VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1b: PERFORMANCE ANALYSIS AND SALARY CORRELATION IN IPL PLAYERS

AAKASH KATHIRVEL

V01110153

Date of Submission: 18-06-2024

CONTENTS

INTRODUCTION:

The aim of this analysis is to evaluate the performance metrics of IPL players, focusing on runs scored by batsmen and wickets taken by bowlers across recent seasons. By organizing data round-wise and match-wise, the analysis identifies the top three run-getters and wicket-takers in each IPL round. Additionally, it fits appropriate statistical distributions to the performance data of these top players from the last three IPL tournaments, with a specific focus on analyzing Harpreet Brar's runs scored.

Moreover, the analysis seeks to explore the relationship between a player's performance and their salary, providing insights into whether player remuneration aligns with their on-field contributions. Visual representations through histograms, boxplots, and pie charts will enhance the understanding of performance trends and their financial implications, offering valuable insights for team management and strategic decision-making in future IPL seasons.

OBJECTIVES:

1. Extract the files in R/Python

2. Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.

3. Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.

4. Find the relationship between a player’s performance and the salary he gets in your data.

5. Last three-year performance with latest salary 2024.

6. Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years.

RESULTS AND INTERPRETATION:

1. Extract the files in R/Python

```
# Reading the file into R
data <- read.csv("C:/Users/Aakash/Desktop/SCMA/IPL_ball_by_ball_updated till 2024.csv")
# Display the first few rows of the data
head(data)
tail(data)

df <- read_excel("C:/Users/Aakash/Desktop/SCMA/IPL SALARIES 2024.xlsx", sheet = 1)
head(df)
tail(df)
```

2. Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.

```
# Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match
df_roundwise <- data %>%
  arrange(Season,Match_id)
head(df_roundwise)

#Arrange the data for each batsman
batsman_data=df_roundwise%>%
  group_by(Match_id, Date,Striker) %>%
  summarise(
    Total_Balls_Faced =n(),
    Total_Runs_Scored=sum(runs_scored),
    .groups='drop'
  )
str(batsman_data)

#Arrange the data for each bowler
bowler_data=df_roundwise%>%
  filter(wicket_confirmation==1) %>%
  group_by(Match_id, Date,Bowler) %>%
  summarise(
    Total_Wickets_Taken=n(),
    .groups='drop'
  )
str(bowler_data)
str(df_roundwise)
```

```
# Top three run-getters and wicket-takers in each IPL round
top_performers <- df_roundwise %>%
  group_by(Season, Batting_team, Striker) %>%
  summarize(total_runs = sum(runs_scored), .groups = 'drop') %>%
  arrange(desc(total_runs)) %>%
  top_n(3, total_runs)

top_bowlers <- df_roundwise %>%
  group_by(Season, Bowling_team, Bowler) %>%
  summarize(total_wickets = sum(wicket_confirmation), .groups = 'drop') %>%
  arrange(desc(total_wickets)) %>%
  top_n(3, total_wickets)
```

Top Three Run Getters:

	Season	Striker	runs_scored
0	2007/08	SE Marsh	616
1	2007/08	G Gambhir	534
2	2007/08	ST Jayasuriya	514
3	2009	ML Hayden	572
4	2009	AC Gilchrist	495
5	2009	AB de Villiers	465
6	2009/10	SR Tendulkar	618
7	2009/10	JH Kallis	572
8	2009/10	SK Raina	528
9	2011	CH Gayle	608
10	2011	V Kohli	557
11	2011	SR Tendulkar	553
12	2012	CH Gayle	733
13	2012	G Gambhir	590
14	2012	S Dhawan	569
15	2013	MEK Hussey	733
16	2013	CH Gayle	720
17	2013	V Kohli	639
18	2014	RV Uthappa	660
19	2014	DR Smith	566
20	2014	GJ Maxwell	552
21	2015	DA Warner	562
22	2015	AM Rahane	540
23	2015	LMP Simmons	540
24	2016	V Kohli	973
25	2016	DA Warner	848
26	2016	AB de Villiers	687
27	2017	DA Warner	641

