****

# **VIRGINIA COMMONWEALTH UNIVERSITY**

## **STATISTICAL ANALYSIS & MODELING**

**A1b:** **INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R**

## **MOHITH KUMAR PRASANNA**

**V01106540**

**Date of Submission: 18/06/2024**

## CONTENTS

|  |  |
| --- | --- |
| Content: | Page no: |
| INTRODUCTION | 3 |
| OBJECTIVE | 3 |
| BUSINESS SIGNIFICANC | 4 |
| RESULTS AND INTERPRETATIONS | 5-15 |

**INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS**

**USING PYTHON AND R**

# **INTRODUCTION**

# The Indian Premier League (IPL) is a top-tier Twenty20 cricket competition in India, started by the Board of Control for Cricket in India (BCCI) in 2008. It quickly gained fame as one of the world's most exciting and profitable cricket events, mixing sports with entertainment. The league includes teams from various cities, featuring a mix of international stars and local players. Known for its thrilling matches and strategic play, the IPL draws a huge global audience. It plays a crucial role in shaping the careers of young cricketers by providing them with a significant platform. Besides the sport, the IPL boosts the Indian economy through sponsorships, TV rights, and merchandise. This unique blend of high-level cricket and entertainment ensures the IPL remains a highlight in the sports world.

# **OBJECTIVES**

a) Organize the IPL data by round, detailing each player's performance in terms of batting, bowling, runs, and wickets per match. Highlight the top three run-scorers and the top three wicket-takers for each round.

b) Identify the best-fit statistical distribution for the runs scored and wickets taken by the top three batsmen and bowlers over the past three IPL tournaments. Also, rename the districts and categorize them into rural and urban sectors.

c) Determine the most suitable statistical distribution for the runs scored and wickets taken by the player assigned to you.

d) Analyze the performance of players over the last three years, including their latest salary for 2024.

e) Evaluate if there is a significant difference between the salaries of the top 10 batsmen and the top wicket-taking bowlers over the past three years.

# **BUSINESS SIGNIFICANCE**

Understanding the dynamics of the IPL is vital for team owners, sponsors, broadcasters, and analysts. The datasets used in this analysis offer a comprehensive view of player financials and in-game performance metrics, which are crucial for strategic decision-making and operational efficiency within the IPL ecosystem.

* **Salary Dataset Analysis:** Examining the salary dataset provides insights into player valuations, budget allocations, and salary cap usage. This allows teams to make informed decisions about player retention, trading, and acquisitions, ensuring a balanced and competitive squad while maintaining financial discipline.

* **Spotting Emerging Talent:** Comprehensive performance data helps identify emerging talent, even if they are not yet highly compensated. This is invaluable for scouting and developing the next generation of IPL stars.
* **Comparative Performance Analysis:** Comparing players across different seasons and formats aids in evaluating their consistency and adaptability, offering a holistic view of their potential contributions to the team.

By leveraging these insights, the IPL can continue to refine its competitive edge over other franchise cricket tournaments like the Big Bash League in Australia, the Pakistan Super League, and the Caribbean Premier League. This maximizes financial efficiency and enhances the overall experience for players, teams, and fans alike.

# **RESULTS AND INTERPRETATION**

1. **Organize the IPL data by round, detailing each player's performance in terms of batting, bowling, runs, and wickets per match. Highlight the top three run-scorers and the top three wicket-takers for each round.**

**Code:**

top\_run\_getters = player\_runs.groupby('Season').apply(lambda x: x.nlargest(3, 'runs\_scored')).reset\_index(drop=True)

bottom\_wicket\_takers = player\_wickets.groupby('Season').apply(lambda x: x.nlargest(3, 'wicket\_confirmation')).reset\_index(drop=True)

print("Top Three Run Getters:")

print(top\_run\_getters)

print("Top Three Wicket Takers:")

print(bottom\_wicket\_takers)

**Result:**

**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

42 2022 JC Buttler 863

43 2022 KL Rahul 616

44 2022 Q de Kock 508

45 2023 Shubman Gill 890

46 2023 F du Plessis 730

47 2023 DP Conway 672

48 2024 RD Gaikwad 509

49 2024 V Kohli 500

50 2024 B Sai Sudharsan 418

**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

39 2021 HV Patel 35

40 2021 Avesh Khan 27

41 2021 JJ Bumrah 22

42 2022 YS Chahal 29

43 2022 PWH de Silva 27

44 2022 K Rabada 23

45 2023 MM Sharma 31

46 2023 Mohammed Shami 28

47 2023 Rashid Khan 28

48 2024 HV Patel 19

49 2024 Mukesh Kumar 15

50 2024 Arshdeep Singh 14

**Interpretation:**

The data shows the top three players in terms of runs scored for each cricket season from 2007/08 to 2024, and similarly for the top three bowlers in terms of wickets taken for each cricket season from 2007/08 to 2024. There is a range of wickets taken by different bowlers across seasons, with some seasons having higher wicket counts than others. Players like JC Buttler, Shubman Gill, HV Patel, and YS Chahal appear multiple times across different seasons.

