPL.

```
# load IPL_Ball_by_Ball_2008_2022 dataset in df2.
df2 = pd.read_csv(r"C:\Users\rohit\OneDrive\Desktop\DOCUMENTATION_AND_IPL_DATASET\IPL_Ball_by_Ball_2008_2022.csv")
```

```
# Total number of rows and columns in the df2.
df2.shape
```

```
(225954, 17)
```

Check data quality results in no duplicates present in the dataset and null values present in 'extra_type', 'player_out', 'kind', and 'fielders_involved' columns. Null values in 'extra_type' column indicate legal deliveries where no extra runs given by bowler. Columns 'player_out', 'kind',

'fielders_involved' contains null values where a player did not get out during the match. It is common for many deliveries.

```
# 'extra_type' column contains null values where balls bowled are legal deliveries.

df2['extra_type'].isnull().sum()
```

```
213905
```

```
# Null values in the 'player_out','kind', 'fielders_involved', where a player did not get out during the match.

print('player_out:', df2['extra_type'].isnull().sum())
print('kind:', df2['extra_type'].isnull().sum())
print('fielders_involved:', df2['extra_type'].isnull().sum())
```

```
player_out: 213905
kind: 213905
fielders_involved: 213905
```

Analyzed the distribution of runs in each over for first and second innings shown in 'Figure 6', providing insights into the scoring patterns over different phases of game. Utilized a pie chart to showcase the distribution of extra types. Distribution of runs scored on each ball.

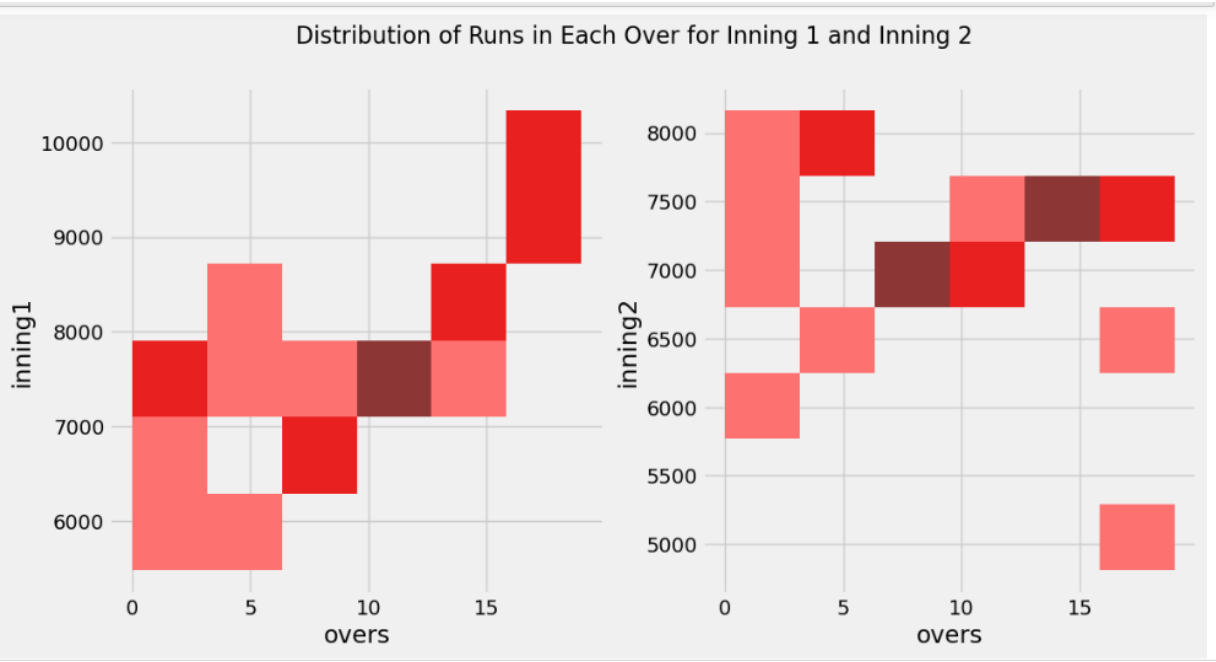

Figure 6: Runs Scored in Overs in IPL

Investigated the number of boundaries hit on each ball of every over which is shown in the ‘Figure 7’, provide insights into the attacking phases of the innings. Analyzed the frequency of player dismissals, focusing on the 'kind' column.

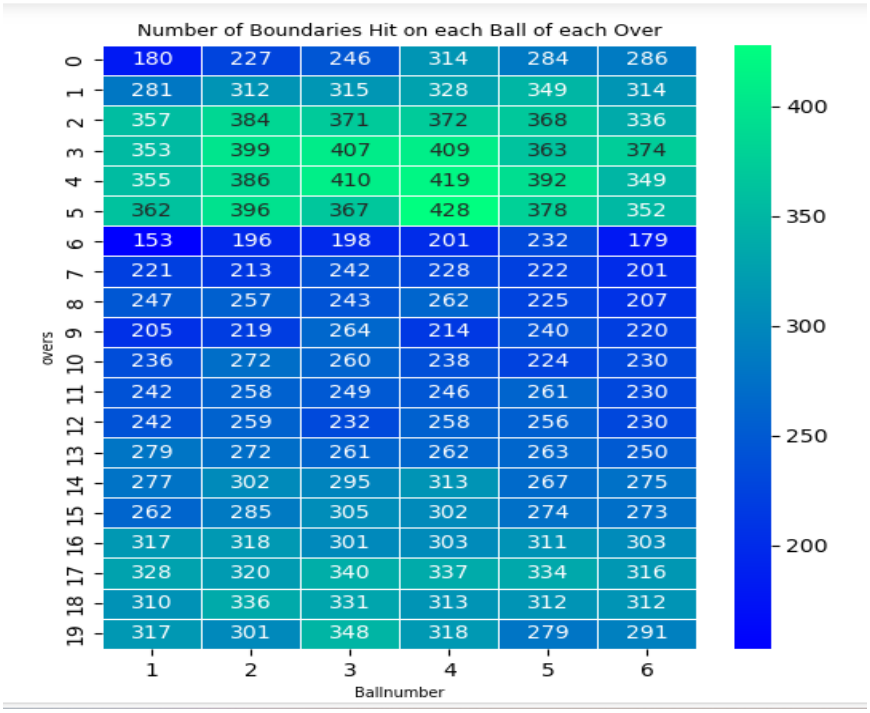

Figure 7: Boundary Hitting Over/Ball

Insights and Implications:

This analysis gives us an even better understanding of the distribution and effects in IPL matches, especially with regard to tosses, extra runs, scoring runs, match locations and differences between teams. Insights into how people decide to toss and which team actually wins the match help us develop a deeper understanding of overall strategy and gameplay. Recognizing the top-performing teams and players adds a layer of context to the broader narrative of IPL seasons. Analyzing the appearances of umpires also reveals something about how people have officiated matches and displays some of what officials in years past actually did. The runs across

overs are a strategic assignment that tells teams when it is best to be aggressive, or defense. Dismissal of players rates notes on how the game is played.

Merging Dataset:

Next, merging of datasets based on the 'ID' column, storing it into df, which contains 225,954 rows and 36 columns. The combined database provides detailed statistics for each ball bowling, as well information on matches, teams, players and other relevant attributes. Dataset df used for further batting, bowling, fielding and team analysis, as well as in feature selection, feature engineering, machine learning.

```
# Merge datasets and store in df.

df = df2.merge(df1, left_on='ID', right_on='ID')
```

Batting Analysis:

The performances of the top five batsmen in the Indian Premier League (IPL) across seasons closely Analyzed which is shown in the 'Figure 8'. Their run-scoring patterns examined to uncover the consistency and adaptability of these batsmen in the dynamic T20 format. An important finding are big hitters in the IPL, those with a strike rate of 150 or more and faced at least 200 balls. This helped to see who could really change the game with their aggressive style of batting. This analysis pinpointed the players who possessed the ability to single-handedly alter the course of a match, which is valuable in T20 format.

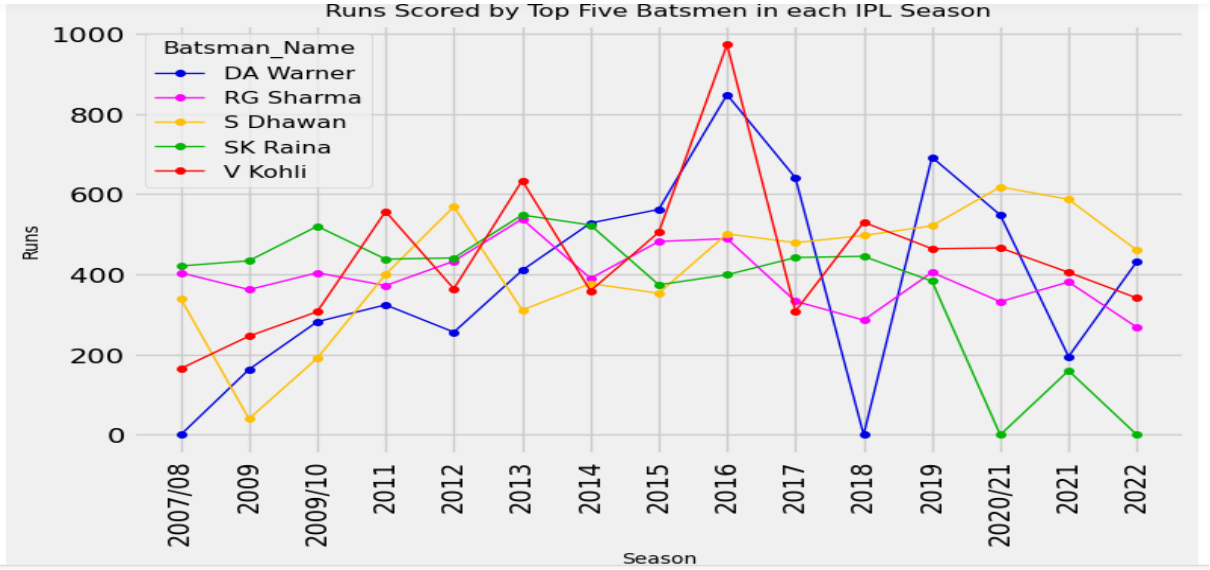

Figure 8: Top Batsmen Runs Analysis

Figured out the top ten players who spent the most time at the non-strike end. This unique viewpoint gave glimpse into player role in partnerships and team strategies. The analysis identified the top ten players who faced the most dot balls. Where batsmen face challenges while rotating strike. Finding top ten players with the most sixes and fours in the IPL celebrated the art

of boundary hitting. These charts which you can see in the ‘Figure 9’ and ‘Figure 10’ captured the aggressive intent of these batsmen clearing the rope’s ability of batsmen.