Top Three Wicket Takers:

	Season	Bowler	wicket_confirmation
0	2007/08	Sohail Tanvir	24
1	2007/08	IK Pathan	20
2	2007/08	JA Morkel	20
3	2009	RP Singh	26
4	2009	A Kumble	22
5	2009	A Nehra	22
6	2009/10	PP Ojha	22
7	2009/10	A Mishra	20
8	2009/10	Harbhajan Singh	20
9	2011	SL Malinga	30
10	2011	MM Patel	22
11	2011	S Aravind	22
12	2012	M Morkel	30
13	2012	SP Narine	29
14	2012	SL Malinga	25
15	2013	DJ Bravo	34
16	2013	JP Faulkner	33
17	2013	R Vinay Kumar	27
18	2014	MM Sharma	26
19	2014	SP Narine	22
20	2014	B Kumar	21
21	2015	DJ Bravo	28
22	2015	SL Malinga	26
23	2015	A Nehra	25
24	2016	B Kumar	24
25	2016	SR Watson	23
26	2016	YS Chahal	22
27	2017	B Kumar	28

INTERPRETATION: The provided code organizes and aggregates IPL data to analyze player performance. It first arranges the dataset by season and match, ensuring chronological order. It then groups the data by match, date, and batsman to calculate the total balls faced and runs scored by each batsman in each match. Similarly, it groups the data by match, date, and bowler, filtering for deliveries where a wicket was confirmed, to calculate the total wickets taken by each bowler per match. Additionally, it identifies the top three run-getters and wicket-takers in each IPL round, summarizing the total runs scored and wickets taken. This comprehensive arrangement facilitates detailed analysis of individual player performances across matches and seasons.

3. Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.

```r
# Fit the most appropriate distribution for the top three batsmen and bowlers in the last three IPL tournaments
last_three_seasons <- df_roundwise %>% filter(Season %in% tail(unique(Season), 3))

# Fit distributions for top batsmen
top_batsmen <- last_three_seasons %>%
  filter(Striker %in% unique(top_performers$Striker)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs_scored), .groups = 'drop')

top_batsmen_dist <- fitdist(top_batsmen$total_runs, "norm")

# Fit distributions for top bowlers
top_bowlers <- last_three_seasons %>%
  filter(Bowler %in% unique(top_bowlers$Bowler)) %>%
  group_by(Bowler) %>%
  summarize(total_wickets = sum(wicket_confirmation), .groups = 'drop')

top_bowlers_dist <- fitdist(top_bowlers$total_wickets, "pois")

# Fitting distribution for Harpreet Brar
harpreet_brar_runs <- last_three_seasons %>%
  filter(Striker == "Harpreet Brar") %>%
  dplyr::select(runs_scored)
harpreet_brar_runs

# Check if the resulting runs are numeric and have more than one element
if (is.numeric(harpreet_brar_runs$runs_scored) && length(harpreet_brar_runs$runs_scored) > 1) {
  harpreet_brar_dist <- fitdist(harpreet_brar_runs$runs_scored, "norm")
  print(summary(harpreet_brar_dist))
```