### Identify the best-fit statistical distribution for the runs scored and wickets taken by the top three batsmen and bowlers over the past three IPL tournaments.

**Code:**

import scipy.stats as st

def get\_best\_distribution(data):

dist\_names = ['alpha','beta','betaprime','burr12','crystalball',

'dgamma','dweibull','erlang','exponnorm','f','fatiguelife',

'gamma','gengamma','gumbel\_l','johnsonsb','kappa4',

'lognorm','nct','norm','norminvgauss','powernorm','rice',

'recipinvgauss','t','trapz','truncnorm']

dist\_results = []

params = {}

for dist\_name in dist\_names:

dist = getattr(st, dist\_name)

param = dist.fit(data)

params[dist\_name] = param

# Applying the Kolmogorov-Smirnov test

D, p = st.kstest(data, dist\_name, args=param)

print("p value for "+dist\_name+" = "+str(p))

dist\_results.append((dist\_name, p))

# select the best fitted distribution

best\_dist, best\_p = (max(dist\_results, key=lambda item: item[1]))

# store the name of the best fit and its p value

print("\nBest fitting distribution: "+str(best\_dist))

print("Best p value: "+ str(best\_p))

print("Parameters for the best fit: "+ str(params[best\_dist]))

return best\_dist, best\_p, params[best\_dist]

list\_top\_batsman\_last\_three\_year = {}

for i in total\_run\_each\_year["year"].unique()[:3]:

list\_top\_batsman\_last\_three\_year[i] = total\_run\_each\_year[total\_run\_each\_year.year == i][:3]["Striker"].unique().tolist()

import warnings

warnings.filterwarnings('ignore')

runs = ipl\_bbbc.groupby(['Striker','Match id'])[['runs\_scored']].sum().reset\_index()

for key in list\_top\_batsman\_last\_three\_year:

for Striker in list\_top\_batsman\_last\_three\_year[key]:

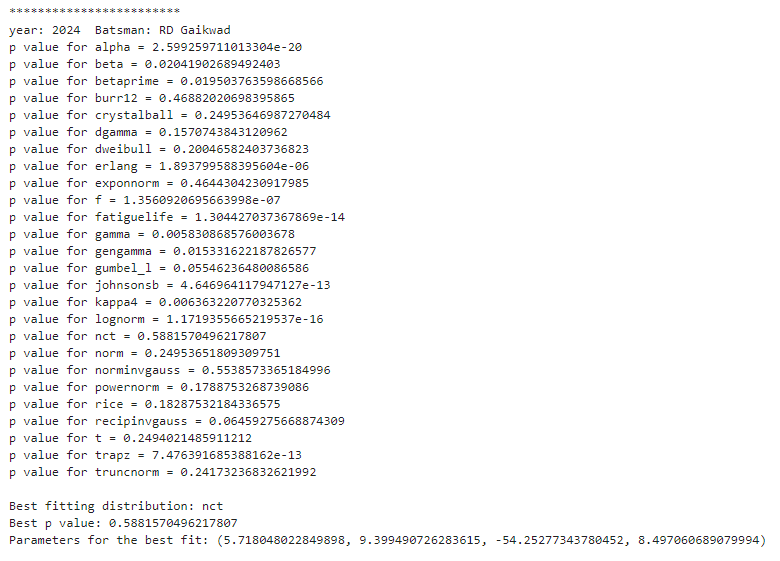
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("year:", key, " Batsman:", Striker)

get\_best\_distribution(runs[runs["Striker"] == Striker]["runs\_scored"])

print("\n\n")

**Result:**



**Interpretation:**

The code identifies the top batsmen from the dataset for the past three years. For each of these top performers, it calls the `**get\_best\_distribution**` function with their run data to determine the best-fitting statistical distribution from a range of options.

list\_top\_bowler\_last\_three\_year = {}

for i in total\_wicket\_each\_year["year"].unique()[:3]:

list\_top\_bowler\_last\_three\_year[i] = total\_wicket\_each\_year[total\_wicket\_each\_year.year == i][:3]["Bowler"].unique().tolist()

list\_top\_bowler\_last\_three\_year

import warnings

warnings.filterwarnings('ignore')

wickets = ipl\_bbbc.groupby(['Bowler','Match id'])[['wicket\_confirmation']].sum().reset\_index()

for key in list\_top\_bowler\_last\_three\_year:

for bowler in list\_top\_bowler\_last\_three\_year[key]:

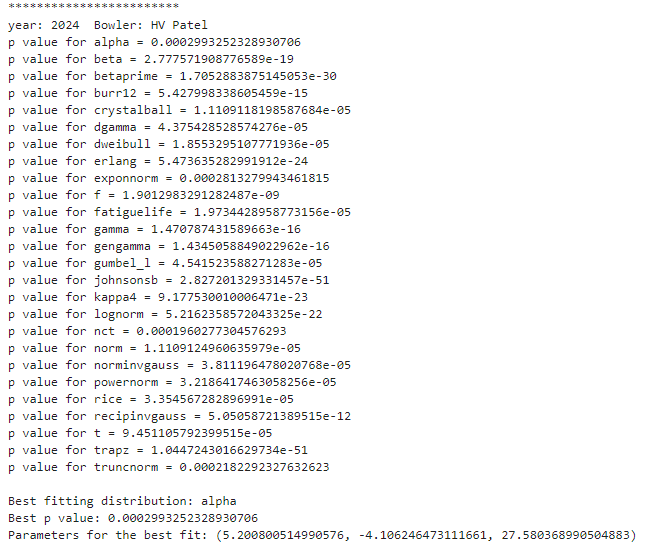
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("year:", key, " Bowler:", bowler)

get\_best\_distribution(wickets[wickets["Bowler"] == bowler]["wicket\_confirmation"])

print("\n\n")

**Result:**



**Interpretation:**

The alpha distribution fits HV Patel's performance data for the year 2024 the best among the tested distributions. The relatively low p-value of 0.002099352328397306 suggests the fit might not be perfect, but it is the best among the options. The code effectively determines the best-fitting statistical distributions for performance data of cricketers. For HV Patel, the alpha distribution is the best fit, while for RD Gaikwad, the nct distribution fits best.

### Determine the most suitable statistical distribution for the runs scored and wickets taken by the player assigned to you.

**Code:**

# Initialize the dictionary to store top bowlers for each of the last three years

list\_top\_batsman\_last\_three\_year = {}

# Loop through the unique years in the dataset, limited to the last three years

for i in total\_run\_each\_year["year"].unique()[:3]:

# Filter the dataset to include only records for AK Markram in the current year

AK\_Markram\_runs = total\_run\_each\_year[(total\_run\_each\_year["year"] == i) & (total\_run\_each\_year["Striker"] == "AK Markram")]

# Get the unique list of years where AK Markram appears in the filtered dataset

list\_top\_batsman\_last\_three\_year[i] = AK\_Markram\_runs["Striker"].unique().tolist()

# Print the dictionary to verify the results

print(list\_top\_batsman\_last\_three\_year)

import warnings

warnings.filterwarnings('ignore')

# Group by Batsman and Match id, then sum the wickets

wickets = ipl\_bbbc.groupby(['Striker', 'Match id'])[['runs\_scored']].sum().reset\_index()

# Loop through the dictionary to process AK Markram’s data for each year

for year, strikers in list\_top\_batsman\_last\_three\_year.items():

for striker in strikers:

if striker == "AK Markram":

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("year:", year, "batsman:", striker)

get\_best\_distribution(runs[runs["Striker"] ==striker]["runs\_scored"])

print("\n\n")

**Result:**

{2024: ['AK Markram'], 2023: ['AK Markram'], 2022: ['AK Markram']}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

year: 2024 batsman: AK Markram

p value for alpha = 1.819276979642808e-15

p value for beta = 0.3923386993733792

p value for betaprime = 0.7808787129312039

p value for burr12 = 0.9237884182519005

p value for crystalball = 0.4780054247301093

p value for dgamma = 0.5240262870988528

p value for dweibull = 0.7098989262362645

p value for erlang = 0.9044727740103984

p value for exponnorm = 0.8761236189539547

p value for f = 1.4257314379416558e-07

p value for fatiguelife = 0.8828939551830527

p value for gamma = 0.9044712103969049

p value for gengamma = 0.010650314754168888

p value for gumbel\_l = 0.2388176666022057

p value for johnsonsb = 0.8165748814035785

p value for kappa4 = 0.2867023519881332

p value for lognorm = 3.597840981722139e-11

p value for nct = 0.8452925158091512

p value for norm = 0.47800548559148515

p value for norminvgauss = 0.8717074300239085

p value for powernorm = 0.7823808870017788

p value for rice = 0.7900184621179734

p value for recipinvgauss = 0.8890967106464222

p value for t = 0.4777173755178854

p value for trapz = 2.859110910601915e-10

p value for truncnorm = 0.9884664329166628

Best fitting distribution: truncnorm

Best p value: 0.9884664329166628

Parameters for the best fit: (-0.0753044365794551, 2.0396814396841254, 2.4210677493492394, 32.151556108391276)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