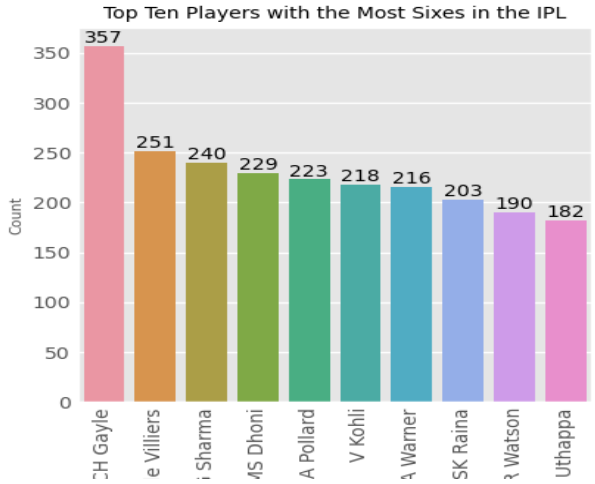

Figure 9: Player with Most Six

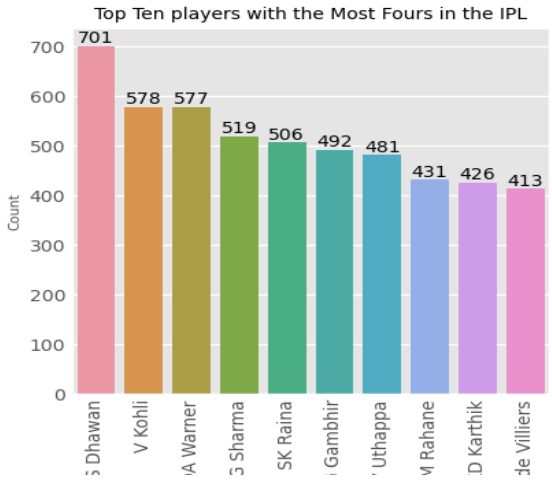

Figure 10: Player with Most Four

Average and strike rate are important aspect for batsmen, finding these for every batsman. Exploring the balance between consistency and aggression, visualizations of the average versus strike rate which you can observe in the ‘Figure 11’. This comparative analysis provided a nuanced understanding scoring runs consistently and doing so at a brisk pace.

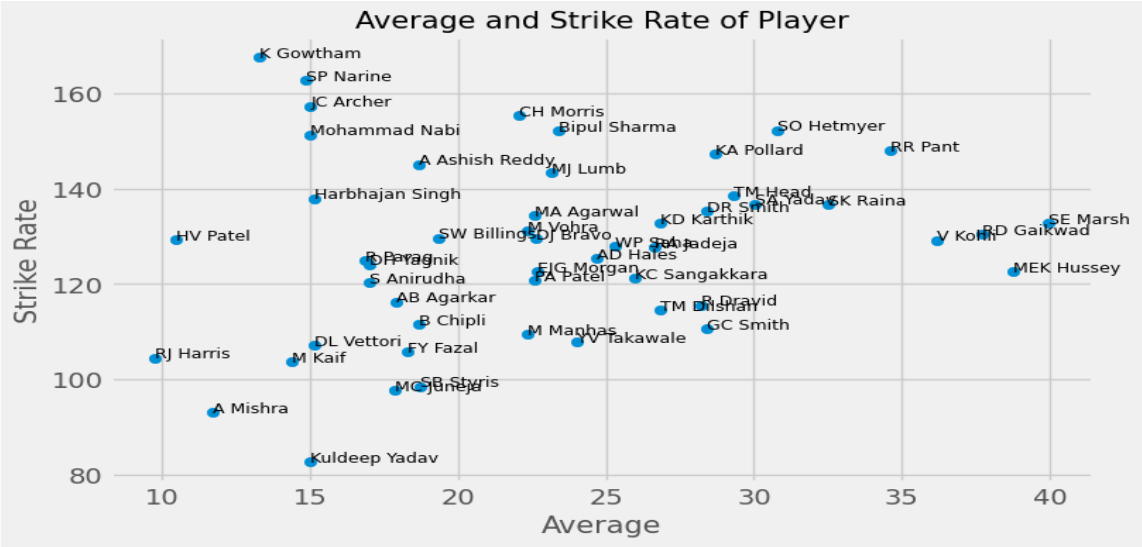

Figure 11: Average and Strike Rate of Batsmen

Figured Orange Cap Winners, that displays player performances averaged over IPL seasons represented in the ‘Figure 12’. The coveted orange cap goes to the batter who points most runs in a season and this statistic is an excellent foundation for providing network analysis on players performances at any game, past or present. Analyze the top five pairs of batsmen who scored the most runs together in the IPL. This showed us how well key players worked together, contributing to the team's overall runs.

A special function was crafted for in-depth exploration of any batsman's career. By simply inputting the player's name, users could access a wealth of information, including matches played, runs scored, balls faced, average, strike rate, fours, sixes, highest score, and milestones like half-centuries and centuries. This function provided a one-stop shop for detailed statistics on a batsman's IPL journey, one of the examples you can see in the ‘Figure 13’.

	SEASON	PLAYER_NAME	RUNS
0	2007/08	SE Marsh	616
1	2009	ML Hayden	572
2	2009/10	SR Tendulkar	618
3	2011	CH Gayle	608
4	2012	CH Gayle	733
5	2013	MEK Hussey	733
6	2014	RV Uthappa	660
7	2015	DA Warner	562
8	2016	V Kohli	973
9	2017	DA Warner	641
10	2018	KS Williamson	735
11	2019	DA Warner	692
12	2020/21	KL Rahul	670
13	2021	RD Gaikwad	635
14	2022	JC Buttler	863

Figure 12: Orange Cap

	INDEX	DATA
0	PLAYER_NAME	AB de Villiers
1	MATCH_PLAYED	183
2	RUNS	5162
3	BALLS_FACED	3403
4	AVERAGE	39.71
5	STRIKE_RATE	151.69
6	FOURS	413
7	SIXES	251
8	HIGH_SCORE	133
9	50	40
10	100	3

Figure 13: Batsmen Statistics

Bowling analysis:

The bowling analysis section gives in depth analysis of bowlers' performances in the Indian Premier League (IPL) which is shown in the 'Figure 14'. It first classifies wickets, revealing the many ways (which are caught, bowled, lbw, stumped, caught and bowled, hit wicket) bowlers have taken out batsmen. Identifying the top five bowlers who, collectively have taken by far the most wickets over all of IPL history. This finding helps cricket enthusiasts and analysts a quick overview of the consistent and impactful bowlers over the league's history. To provide a more detailed perspective, find the wickets of top bowlers in the individual IPL seasons. Moving forward, analysis on the bowlers who given the most extra runs in the IPL, which gives idea of areas for improvement and strategic consideration.

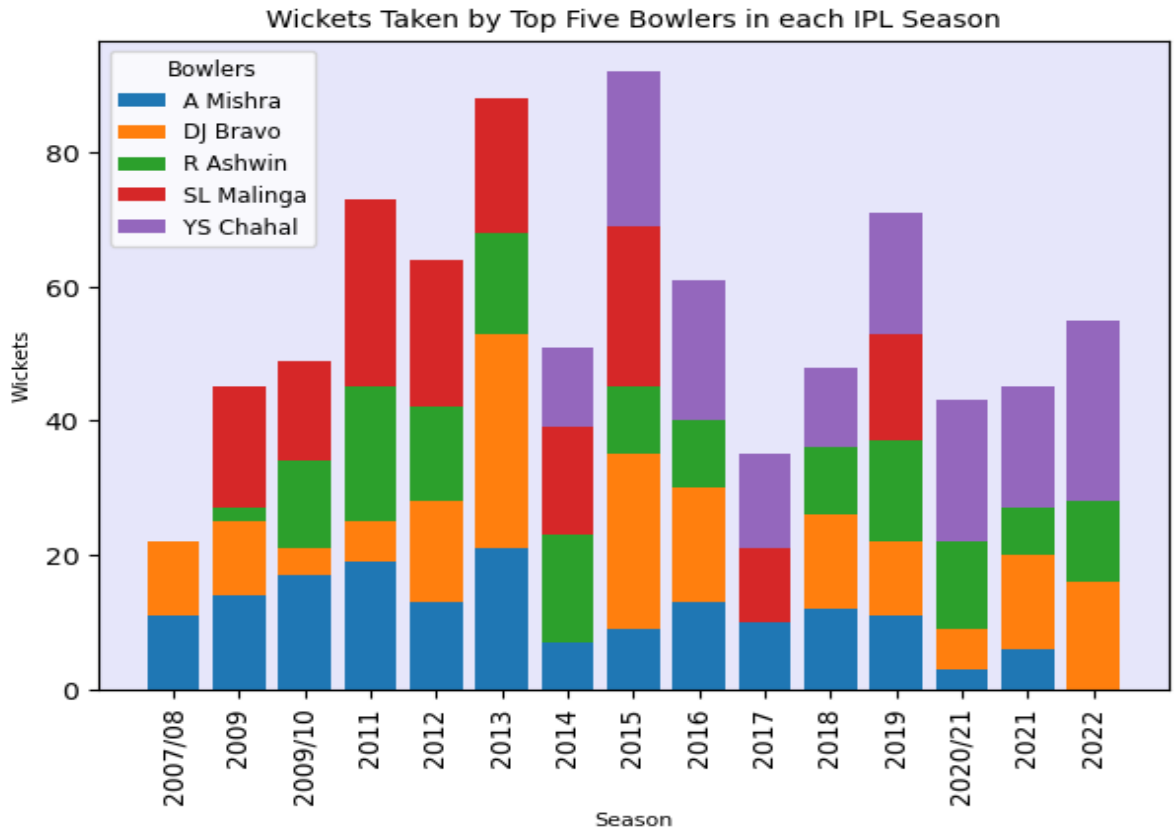

Figure 14: Top Bowlers Wickets Analysis

Next, finding number of super overs bowled by bowlers', provides information about who can bowl in high pressure situations and the ability of certain bowlers in such scenarios. Taking five wickets it's not easy task in T20 format. Finding bowlers with five-wicket hauls also recognized as the most outstanding efforts in the IPL. This shows individual performance and contributions to team to win the match. A good balance between a low bowling average and a low strike rate is important in bowler's carrier. Analysis of average and strike of each bowler in the IPL which you can see in the 'Figure 15'. This finding shows clear indication of one's regularity, ability of taking wickets and overall impact on the game.