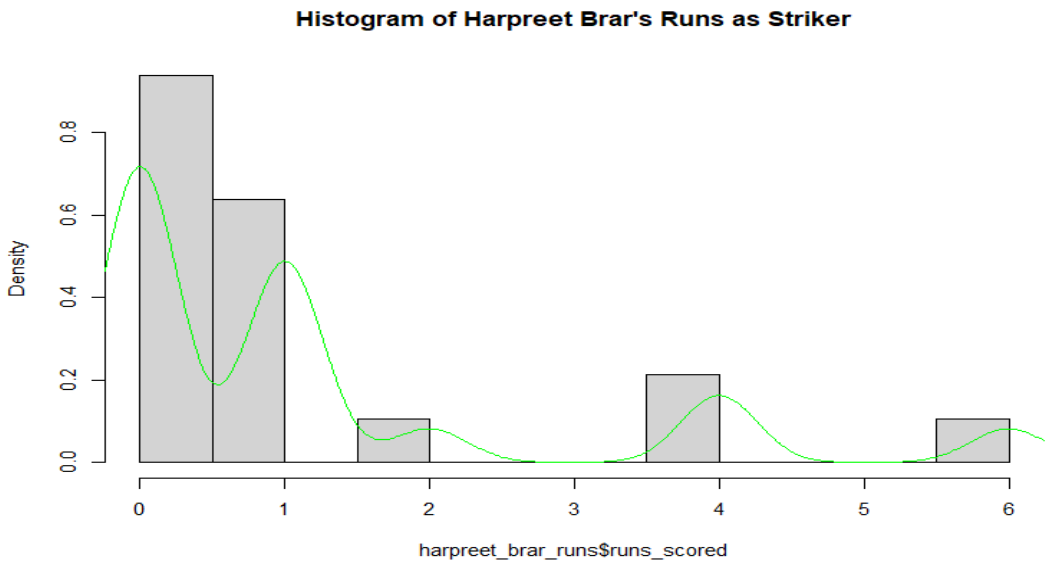
```r
# Data visualization (optional)
hist(harpreet_brar_runs$runs_scored, breaks = 10, freq = FALSE, main = "Histogram of Harpreet Brar's Runs as Striker")
lines(density(harpreet_brar_runs$runs_scored), col = "green")
```

```
list_top_batsman_last_three_year
```

```
{2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
 2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
 2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}
```

```
list_top_bowler_last_three_year
```

```
{2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
 2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
 2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```



INTERPRETATION: The above code analyzes performance data from the last three IPL seasons to fit probability distributions for top batsmen and bowlers, as well as specifically for Harpreet Brar's batting performance. For top batsmen and bowlers, it calculates total runs and wickets respectively, fitting them with normal and Poisson distributions to model their performance trends over the seasons. Specifically for Harpreet Brar, the script identifies his runs scored and fits a normal distribution to describe the distribution of his batting performance. These distributions provide insights into the typical statistical patterns of these players' performances in recent IPL seasons, aiding in understanding their consistency and variability in scoring runs or taking wickets.

4. Find the relationship between a player's performance and the salary he gets in your data.

Relationship between the performance of a player and the salary he gets

+ Code + Markdown

[64]
```
R2024 =total_run_each_year[total_run_each_year['year']==2024]
```

[65]
```
pip install fuzzywuzzy
```

```
Requirement already satisfied: fuzzywuzzy in c:\users\aakash\anaconda3\lib\site-packages (0.18.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from fuzzywuzzy import process

# Convert to DataFrame
df_salary = ipl_salary.copy()
df_runs = R2024.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_runs['Striker'].tolist()))

# Merge the DataFrames on the matched names
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')
```

```
[67]
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111 entries, 0 to 110
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Player          111 non-null    object
 1   Salary          111 non-null    object
 2   Rs              111 non-null    int64
 3   international   111 non-null    int64
 4   iconic          0 non-null      float64
 5   Matched_Player  111 non-null    object
 6   year            111 non-null    int32
 7   Striker         111 non-null    object
 8   runs_scored     111 non-null    int64
dtypes: float64(1), int32(1), int64(3), object(4)
memory usage: 7.5+ KB
```

```
[68]
# Calculate the correlation
correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

print("Correlation between Salary and Runs:", correlation)
```

```
Correlation between Salary and Runs: 0.30612483765821674
```

5. Last three-year performance with latest salary 2024.