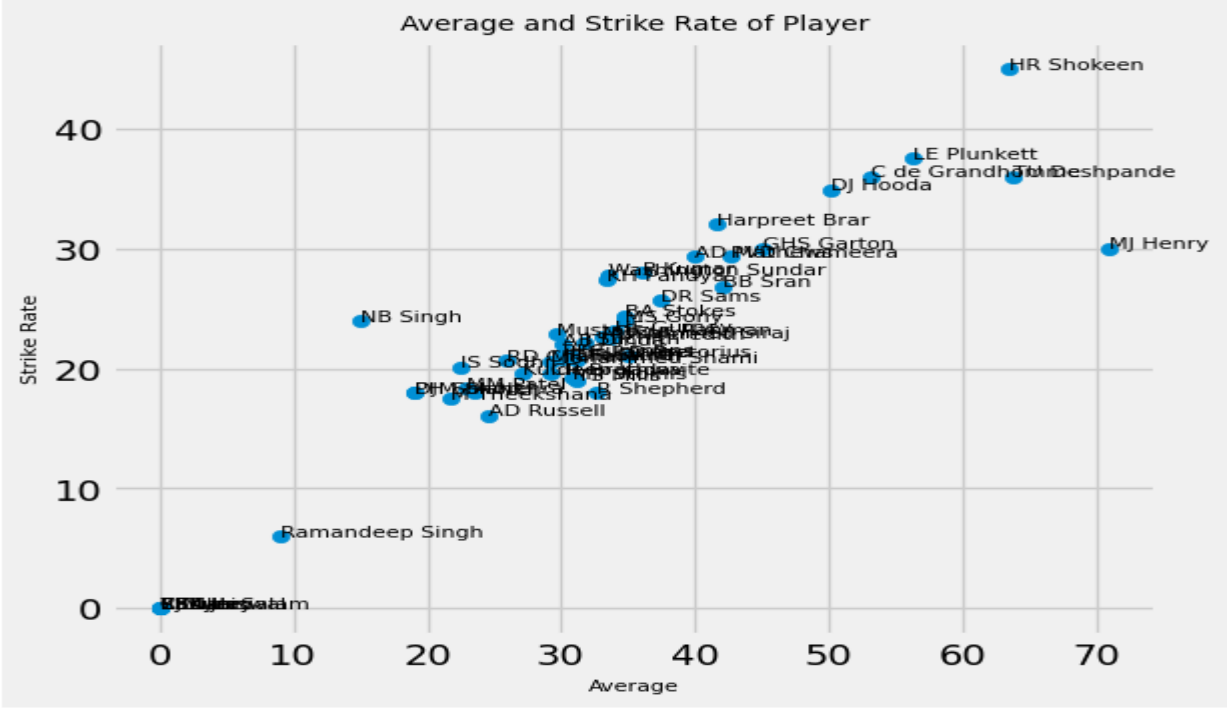

Figure 15: Average and Strike Rate of Bowler

The bowler who takes most number of wickets in the each IPL season, awarded with purple cap. Finding purple the purple cap winner us important which is shown in 'Figure 16' because, as

player not only earns recognition from the franchise but may also lead to a potential call-up from the national team.

Lastly, the function is created where by inputting the bowler's name, users gain access to valuable information about bowler statistical carrier, including matches played, wickets taken, balls bowled, economy rate, average, strike rate, five-wicket hauls, four-wicket hauls, and the bowler's best performance. Using this function one can get stats of any bowler an example you can see in the 'Figure 17'.

	SEASON	PLAYER_NAME	WICEKTS
0	2007/08	Sohail Tanvir	22
1	2009	RP Singh	23
2	2009/10	PP Ojha	21
3	2011	SL Malinga	28
4	2012	M Morkel	25
5	2013	DJ Bravo	32
6	2014	MM Sharma	23
7	2015	DJ Bravo	26
8	2016	B Kumar	23
9	2017	B Kumar	26
10	2018	AJ Tye	24
11	2019	Imran Tahir	26
12	2020/21	K Rabada	30
13	2021	HV Patel	32
14	2022	YS Chahal	27

Figure 16: Purple Cap

	INDEX	DATA
0	PLAYER_NAME	Harbhajan Singh
1	MATCH_PLAYED	163
2	WICKETS	150
3	BALLS_BOWLED	3416
4	OVERS	569
5	ECONOMY	7.08
6	AVERAGE	26.87
7	STRIKE_RATE	22.77
8	5Wh	1
9	4Wh	1
10	HIGH_SCORE	5/18

Figure 17: Bowler Statistics

Fielding analysis:

Fielding is important aspect of while playing cricket, Fielding analysis in the IPL reveals highly important metrics on field such as total catches, stumpings and runouts. Here, finding total number of catches taken, stumpings and run outs in the IPL, there are 7150 catches taken, 1000 runouts and 325 stumpings in the IPL till 2022.

Also top ten players with the most catches and runouts provides a clear perspective on the standout fielders in the IPL. Top players with most stumpings shows the wicketkeeping ability and fitness of players.

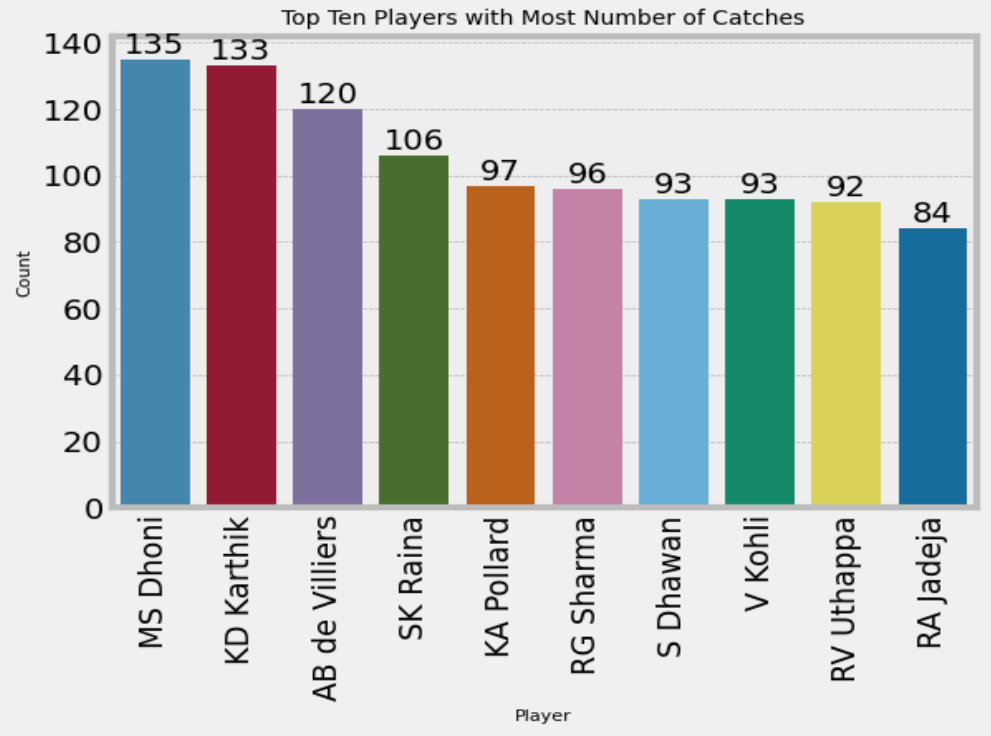

Figure 18: Most Number of Catches

Team analysis:

In team-based analysis, examining how each team performed against others teams. The heatmaps in the ‘Figure 19’ and ‘Figure 20’ shows runs scored and wickets taken by teams while playing against all other teams unveiled strategic patterns and strengths, providing valuable insights into the dynamics of team matchups and their tactical approaches.

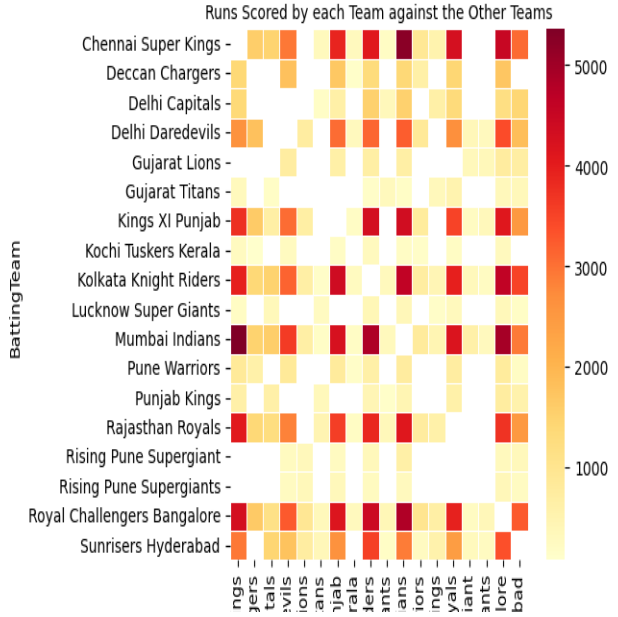

Figure 19: Runs Against Each Other

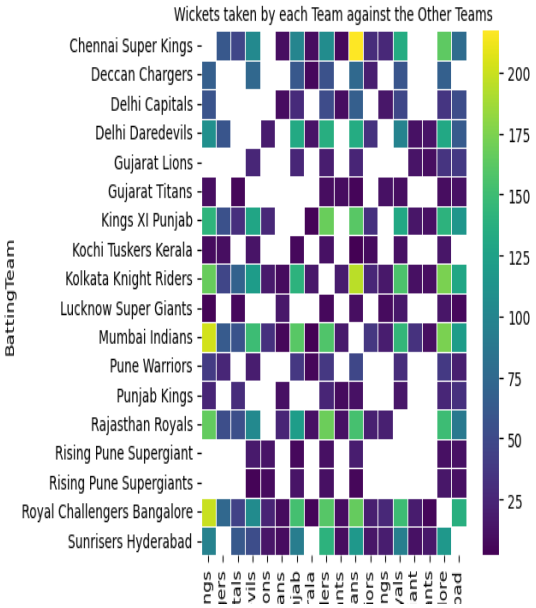

Figure 20: Wickets Against Each Other

Finding the ratio of toss wins to match wins gives insights into teams' decision-making process. Identifying teams that have qualified (ending up as top four in points table) the most times provides insight into their overall performance and consistency in the IPL.

Lastly, the visualization of IPL trophies won by each team is representation of historical achievements of franchises. This overview highlights the dominance and success of certain teams at lifting the IPL trophy. Collectively these analyzes provide a complete perspective on team play

in the IPL, including batting and bowling performances as well as strategic decisions and historical achievements.

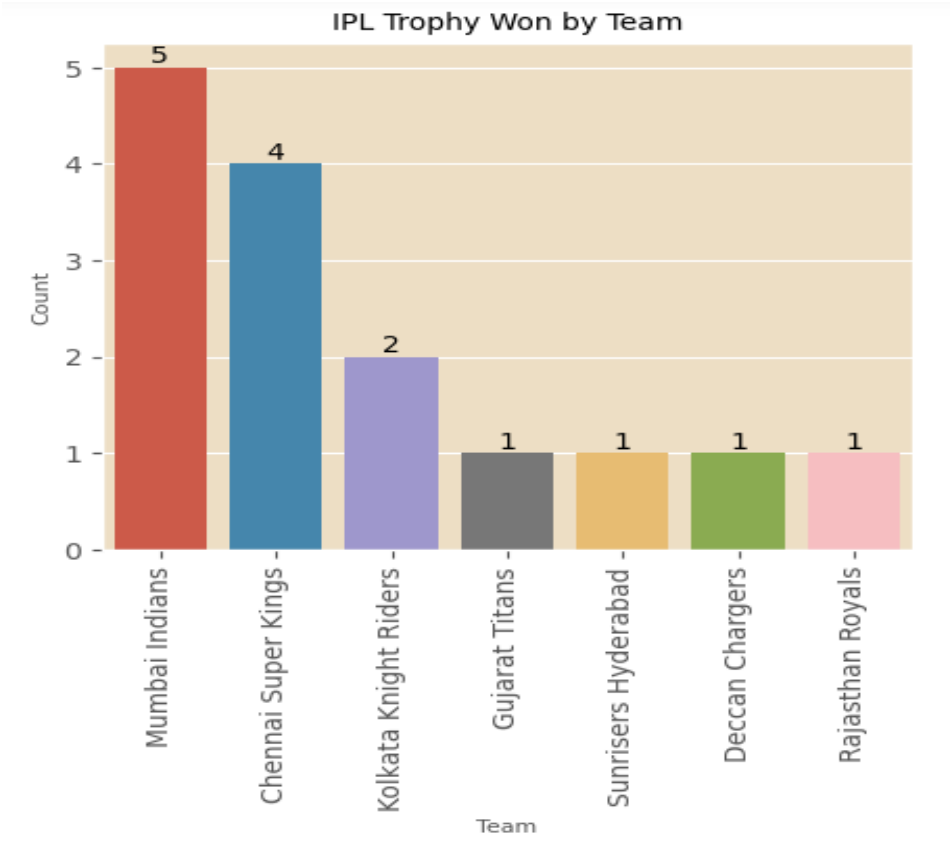

Figure 21: IPL Trophies Won by Teams

Feature Engineering:

This code segment addresses missing values in the 'City' column of df1 based on information available in the 'Venue' column. If 'City' is null and 'Venue' is 'Dubai International Cricket Stadium', the 'City' is filled with 'Dubai'. Similarly, if 'City' is null and 'Venue' is 'Sharjah Cricket Stadium', the 'City' is filled with 'Sharjah'.

```
# To keeping it clean, importing the data again.
# IPL_Matches_2008_2022 dataset in df1.
# IPL_Ball_by_Ball_2008_2022 dataset in df2

df1 = pd.read_csv(r"C:\Users\rohit\OneDrive\Desktop\DOCUMENTATION_AND_IPL_DATASET\IPL_Matches_2008_2022.csv")
df2 = pd.read_csv(r"C:\Users\rohit\OneDrive\Desktop\DOCUMENTATION_AND_IPL_DATASET\IPL_Ball_by_Ball_2008_2022.csv")

df1.loc[(df1['City'].isnull()) & (df1['Venue'] == 'Dubai International Cricket Stadium'), 'City'] = 'Dubai'
df1.loc[(df1['City'].isnull()) & (df1['Venue'] == 'Sharjah Cricket Stadium'), 'City'] = 'Sharjah'
```

A new dataframe ‘target_df’ is created to store information about the target runs set by the team batting first in each match. The resulting dataframe ‘temp_df1’ contains all the columns from df1 and the additional 'Target' column from ‘target_df’.

```
# Creating new feature called 'Target'.

# The target is the number of runs a team needs to score to win a match.
# The team that bats first sets this target in the first innings,
# and the other team tries to score more runs than that to win in the second innings.

target_df = df2[df2['innings']==1].groupby('ID')['total_run'].sum().reset_index()
target_df.rename(columns={'total_run':'Target'}, inplace=True)
target_df['Target'] = target_df['Target'] + 1
target_df
```

Over the course of IPL seasons, teams might undergo name changes. This code ensures consistency by replacing old team names with their corresponding new names in relevant columns (Team1, Team2, and WinningTeam) within the ‘temp_df1’ dataframe. Therefore, it is very important that team names remain consistent to ensure accurate interpretation and analysis of the data.

```
# In the IPL, some teams have changed their names after a few seasons.
# 1. 'Deccan Chargers' changed their name to 'Sunrisers Hyderabad'.
# 2. 'Delhi Daredevils' changed their name to 'Delhi Capitals'.
# 3. 'Kings XI Punjab' changed their name to 'Punjab Kings'.
# Here, replacing team's old name with their new name in the temp_df1.

temp_df1['Team1'] = temp_df1['Team1'].replace('Deccan Chargers','Sunrisers Hyderabad')
temp_df1['Team2'] = temp_df1['Team2'].replace('Deccan Chargers','Sunrisers Hyderabad')

temp_df1['Team1'] = temp_df1['Team1'].replace('Delhi Daredevils','Delhi Capitals')
temp_df1['Team2'] = temp_df1['Team2'].replace('Delhi Daredevils','Delhi Capitals')

temp_df1['Team1'] = temp_df1['Team1'].replace('Kings XI Punjab','Punjab Kings')
temp_df1['Team2'] = temp_df1['Team2'].replace('Kings XI Punjab','Punjab Kings')

temp_df1['WinningTeam'] = temp_df1['WinningTeam'].replace('Deccan Chargers','Sunrisers Hyderabad')
temp_df1['WinningTeam'] = temp_df1['WinningTeam'].replace('Delhi Daredevils','Delhi Capitals')
temp_df1['WinningTeam'] = temp_df1['WinningTeam'].replace('Kings XI Punjab','Punjab Kings')
```

These subsequent lines filter the data in ‘temp_df1’, retaining only those rows whose ‘Team1’ and ‘Team2’ belong to teams which participated in all IPL seasons. This ensures that the dataset (temp_df1) is refined to include matches only among the specified set of teams that have been consistent participants across all IPL seasons.

```
# Teams that have participated in all the IPL seasons.

team_list = ['Rajasthan Royals',
 'Royal Challengers Bangalore',
 'Sunrisers Hyderabad',
 'Delhi Capitals',
 'Chennai Super Kings',
 'Kolkata Knight Riders',
 'Punjab Kings',
 'Mumbai Indians']
```

```
# Keeping the data of only those teams that have participated in all the seasons of the IPL in temp_df1.

temp_df1 = temp_df1.loc[temp_df1['Team1'].isin(team_list)]
temp_df1 = temp_df1.loc[temp_df1['Team2'].isin(team_list)]
```

There are only fifteen matches decided by the 'DLS' method. As matches affected by rain or other disruptions may have different dynamics, and including them in the training data is not right. Also, the matches decided by ‘Super Over’ are not considered regular and can significantly differ from standard match scenarios. By excluding DLS and Super Over decided matches, the analysis and

machine learning model will focus on typical and frequent match conditions, providing a more representative and meaningful dataset for the intended purposes.

```
# So, in matches decided by the DLS method, one inning didn't complete properly.
# Therefore, dropping the data where matches decided by DLS method.

temp_df1 = temp_df1[temp_df1['method'].isnull()]
```

```
# Superover is not regular type to decided match winner.
# Therefore, dropping the data where match winner decided by SuperOver.

temp_df1 = temp_df1[temp_df1['SuperOver']=='N']
```

As team names change from season to season. The 'BattingTeam' column in the df2 dataset is adjusted accordingly as well. 'BattingTeam' names are checked against the predefined list of teams (team_list), and keeping only those data where team has played all seasons of IPL.

```
# Here, replacing team's old name with their new name in the df2.

df2['BattingTeam'] = df2['BattingTeam'].replace('Deccan Chargers','Sunrisers Hyderabad')
df2['BattingTeam'] = df2['BattingTeam'].replace('Delhi Daredevils','Delhi Capitals')
df2['BattingTeam'] = df2['BattingTeam'].replace('Kings XI Punjab','Punjab Kings')
```

```
# Keeping the data of only those teams that have participated in all the seasons of the IPL in df2.

df2 = df2.loc[df2['BattingTeam'].isin(team_list)]
```

```
# Merging df2 with temp_df1 and storing it into temp_df2.

temp_df2 = temp_df1.merge(df2, on='ID')
temp_df2.sample(2)
```

A new feature named 'BowlingTeam' is created in the 'temp_df2' dataset. This feature is designed to represent the team that is currently bowling during a second inning of the match.

```
# Creating new feature called 'BowlingTeam'.

temp_df2['BowlingTeam'] = temp_df2.apply(lambda x: x['Team2'] if x['BattingTeam']==x['Team1'] else x['Team1'], axis=1)
```

A number of columns in the 'temp_df2' dataset are not needed for further analysis and model training. Therefore, dropping unnecessary columns from ‘temp_df2’.