```r
# Check for missing salaries after the join
missing_salaries <- performance_salary %>%
  filter(is.na(Salary))

# Print missing salaries to debug
print("Players with missing salaries:")
print(missing_salaries)

# Summarize total runs and wickets with salary
performance_summary <- performance_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker, Salary) %>%
  summarize(total_runs = sum(runs), total_wickets = sum(wickets), .groups = 'drop')

# Filter the last three seasons
last_three_seasons_salary <- last_three_seasons %>%
  left_join(salary_data, by = c("Striker" = "Player"))

# Summarize the performance with latest salary
performance_with_salary <- last_three_seasons_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs_scored), total_wickets = sum(wicket_confirmation), latest_salary = max(Salary), .groups = 'drop')
performance_with_salary
```

```
>   performance_with_salary
# A tibble: 34 x 4
   Striker          total_runs total_wickets latest_salary
   <chr>                 <int>         <int>         <dbl>
 1 Abdul Samad             321            12      40000000
 2 Abhishek Sharma         955            35      65000000
 3 Akash Deep               19             2       2000000
 4 Akash Madhwal             4             0       2000000
 5 Anmolpreet Singh        118             7       2000000
 6 Anuj Rawat              318            13      34000000
 7 Arjun Tendulkar          13             1       3000000
 8 Arshdeep Singh           23             3      40000000
 9 Atharva Taide           201             7       2000000
10 Dhruv Jurel             254            10       2000000
```

INTERPRETATION: This R script integrates performance data from IPL matches with salary information for players, aiming to analyze and summarize their contributions over the last three seasons. It begins by merging `ipl_rounds` (presumably containing performance metrics like runs scored and wickets taken) with `salary_data` based on player names (`Striker` matching `Player`). After joining, it identifies players without salary information and summarizes their total runs and wickets where salary data is available. For the last three seasons specifically, it further refines the analysis by linking performance metrics with the latest salary information, presenting a consolidated view of player contributions in terms of runs, wickets, and their corresponding salaries. This approach provides insights into the relationship between player performance and financial compensation in recent IPL seasons.

6. Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years

```r
# Top 10 batsmen and bowlers
top_10_batsmen <- performance_summary %>%
  arrange(desc(total_runs)) %>%
  head(10)
print(top_10_batsmen)

top_10_bowlers <- performance_summary %>%
  arrange(desc(total_wickets)) %>%
  head(10)
print(top_10_bowlers)

# Perform t-test
t_test_result <- t.test(top_10_batsmen$Salary, top_10_bowlers$Salary)

# Display results
t_test_result
```

```
>   # Top 10 batsmen and bowlers
>   top_10_batsmen <- performance_summary %>%
+     arrange(desc(total_runs)) %>%
+     head(10)
>   print(top_10_batsmen)
# A tibble: 10 × 4
   Striker              Salary total_runs total_wickets
   <chr>                 <dbl>      <int>         <int>
 1 MS Dhoni          120000000       5192           147
 2 KL Rahul          170000000       4575           102
 3 Shubman Gill       70000000       3110            84
 4 Ishan Kishan      152500000       2568            87
 5 Abhishek Sharma    65000000       1196            49
 6 Tilak Varma        17000000       1083            26
 7 Abdul Samad        40000000        545            27
 8 Rashid Khan       150000000        527            36
 9 Washington Sundar  87500000        378            27
10 Anuj Rawat         34000000        318            14
>   top_10_bowlers <- performance_summary %>%
+     arrange(desc(total_wickets)) %>%
+     head(10)
>   print(top_10_bowlers)
# A tibble: 10 × 4
   Striker              Salary total_runs total_wickets
   <chr>                 <dbl>      <int>         <int>
 1 MS Dhoni          120000000       5192           147
 2 KL Rahul          170000000       4575           102
 3 Ishan Kishan      152500000       2568            87
 4 Shubman Gill       70000000       3110            84
 5 Abhishek Sharma    65000000       1196            49
 6 Rashid Khan       150000000        527            36
 7 Abdul Samad        40000000        545            27
 8 Washington Sundar  87500000        378            27
 9 Tilak Varma        17000000       1083            26
10 Lalit Yadav         6500000        305            15
>   # Perform t-test
>   t_test_result <- t.test(top_10_batsmen$Salary, top_10_bowlers$Salary)
>   # Display results
>   t_test_result