Next step is vital for building a model that predicts match outcomes based on the second innings. The focus is on predicting the winner based on the second innings, where the team2 is chasing the target set by team1.

```
# Keeping the data of the second innings as the model will predict the winner using the data from the second innings.

# After team1 finishes batting, team2 takes its turn.
# The second inning is when team2 chases the target set by team1 in the first innings.

temp_df2 = temp_df2[temp_df2['innings']==2]
```

The ‘current_score’ feature cumulatively adds up the total runs scored by the batting team until the current ball, facilitating a dynamic representation of the team's progress. Addressing potential bugs, as runs lefts can’t be negative, its mathematical error, ‘runs_left’ feature ensures that negative values in 'runs_left_temp' are replaced with 0, reflecting the corrected count of remaining runs.

```
# Current batting score means the number of runs scored by the batting team till the current ball.

temp_df2['current_score'] = temp_df2.groupby('ID')['total_run'].cumsum()
```

```
# Creating new feature called 'runs_left_temp'.

# Runs left meaning is after scoring a certain number of runs, the remaining runs that the team needs to reach the

temp_df2['runs_left_temp'] = temp_df2['Target'] - temp_df2['current_score']
```

```
# Fixing bugs in the 'runs_left_temp' column and creating new feature called 'runs_left'.

temp_df2['runs_left'] = temp_df2.apply(lambda x: x['runs_left_temp'] if x['runs_left_temp']>=0 else 0, axis=1)
```

To handle null values in 'extra_type' column, ‘extra_type_modified’ feature is created, this feature replaces null values with 'not an extra', providing clarity in distinguishing between legal and extra

deliveries. In cricket, an over consists of six legal deliveries, but the 'ballnumber' column in the dataset incorrectly counted extra deliveries (wides, no-balls, or penalty) alongside the legal ones. That's why the count of ball numbers reached 10, whereas it should have been limited to 6. The dataset is sorted based on the 'ID' column in descending order. This step is important to maintain the chronological order of the matches. The 'ballnumber_modified' feature created that accurately store the count within the permissible range of 1 to 6 for a standard over. This step contributes to the overall data integrity, a fundamental aspect for subsequent analyses and machine learning model training.

```
# Creating new feature called 'extra_type_modified'.
# 'extra_type_modified' column contains null values where balls bowled are legal deliveries.
# Filling nan values in the column with 'not an extra'.

temp_df2['extra_type_modified'] = temp_df2['extra_type']
temp_df2['extra_type_modified'].fillna('not an extra', inplace=True)
```

```
# Column 'ballnumber_modified' ball number count.

temp_df2['ballnumber_modified'].value_counts()
```

```
1    15171
2    15110
3    15042
4    14916
5    14854
6    14243
0      615
Name: ballnumber_modified, dtype: int64
```

The 'balls left' column acts as a moving measure, representing the number of balls that remain for the chasing team to reach its target. The specific number is calculated by deducting the cumulative count of deliveries (considering overs and modified ball numbers) from the total available balls in the match plus six (126). The ‘wickets_data' feature is created to mark 'player out' with 1 and 0 for instances where a player is not out, providing a binary indicator. The

'wickets_left' column is created based on the 'wickets_data' feature. Initially, cumulative sum of 'wickets_data' for each match provides the count of wickets that have fallen until a given ball. Subtracting this count from the fixed total of 10 (there is 10 wickets in an inning) gives the 'wickets_left' column. Column represents the number of batsmen who are yet to bat in the ongoing innings.

```
# Creating new feature called 'balls_left'.
# After each delivery, one ball is deducted from the total number of balls.
# Balls left refers to the number of balls remaining to reach the remaining target.

temp_df2['balls_left'] = 126 - (((temp_df2['overs']+1)*6) + (temp_df2['ballnumber_modified']))
```

```
# Creating new feature called 'wickets_data'.
# In the wickets_data column replacing 'player out' with 1 and all remaining data (where player is not got out) w

temp_df2['wickets_data'] = temp_df2['player_out']
temp_df2['wickets_data'].fillna(0, inplace=True)
temp_df2['wickets_data'] = temp_df2.apply(lambda x: 1 if x['wickets_data']!=0 else x['wickets_data'], axis=1)
temp_df2['wickets_data'] = temp_df2['wickets_data'].astype(int)
```

```
# Creating new feature called 'wickets_left'.
# Wickets left means count of number of batsmen still yet to bat in the current innings.

wickets_count = temp_df2.groupby('ID')['wickets_data'].cumsum().values
temp_df2['wickets_left'] = 10 - wickets_count
```

To measure the scoring rate of batting team during its innings a ‘CRR’ feature is introduced. It shows the average number of runs scored per over, 'CRR' gives a good indication of how well things are going for the team at present. It is calculated as the product of the current score and 6 (number of balls per over), divided by the remaining balls in the inning. ‘RRR’ refers to the average number of runs required for the chasing team per over in order to reach their target and win a match. RRR, which is calculated as the ratio of remaining runs multiplied by six to remaining balls.

```
# Creating new feature called 'CRR'.
# Current run rate (CRR) is a statistical term where the average number of runs batting team i
# per over while batting.

temp_df2['CRR'] = (temp_df2['current_score']*6)/(120-temp_df2['balls_left'])
```

```
# Creating new feature called 'RRR'.
# Required run rate means the average number of runs chasing team needs to score per over to w

temp_df2['RRR'] = (temp_df2['runs_left']*6)/(temp_df2['balls_left'])
```

The 'match_result' column is created to categorize the outcome of each match. If the chasing team successfully achieves the target, the corresponding entry in 'match_result' is marked as 1. Conversely, if bowling team successfully defends the target, the entry is marked as 0. This binary classification is vital for training a machine learning model to predict match results based on various features. This is targeting variable of model.

```
temp_df2['match_result'] = temp_df2.apply(lambda x: 1 if x['BattingTeam']==x['WinningTeam'] else 0, axis=1)
```

Missing values are present in the 'RRR' column, where the chasing team has already achieved the target, these NaN entries are replaced with 0. The 'CRR' and 'RRR' columns contains infinite values under certain circumstances. Its mathematical errors while applying formulas. These cases are addressed to maintain data integrity. For 'CRR', if the number of balls left is the maximum (120), when bowler bowls first ball for extra runs, indicating the start of the inning, 'CRR' is set to 0. For 'RRR', when the remaining balls and runs are both 0, representing a completed chase, 'RRR' is set to 0. In cases where runs are remaining but no balls left, 'RRR' is set to a large value (500) to reflect the urgency of scoring with no balls remaining.

```
# Maximum value in the column 'CRR' and 'RRR' is infinite.
# Handling infinite values through logical replacements.

temp_df2['CRR'] = temp_df2.apply(lambda x: 0 if x['balls_left']==120 else x['CRR'], axis=1)
temp_df2['RRR'] = temp_df2.apply(lambda x: 0 if (x['balls_left']==0 and x['runs_left']==0) else (500 if (x['balls_left']==0 a
```

The dataset 'temp_df2' has undergone extensive feature engineering, resulting in the creation of twelve new features customized for training a machine learning model. These features include

various aspects crucial for predicting match outcomes. Now, selecting and keeping only the important columns which required for the next stage of processing and modeling.

```
1   City                 89951 non-null  object
2   WinningTeam          89951 non-null  object
3   Target               89951 non-null  int64
4   innings              89951 non-null  int64
5   overs                89951 non-null  int64
6   ballnumber           89951 non-null  int64
7   batter               89951 non-null  object
8   bowler               89951 non-null  object
9   non-striker          89951 non-null  object
10  extra_type           4743 non-null   object
11  batsman_run          89951 non-null  int64
12  extras_run           89951 non-null  int64
13  total_run            89951 non-null  int64
14  non_boundary         89951 non-null  int64
15  isWicketDelivery     89951 non-null  int64
16  player_out           4411 non-null   object
17  kind                 4411 non-null   object
18  fielders_involved    3105 non-null   object
19  BattingTeam          89951 non-null  object
20  BowlingTeam          89951 non-null  object
21  current_score        89951 non-null  int64
22  runs_left_temp       89951 non-null  int64
23  runs_left            89951 non-null  int64
24  extra_type_modified  89951 non-null  object
25  ballnumber_modified  89951 non-null  int64
26  balls_left           89951 non-null  int64
27  wickets_data         89951 non-null  int32
28  wickets_left         89951 non-null  int32
29  CRR                  89951 non-null  float64
30  RRR                  89923 non-null  float64
31  match_result         89951 non-null  int64
```

Final DataFrame:

The new dataframe 'new_df' has carefully created for machine learning, containing relevant features and the target variable. It consists of 89951 rows and 10 columns, with no missing or infinite values, this clean and processed dataset, comprising both independent and dependent variables, this processed dataset is now ready for effective model training, offering valuable insights into IPL match outcomes.

```
new_df.head()
```

	BattingTeam	BowlingTeam	City	Target	runs_left	balls_left	wickets_left	CRR	RRR	match_result
127	Rajasthan Royals	Royal Challengers Bangalore	Ahmedabad	158	158	119	10	0.0	7.966387	1
128	Rajasthan Royals	Royal Challengers Bangalore	Ahmedabad	158	158	118	10	0.0	8.033898	1
129	Rajasthan Royals	Royal Challengers Bangalore	Ahmedabad	158	152	117	10	12.0	7.794872	1
130	Rajasthan Royals	Royal Challengers Bangalore	Ahmedabad	158	148	116	10	15.0	7.655172	1
131	Rajasthan Royals	Royal Challengers Bangalore	Ahmedabad	158	148	115	10	12.0	7.721739	1

```
# Number of rows and columns in the new_df.

new_df.shape
```

```
(89951, 10)
```

The 'new_df' dataset has been shuffled to removes any potential bias during the model development part. Shuffling prevents the model from learning patterns specific to its particular grouping of match related data, it ensures a more robust and unbiased distribution of data.

```
# Shuffling the new_df dataset to avoid bias in model development.
# As currently, all data related to a particular match is grouped together.

new_df = new_df.sample(new_df.shape[0])
```

The scatter plot in the ‘Figure 22’ clearly shows inverse relationship between the target and the match result. A smaller target means a higher probability of victory, indicated by a match result

of 1. On other side, higher target is associated with a lower chance of winning, represented by a match result of 0.

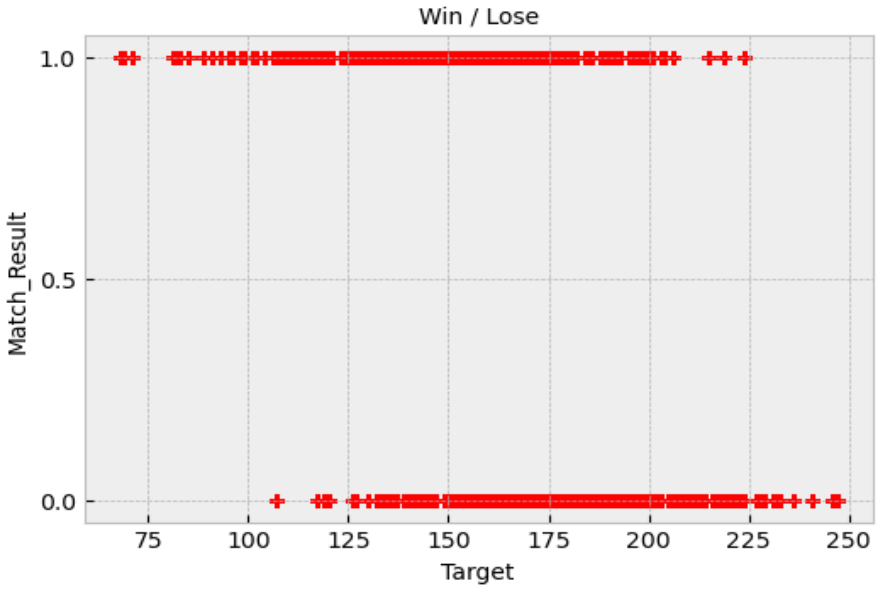

Figure 22: Win and Lose Based on Target

Machine Learning:

In preparing the dataset for machine learning model, features are chosen as independent variables, 'BattingTeam', 'BowlingTeam', 'City', 'Target', 'runs_left', 'balls_left', 'wickets_left', 'CRR', 'RRR' store in X to predict the dependent variable, which is the 'match_result' store in y.

```
# Selecting features for the machine learning model from 'new_df'.
X = new_df[['BattingTeam','BowlingTeam','City','Target','runs_left','balls_left','wickets_left','CRR','RRR']]
# Selecting the target variable for the model, which is 'match_result'.
y = new_df['match_result']
```

The train_test_split function imported from the scikit-learn library, this method divides dataset into two parts, one for training the model and other is for testing its performance. The resulting shapes includes, training set around 71,960 examples, and the testing set about 17,991 examples.

```
from sklearn.model_selection import train_test_split # Importing train_test_split from sklea
# Here, splitting the dataset into training and testing sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
# Shapes of the training and testing feature sets.
print(X_train.shape, X_test.shape)
```

In the dataset, features are categorized as categorical or non-categorical variables. The categorical columns, present in dataset are 'BattingTeam,' 'BowlingTeam,' and 'City'. On the other hand, non-categorical columns, including 'Target,' 'runs_left,' 'balls_left,' 'wickets_left,' 'CRR' (Current Run Rate), and 'RRR' (Required Run Rate), involve numerical data.

```
# List of categorical columns.
cat_cols = ['BattingTeam','BowlingTeam','City']
# List of non-categorical columns.
non_cat_cols = ['Target','runs_left','balls_left','wickets_left','CRR','RRR']
```

Logistic regression:

Logistic regression is a statistical model adapted for binary classification problems, particularly in supervised learning. It serves to predict the probability of an instance belonging to a specific class, typically denoted as True/False or 0/1. Optimizing the logistic or sigmoid function, input features are transformed into probabilities, facilitating accurate predictions. Renowned for its simplicity, efficiency, and interpretability, logistic regression is a preferred choice for two-sided outcomes, such as win or loss predictions in sports analytics. Its versatility shines in modeling the relationship between features and the likelihood of a team winning, effectively capturing the complex dynamics inherent in cricket matches.

The code uses a scikit-learn tool to set up a machine learning pipeline for the logistic regression model. This pipeline involves two main steps which are data transformation and model training. The 'ColumnTransformer' class is used to handle categorical and non-categorical columns separately.

Data Transformation:

The 'ColumnTransformer' is structured with two transformers:

'OHE' (One-Hot Encoder): This transformer applies one-hot encoding to categorical columns ('BattingTeam,' 'BowlingTeam,' 'City'). One-hot encoding converts categorical variables into binary

vectors, create separate columns for each category and shows the presence or absence of that category with binary values (0/1). Also, drops the first category for each categorical column to avoids the multicollinearity issue.

'PASS': This transformer passes through non-categorical columns ('Target,' 'runs_left,' 'balls_left,' 'wickets_left,' 'CRR,' 'RRR') without any transformation, as they already contain numerical data.

```
from sklearn.compose import ColumnTransformer  # Importing the ColumnTrans
from sklearn.preprocessing import OneHotEncoder  # Importing the OneHotEnc
```

```
# Creating a ColumnTransformer for column transformations.
# 'OHE' for categorical columns. (onehotencoding apply)
# 'PASS' for non-categorical columns. (passed without transformation)

transformer = ColumnTransformer(transformers=[
        ('OHE', OneHotEncoder(sparse=False, drop='first'), cat_cols),
        ('PASS', 'passthrough', non_cat_cols)
    ],remainder='drop')
```

Model Training:

The 'Pipeline' class is used to assemble the sequence of transformations and the final logistic regression model.

'STEP1': This step represents the data transformation, which includes both one-hot encoding for categorical columns and passing through non-categorical columns.

'STEP2': This step involves the logistic regression model with the 'liblinear' solver, suitable for binary classification tasks.

```
from sklearn.linear_model import LogisticRegression # Importing the LogisticRe
from sklearn.pipeline import Pipeline # Importing the Pipeline from sklearn li
from sklearn import set_config # Importing set_config to display the pipeline.
set_config(display='diagram')
```

```
# Creating pipeline for one hot encoding and logistic regression model.

pipe = Pipeline([
    ('STEP1', transformer),
    ('STEP2', LogisticRegression(solver='liblinear'))
])
```

The next step, training the machine learning model using the training data. Visualization of created pipeline is below in the ‘Figure 23’.

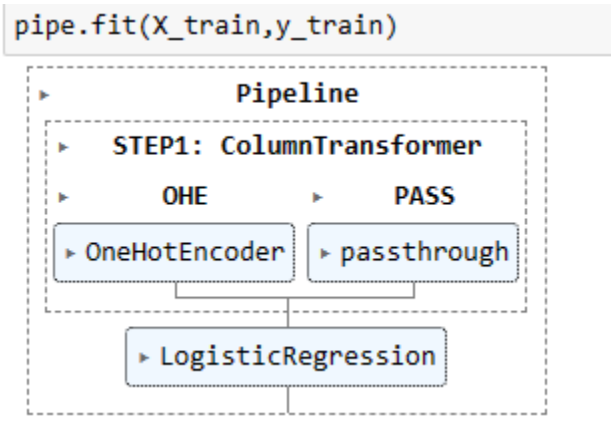

Figure 23: Pipeline

Next, Weights of features, each feature has different significance in the context of predicting the target variable. The weights assigned to these features which shown in the ‘Figure 24’ indicate their impact on the logistic regression model. Considering features like 'BattingTeam_Delhi Capitals,’ and 'City_Dubai' carry negative weights, tells that an increase in these factors tends to decrease the likelihood of the target variable. On other hand, features like 'BowlingTeam_Sunrisers Hyderabad,' 'RRR,' and 'runs_left' have positive weights, tells that an increase in these factors contributes positively to the prediction of the target variable. 'City_Cape Town' holds the lowest weight of -2.31 among all features. This shows that matches played in Cape Town are less favorable for successful chases, suggesting that the team batting first often defends their total effectively at this venue. With the highest weight of +2.50 among all features, 'City_Indore' indicates that matches played in Indore significantly favor successful chases, suggesting that the team batting second in Indore holds a considerable advantage in winning the

match. The analysis of feature weights helps in understanding the importance of each factor in the prediction of model.

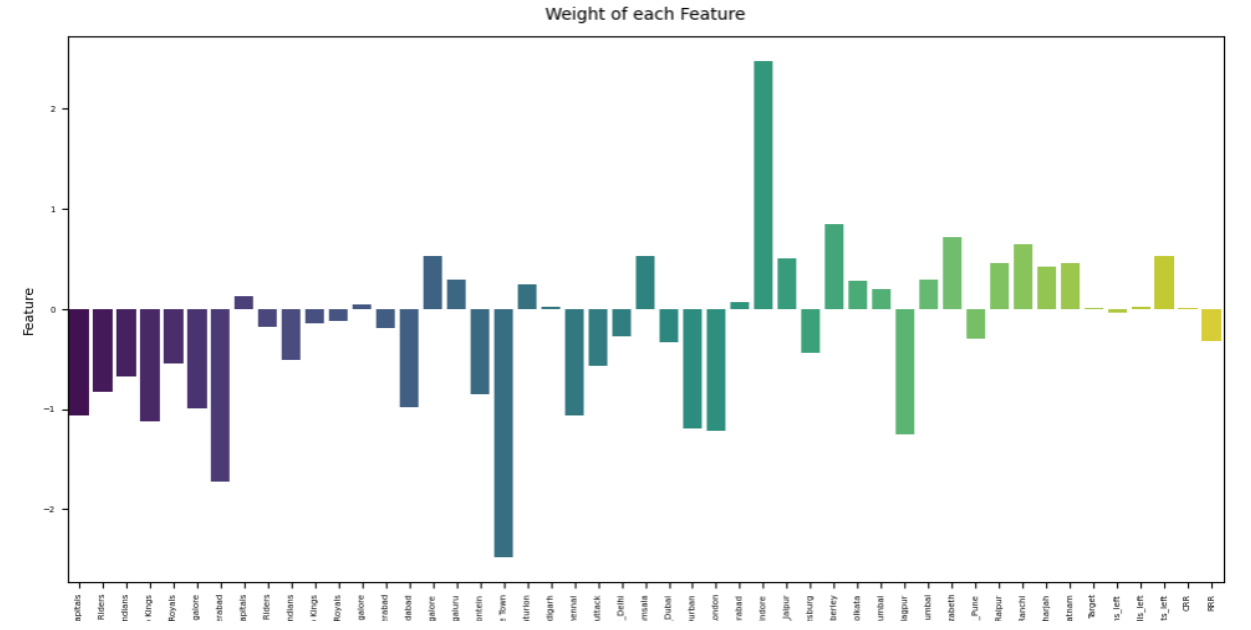

Figure 24: Weight of Each Feature

The logistic regression model, trained using the created pipeline, achieved an accuracy score of approximately **80.63%** on the test set (X_test and y_test). Similarly, the model got accuracy score of around **80.91%** on the training set (X_train and y_train), indicating a good fit to the training data. The accuracy scores show how well the model can predict outcome of match.

```
# Predictions on test set.

y_pred = pipe.predict(X_test)
```

```
# Predictions on training set.

y_pred_train = pipe.predict(X_train)
```

```
# Calculating accuracy score on test set.

accuracy_score(y_test,y_pred)
```

```
# Calculating accuracy score on the train set.

accuracy_score(y_train,y_pred_train)
```