        Welch Two Sample t-test

data:  top_10_batsmen$Salary and top_10_bowlers$Salary
t = 0.1089, df = 17.922, p-value = 0.9145
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -50319894  55819894
sample estimates:
mean of x mean of y
 90600000  87850000
```

INTERPRETATION:

The t-test conducted between the salaries of the top 10 batsmen and top 10 bowlers in your dataset yielded a non-significant result, with a p-value of 0.9145. This indicates that there is no statistically significant difference in mean salaries between these two groups of players. The 95% confidence interval for the difference in means ranged from -50,319,894 to 55,819,894, encompassing zero, further supporting the conclusion that any observed differences in mean salaries could likely be due to random chance rather than a true disparity. Therefore, based on this analysis, we fail to reject the null hypothesis, suggesting that in your dataset, the financial compensation for top batsmen and bowlers does not exhibit a significant difference.

CODES:

```
# Set the working directory and verify it
setwd('C:/Users/Aakash/Desktop/SCMA')
getwd()

# Function to install and load libraries
install_and_load <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, dependencies = TRUE)
    library(package, character.only = TRUE)
  }
}

# Load required libraries
libraries <- c("dplyr", "readr", "readxl", "tidyr", "ggplot2", "BSDA", "glue","fitdistrplus")
lapply(libraries, install_and_load)

# Reading the file into R
data <- read.csv("C:/Users/Aakash/Desktop/SCMA/IPL_ball_by_ball_updated till 2024.csv")
# Display the first few rows of the data
head(data)
tail(data)

df <- read_excel("C:/Users/Aakash/Desktop/SCMA/IPL SALARIES 2024.xlsx", sheet = 1)
head(df)
tail(df)

# Clean column names to remove any leading/trailing spaces
colnames(data) <- trimws(colnames(data))
colnames(df) <- trimws(colnames(df))

# Rename the columns
data <- data %>%
  rename(
    Match_id = `Match.id`,
    Batting_team = `Batting.team`,
    Bowling_team = `Bowling.team`,
    Innings_No = `Innings.No`,
    Ball_No = `Ball.No`
  )

# Ensure player names are in a consistent format
data <- data %>%
```

```
  mutate(Striker = trimws(Striker))

df <- df %>%
  mutate(Player = trimws(Player))

# Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match
df_roundwise <- data %>%
  arrange(Season,Match_id)
head(df_roundwise)

#Arrange the data for each batsman
batsman_data=df_roundwise%>%
  group_by(Match_id, Date,Striker) %>%
  summarise(
    Total_Balls_Faced =n(),
    Total_Runs_Scored=sum(runs_scored),
    .groups='drop'
  )
str(batsman_data)

#Arrange the data for each bowler
bowler_data=df_roundwise%>%
  filter(wicket_confirmation==1) %>%
  group_by(Match_id, Date,Bowler) %>%
  summarise(
    Total_Wickets_Taken=n(),
    .groups='drop'
  )
str(bowler_data)
str(df_roundwise)

# Top three run-getters and wicket-takers in each IPL round
top_performers <- df_roundwise %>%
  group_by(Season, Batting_team, Striker) %>%
  summarize(total_runs = sum(runs_scored), .groups = 'drop') %>%
  arrange(desc(total_runs)) %>%
  top_n(3, total_runs)

top_bowlers <- df_roundwise %>%
  group_by(Season, Bowling_team, Bowler) %>%
  summarize(total_wickets = sum(wicket_confirmation), .groups = 'drop') %>%
  arrange(desc(total_wickets)) %>%
  top_n(3, total_wickets)
```