The cross-validated accuracy score, obtained by splitting the dataset (X and y) into five subsets (cv=5), and the model trained and evaluated five times, result a mean accuracy of approximately 80.8%. Which shows that the model is neither overfitting nor underfitting. This score gives a better idea of the model's overall performance on different subsets of data. It shows that accuracies in predicting match outcomes are more or less maintained at a constant level across the various sets.

```
from sklearn.model_selection import cross_val_score # # Importing cross_val_s
```

```
# Performing cross-validated to check accuarcy.
# X, y split five times, and model is trained and evaluated on each time.

cross_val_score(pipe, X, y, cv=5, scoring='accuracy').mean()
```

The confusion matrix visually represents the performance of the machine learning model on the test set. This heatmap map shows, the distribution of correct and incorrect predictions. The classification report gives detailed evaluation of model performance. For class 0, representing instances where the team failed to chase the target. And for class 1, indicates successful run chase chases. These metrics shows balanced performance of the model in predicting match outcomes.

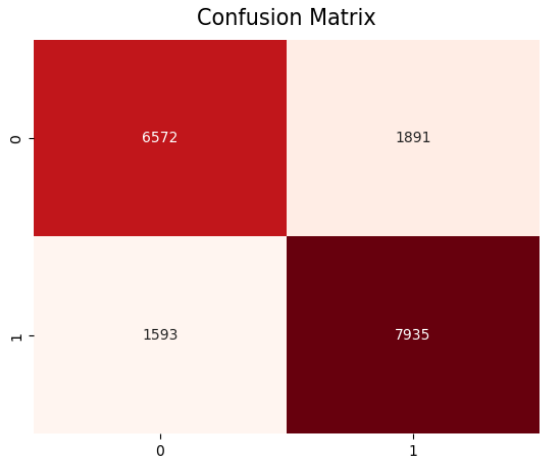

Figure 25: Confusion Matrix

	precision	recall	f1-score	support
0	0.80	0.78	0.79	8463
1	0.81	0.83	0.82	9528
accuracy			0.81	17991
macro avg	0.81	0.80	0.81	17991
weighted avg	0.81	0.81	0.81	17991

Figure 26: Classification Report

Function created named ‘find_win_probability’ which calculates and returns the winning probabilities for both batting and bowling teams throughout the second innings after each over. The function takes a match ID as input and uses the trained machine learning model (pipe) to predict the winning probabilities. It iterates through each over of the innings, updating the target runs and wickets left after each over.

The dataframe ‘prob_df’ (returns) with columns such as over number, runs scored, wickets gone, batting team's winning probability, and bowling team's winning probability. In with match ID 1178400 (Chennai Super Kings vs. Rajasthan Royals), the function returns a dataframe (temp_df) showing winning probabilities after each over. Visual representation in the ‘Figure 17’ shows the dynamic view of the win/lose probability for both the batting team and bowling team after each over's performance.

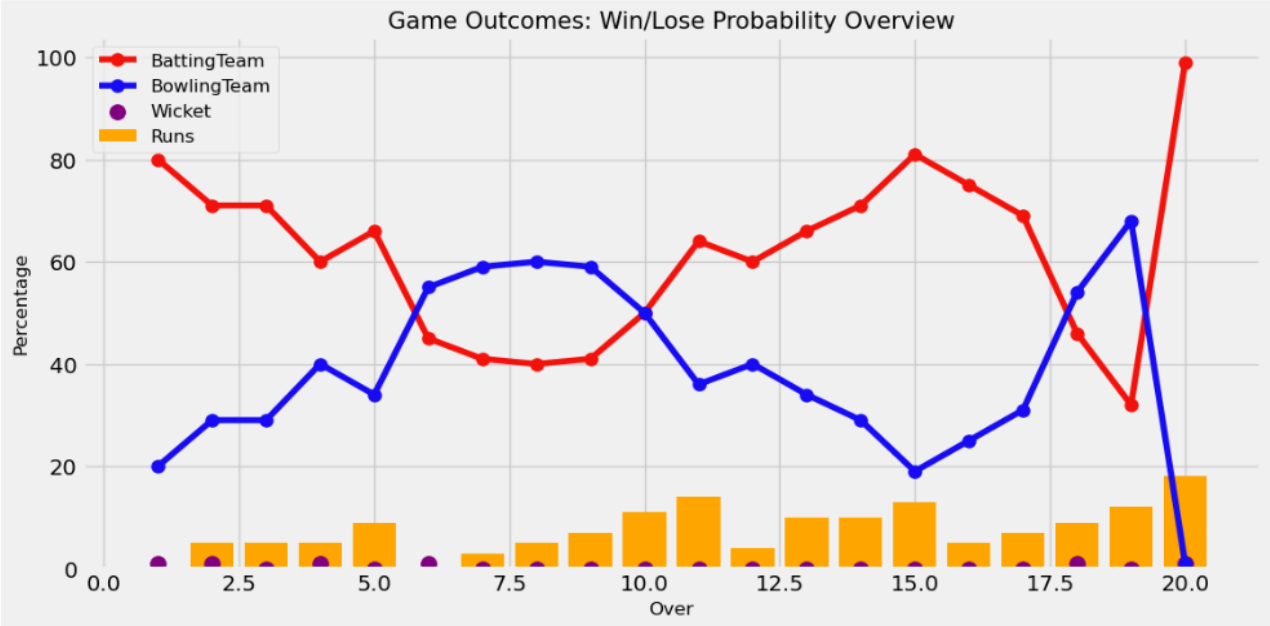

Figure 27: Win and Lose Probability after each Over

Model Deployment:

In the final steps of the project, the trained machine learning model saved and then loaded for deployment. The pipeline, consisting of the logistic regression model and preprocessing steps, saved to a file named 'IPL_prediction.pkl' using the pickle module, model file can be easily stored allows for reuse.

```
import pickle # Importing the pickle module.
```

```
# Saving trained machine learning pipeline.
pickle.dump(pipe, open('IPL_prediction.pkl','wb'))
```

```
# Loading a saved machine learning pipeline ('IPL_prediction.pkl')
loaded_model = pickle.load(open('IPL_prediction.pkl','rb'))
```

To make predictions on new data, a function ‘input_df’ created which takes raw input data and convert it into a dataFrame for input into the loaded machine learning model.

```
# Creating function that takes raw data as input and returns datafra

def input_df(bat,bowl,city,target,runs,balls,wickets,crr,rrr):

    input_data = pd.DataFrame([{
        'BattingTeam':bat,
        'BowlingTeam':bowl,
        'City':city,
        'Target':target,
        'runs_left':runs,
        'balls_left':balls,
        'wickets_left':wickets,
        'CRR':crr,
        'RRR':rrr,
    }])

    return input_data
```

An example for prediction demonstrated using the loaded model. Considering, the input data represented a match between 'Royal Challengers Bangalore' (batting team) and 'Kolkata Knight Riders' (bowling team) in the city of 'Mumbai,' with a target score of 180, 180 runs left to scored, 120 balls remaining, 10 wickets left, a current run rate (CRR) of 0, and a required run rate (RRR) of 9.00. The loaded model predicted a 'High Probability of LOSS' for the batting team. This shows

the successful deployment and utilization of the trained model for real-time predictions on new match scenarios.

```
data = input_df('Royal Challengers Bangalore','Kolkata Knight Riders','Mumbai',180,180,120,10,0,9.00)
# Printing answer.
if loaded_model.predict(data)[0]==1:
    print('High Probability of WIN')
else:
    print('High Probability of LOSS')
```

```
High Probability of LOSS
```

CHALLENGES

The project's journey is filled with valuable insights and information, not without its challenges and has its equal share. The handling and cleaning of the raw data was one of the biggest obstacles encountered. Given the various sources and formats of the data, reconciling inconsistencies, dealing with missing values, and ensuring data uniformity posed a significant challenge. The need to implement robust data preprocessing methods and thorough exploratory data analysis (EDA) was crucial in overcoming this challenge. Next challenge was encountered during the feature engineering phase. Developing new features for the model's prediction capabilities required a clear understanding of cricket context and IPL dataset. As for the calculation of remaining balls, how many wickets left to bowl out and dynamic current and required run rates needs special attention as well as deep knowledge about this domain. Addressing bugs and inaccuracies in the columns was an ongoing challenge, particularly when it came to counting balls accurately, which holds the wrong data. The process of bug fixing involved regular reviews and debugging sessions. Where, carefully examine the code, identify errors, and make necessary corrections to ensure that the model provided accurate and reliable results. In the realm of machine learning, refining the logistic regression model to achieve the best predictive performance posed its own challenges. Lastly, In the future scope and scalability of the project introduced challenges related to real-time data integration, external factor or variable consideration. In fact, overcoming these difficulties has been an important part of the project's development and innovation.

FUTURE SCOPE

Data Integration in real time, use programs like APIs or web scraping to get live match predictions by using real-time data. Let the model change as game situations happen quickly and give fast information. Considering external factors, such as look into combining things outside like the weather condition, player injury and team connections. Create more community involvement by using social media to show model ideas, predictions and news. Encourage discussions and feedback to flow excitement between fans. Work with cricket experts, statisticians or ex-players to get knowledge from their experience. Use their understanding to improve the model's prediction, which will result in better and more precise predictions. Develop interactive app or website that gives detailed data about cricket match scorecard, players, teams, batting, bowling, fielding, etc. Use the study done to give interesting pictures and info for cricket fans.

CONCLUSION

This project unfolds as a meticulous exploration of IPL cricket. It looks closely at information, finds out what it means and makes smart predictions using them to capture the fast changing nature of this sport. The initial stages of univariate, bivariate, and multivariate analysis important for finding patterns and getting useful information from crucial columns needed to understand the game's complex parts.

By combining the datasets using the common 'ID' column, created a unified dataset called 'df.' This approach enabled a thorough analysis, considering various factors together for a more complete understanding. The subsequent phases of batting, bowling and fielding analysis delved deep into player performances, offering a great understanding about individual contributions to the game. A notable feature emerged in the form of functions capable of furnishing detailed career statistics for a given player a testament to the project's commitment to providing valuable insights at the micro-level. Team bases analyses provided a panoramic view of IPL dynamics, showing how each team did over different periods. The in-depth exploration of match data, aspects like toss outcomes and target settings provided a proper perspective on the multifaceted nature of cricket matches.

As traverse the journey from data analysis to making predictions, the mix of statistical exploration and machine learning has become a strong tool for predicting outcomes. Feature engineering played crucial part, with the creation of new features like 'CRR,' 'RRR,' 'balls_left,' and 'wickets_left,' offering deeper insights into the evolving dynamics of a cricket match. The

logistic regression model, carefully trained and tested, is now a proof of the project's ability to predict. It gives win/loss probabilities of the match after each over in real time.

Looking forward, the project's future scope tells future possibilities like real-time data integration, external factor assimilation, community engagement, and expert collaboration promise continued refinement and innovation in the realm of sports analytics. Making an app or website that gives cricket fans lots of numbers and facts, makes them like it even more.

In the end, the project is not only looking at numbers and goes beyond data analysis, it is an exploration into the heart of IPL cricket, unravel the game's complexities using data science, unveiling patterns that enhance understanding of cricket.

APPENDIX

1. Code of function which gives batsmen statistics.

```
def find_data(val):
    data_list = []
    matches = df1[df1['Team1Players'].str.contains(val)].shape[0] +
df1[df1['Team2Players'].str.contains(val)].shape[0]
    runs = sum(df[(df['batter']==val) & (df['innings'].isin([1,2]))]['batsman_run'])
    temp_df = df[(df['innings'].isin([1,2])) & (df['batter']==val)]
    ball_faced = (temp_df.shape[0]) - (temp_df[temp_df['extra_type']=='wides'].shape[0])
    out = df[(df['innings'].isin([1,2])) & (df['player_out']==val)].shape[0]
    if out!=0:
        avg = round(runs/out,2)
    else:
        avg = 0
    strike_rate = round((runs/ball_faced)*100,2)
    fours = df[(df['batter']==val) & (df['innings'].isin([1,2])) & (df['batsman_run']==4) &
(df['non_boundary']==0)].shape[0]
    sixes = df[(df['batter']==val) & (df['innings'].isin([1,2])) & (df['batsman_run']==6) &
(df['non_boundary']==0)].shape[0]
    temp_list1 = []
    temp_list2 = []
    temp_list3 = []
    for data in season_list:
        high_score = 0
        fifty = 0
        try:
```

```
        high_score = df[(df['batter']==val) & (df['innings'].isin([1,2])) &
(df['Season']==data)].groupby('MatchNumber')['batsman_run'].sum().sort_values(ascending=False).valu
es[0]
        fifty = df[(df['batter']==val) & (df['innings'].isin([1,2])) &
(df['Season']==data)].groupby('MatchNumber')['batsman_run'].sum()
        fifty = fifty[(fifty>=50) & (fifty<100)].shape[0]
        hundred = df[(df['batter']==val) & (df['innings'].isin([1,2])) &
(df['Season']==data)].groupby('MatchNumber')['batsman_run'].sum()
        hundred = hundred[hundred>=100].shape[0]
      except Exception:
        high_score = 0
        fifty = 0
        hundred = 0
      temp_list1.append(high_score)
      temp_list2.append(fifty)
      temp_list3.append(hundred)
   high_score = max(temp_list1)
   fifty = sum(temp_list2)
   hundred = sum(temp_list3)
   data_list = data_list + [val, matches, runs, ball_faced, avg, strike_rate, fours, sixes, high_score, fifty,
hundred]
   return data_list
```

2. Code of fixing bugs in ballnumber column.

```
match_id_list = temp_df2['ID'].value_counts().index.sort_values(ascending=False).tolist()
ball_no_list = []
for val in match_id_list:
   temp_data = temp_df2[temp_df2['ID']==val]
```

```
    data = temp_data['extra_type_modified'].tolist()
    final_list = []
    new_list = []
    count = 1
    prev_val = 0
    for i in range(len(data)):
        if data[i]=='wides' or data[i]=='noballs' or data[i]=='penalty':
            new_list.append(prev_val)
        else:
            prev_val = count
            new_list.append(count)
            count = count + 1
        if i==(len(data)-1):
            final_list = final_list + new_list
            break
        if count==7:
            final_list = final_list + new_list
            new_list = []
            count = 1
            prev_val = 0
    ball_no_list = ball_no_list + final_list
temp_df2['ballnumber_modified'] = ball_no_list
```

3. Code of function which gives winning probability after each over.

```
prob_df = pd.DataFrame()
def find_win_probability(match_id):
```

```
    temp_df = temp_df2[(temp_df2['ID']==match_id) & (temp_df2['ballnumber_modified']==6)]
    temp_df=
temp_df[['BattingTeam','BowlingTeam','City','Target','runs_left','balls_left','wickets_left','CRR','RRR']]
    loop_run = temp_df.shape[0]
    target = temp_df['Target'].values[0]
    wickets_left = 10
    print('BattingTeam:', temp_df['BattingTeam'].values[0])
    print('BowlingTeam:', temp_df['BowlingTeam'].values[0])
    print('Target:', target)
    list1 = []
    list2 = []
    list3 = []
    list4 = []
    list5 = []
    for i in range(loop_run):
        val = temp_df.iloc[i:i+1, :]
        over_no = (i+1)
        runs_scored = target - val['runs_left'].values[0]
        wickets = wickets_left - val['wickets_left'].values[0]
        prediction = pipe.predict_proba(val)
        win_bat_percent = round(prediction[0,1],2)*100
        win_bowl_percent = round(prediction[0,0],2)*100
        list1.append(over_no)
        list2.append(runs_scored)
        list3.append(wickets)
        list4.append(win_bat_percent)
        list5.append(win_bowl_percent)
        target = val['runs_left'].values[0]
```

```
        wickets_left = val['wickets_left'].values[0]
    prob_df['over_no'] = list1
    prob_df['runs_scored'] = list2
    prob_df['wickets_gone'] = list3
    prob_df['bat_win_per'] = list4
    prob_df['bowl_win_per'] = list5
    return prob_df


plt.style.use('fivethirtyeight')
plt.figure(figsize=(15,10))
plt.plot(temp_df['over_no'], temp_df['bat_win_per'], marker='.', markersize=16, linewidth=4,
color='#F3120B', label='BattingTeam')
plt.plot(temp_df['over_no'], temp_df['bowl_win_per'], marker='.', markersize=16, linewidth=4,
color='#160BF3', label='BowlingTeam')
plt.bar(temp_df['over_no'], temp_df['runs_scored'], color='orange', label='Runs')
plt.scatter(temp_df['over_no'], temp_df['wickets_gone'], color='purple', marker='o', label='Wicket',
s=100)
plt.title('Game Outcomes: Win/Lose Probability Overview', fontsize=22)
plt.xlabel('Over', fontsize=20)
plt.ylabel('Percentage', fontsize=20)
plt.legend(fontsize='medium')
plt.show()
```

REFERENCES

1. Nimmagadda, A., Kalyan, N. V., Venkatesh, M., Teja, N. N. S., & Raju, C. G. (Year not provided). Cricket score and winning prediction using data mining. International Journal of Advance Research and Development, 3(3). Retrieved from www.ijarnd.com.

2. Petersen, C., Pyne, D. B., Portus, M. J., & Dawson, B. (2008). Analysis of Twenty/20 Cricket performance during the 2008 Indian Premier League. International Journal of Performance Analysis in Sport, 8(3), 63-69. doi: 10.1080/24748668.2008.11868448.

3. Vistro, D. M., Rasheed, F., & David, L. G. (2019). The Cricket Winner Prediction with Application of Machine Learning and Data Analytics. International Journal of Scientific & Technology Research, 8(9), ISSN 2277-8616.

4. Sudhamathy, G., & Raja Meenakshi, G. (2020). Prediction on IPL Data Using Machine Learning Techniques in R Package. ICTACT Journal on Soft Computing, 11(1), ISSN: 2229-6956 (online). doi: 10.21917/ijsc.2020.0313.

5. Saikia, H., Bhattacharjee, D., & Lemmer, H. H. (2012). Predicting the Performance of Bowlers in IPL: An Application of Artificial Neural Network. International Journal of Performance Analysis in Sport, 12(1), 75-89. doi: http://dx.doi.org/10.1080/24748668.2012.11868584.

6. Jaipuria, S., & Jha, S. K. (2022). A study on the influence of toss result, toss decision, and venue on the outcome of IPL cricket match. Umsawli, Shillong, Int. J. Sport Management and Marketing, 22(3/4).

7. Singh, S., & Kaur, P. (2017). IPL Visualization and Prediction Using HBase. Information Technology and Quantitative Management (ITQM2017), Procedia Computer Science, 122, 910–915. doi: 10.1016/j.procs.2017.11.454.

8. Sinha, S., Tripathi, P., Vishwakarma, A., & Sankhe, A. (2020). IPL Win Prediction System to Improve Team Performance using SVM. International Journal of Future Generation Communication and Networking, 13(1s), 17-23.

9. Singhal, A., Agarwal, D., Singh, E., Valecha, R., & Malik, R. (2023). IPL Analysis and Match Prediction. In V. Bhateja, K. V. N. Sunitha, Y. W. Chen, & Y. D. Zhang (Eds.), Intelligent System Design (Vol. 494, Lecture Notes in Networks and Systems). Springer, Singapore. https://doi.org/10.1007/978-981-19-4863-3_3.

10. Thorat, P., Buddhivant, V., & Sahane, Y. (April 2021). Review paper on cricket score prediction. Engineering, Information Technology, VPPCOE & VA, Mumbai, Maharashtra, India.

11. Patil, N. M., Sequeira, B. H., Gonsalves, N. N., & Singh, A. A. Cricket team prediction using machine learning techniques. Fr. Conceicao Rodrigues College of Engineering, Mumbai, Maharashtra, India.

12. Kapadia, K., Abdel-Jaber, F., Thabtah, F., & Hadi, W. (November 2019). Sport analytics for cricket game results using machine learning: An experimental study.