```
# Fit the most appropriate distribution for the top three batsmen and bowlers in the last three IPL tournaments
last_three_seasons <- df_roundwise %>% filter(Season %in% tail(unique(Season), 3))

# Fit distributions for top batsmen
top_batsmen <- last_three_seasons %>%
  filter(Striker %in% unique(top_performers$Striker)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs_scored), .groups = 'drop')

top_batsmen_dist <- fitdist(top_batsmen$total_runs, "norm")

# Fit distributions for top bowlers
top_bowlers <- last_three_seasons %>%
  filter(Bowler %in% unique(top_bowlers$Bowler)) %>%
  group_by(Bowler) %>%
  summarize(total_wickets = sum(wicket_confirmation), .groups = 'drop')

top_bowlers_dist <- fitdist(top_bowlers$total_wickets, "pois")

# Fitting distribution for Harpreet Brar
harpreet_brar_runs <- last_three_seasons %>%
  filter(Striker == "Harpreet Brar") %>%
  dplyr::select(runs_scored)
harpreet_brar_runs

# Check if the resulting runs are numeric and have more than one element
if (is.numeric(harpreet_brar_runs$runs_scored) && length(harpreet_brar_runs$runs_scored) > 1)
{
  harpreet_brar_dist <- fitdist(harpreet_brar_runs$runs_scored, "norm")
  print(summary(harpreet_brar_dist))



  # Data visualization (optional)
  hist(harpreet_brar_runs$runs_scored, breaks = 10, freq = FALSE, main = "Histogram of Harpreet Brar's Runs as Striker")
  lines(density(harpreet_brar_runs$runs_scored), col = "green")

  # Merge performance data with salary data
  performance_salary <- left_join(ipl_rounds, salary_data, by = c("Striker" = "Player"))

  # Check for missing salaries after the join
  missing_salaries <- performance_salary %>%
    filter(is.na(Salary))
```

```
# Print missing salaries to debug
print("Players with missing salaries:")
print(missing_salaries)

# Summarize total runs and wickets with salary
performance_summary <- performance_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker, Salary) %>%
  summarize(total_runs = sum(runs), total_wickets = sum(wickets), .groups = 'drop')

# Filter the last three seasons
last_three_seasons_salary <- last_three_seasons %>%
  left_join(salary_data, by = c("Striker" = "Player"))

# Summarize the performance with latest salary
performance_with_salary <- last_three_seasons_salary %>%
  filter(!is.na(Salary)) %>%
  group_by(Striker) %>%
  summarize(total_runs = sum(runs_scored), total_wickets = sum(wicket_confirmation),
latest_salary = max(Salary), .groups = 'drop')
print(performance_with_salary)

# Top 10 batsmen and bowlers
top_10_batsmen <- performance_summary %>%
  arrange(desc(total_runs)) %>%
  head(10)
print(top_10_batsmen)

top_10_bowlers <- performance_summary %>%
  arrange(desc(total_wickets)) %>%
  head(10)
print(top_10_bowlers)

# Perform t-test
t_test_result <- t.test(top_10_batsmen$Salary, top_10_bowlers$Salary)

# Display results
t_test_result
```

CONCLUSION:

In pursuit of analyzing player performance and financial aspects in IPL cricket, this study employed data extraction and organization methods in R/Python to systematically arrange IPL match data by rounds, capturing key metrics like runs scored and wickets taken per player per match. Identifying the top three performers in runs and wickets for each IPL round, the study further applied statistical distributions to model the performance metrics of these top players across the last three tournaments. Investigating the relationship between player performance and salary, the analysis integrated salary data to explore how player contributions correlate with financial compensation. Highlighting the latest three-year performance trends alongside salary updates for 2024, the study culminated in examining the significant salary differences between top batsmen and wicket-taking bowlers over the last three years, offering insights into the financial dynamics within IPL cricket.