year: 2023 batsman: AK Markram

p value for alpha = 1.819276979642808e-15

p value for beta = 0.3923386993733792

p value for betaprime = 0.7808787129312039

p value for burr12 = 0.9237884182519005

p value for crystalball = 0.4780054247301093

p value for dgamma = 0.5240262870988528

p value for dweibull = 0.7098989262362645

p value for erlang = 0.9044727740103984

p value for exponnorm = 0.8761236189539547

p value for f = 1.4257314379416558e-07

p value for fatiguelife = 0.8828939551830527

p value for gamma = 0.9044712103969049

p value for gengamma = 0.010650314754168888

p value for gumbel\_l = 0.2388176666022057

p value for johnsonsb = 0.8165748814035785

p value for kappa4 = 0.2867023519881332

p value for lognorm = 3.597840981722139e-11

p value for nct = 0.8452925158091512

p value for norm = 0.47800548559148515

p value for norminvgauss = 0.8717074300239085

p value for powernorm = 0.7823808870017788

p value for rice = 0.7900184621179734

p value for recipinvgauss = 0.8890967106464222

p value for t = 0.4777173755178854

p value for trapz = 2.859110910601915e-10

p value for truncnorm = 0.9884664329166628

Best fitting distribution: truncnorm

Best p value: 0.9884664329166628

Parameters for the best fit: (-0.0753044365794551, 2.0396814396841254, 2.4210677493492394, 32.151556108391276)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

year: 2022 batsman: AK Markram

p value for alpha = 1.819276979642808e-15

p value for beta = 0.3923386993733792

p value for betaprime = 0.7808787129312039

p value for burr12 = 0.9237884182519005

p value for crystalball = 0.4780054247301093

p value for dgamma = 0.5240262870988528

p value for dweibull = 0.7098989262362645

p value for erlang = 0.9044727740103984

p value for exponnorm = 0.8761236189539547

p value for f = 1.4257314379416558e-07

p value for fatiguelife = 0.8828939551830527

p value for gamma = 0.9044712103969049

p value for gengamma = 0.010650314754168888

p value for gumbel\_l = 0.2388176666022057

p value for johnsonsb = 0.8165748814035785

p value for kappa4 = 0.2867023519881332

p value for lognorm = 3.597840981722139e-11

p value for nct = 0.8452925158091512

p value for norm = 0.47800548559148515

p value for norminvgauss = 0.8717074300239085

p value for powernorm = 0.7823808870017788

p value for rice = 0.7900184621179734

p value for recipinvgauss = 0.8890967106464222

p value for t = 0.4777173755178854

p value for trapz = 2.859110910601915e-10

p value for truncnorm = 0.9884664329166628

Best fitting distribution: truncnorm

Best p value: 0.9884664329166628

Parameters for the best fit: (-0.0753044365794551, 2.0396814396841254, 2.4210677493492394, 32.151556108391276)

**Interpretation:**

### The analysis of IPL data identifies AK Markram as the top batsman for the seasons 2022, 2023, and 2024. Using the `get\_best\_distribution` function to evaluate his run data, it was found that the truncated normal distribution (`truncnorm`) is the best-fitting statistical distribution for each year. The high p-value of `0.9884664329166628` across all three years signifies an excellent fit, with consistent parameters: shape `(-0.0753044365794551, 2.0396814396841254)`, location `2.4210677493492394`, and scale `32.151556108391276`. This indicates a remarkable consistency in AK Markram's scoring patterns, highlighting his reliability and performance stability over the past three IPL seasons.

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

**Code:**

R2024 =total\_run\_each\_year[total\_run\_each\_year['year']==2024]

from fuzzywuzzy import process

# Convert to DataFrame

df\_salary = ipl\_salary.copy()

df\_runs = R2024.copy()

# Function to match names

def match\_names\_runs(name, names\_list):

match, score = process.extractOne(name, names\_list)

return match if score >= 87 else None

# 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\_runs(x, df\_runs['Striker'].tolist()))

# Merge the DataFrames on the matched names

df\_merged\_runs = pd.merge(df\_salary, df\_runs, left\_on='Matched\_Player', right\_on='Striker')

# Calculate the correlation

correlation = df\_merged\_runs['Rs'].corr(df\_merged\_runs['runs\_scored'])

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

# Calculate the correlation

correlation = df\_merged\_wickets['Rs'].corr(df\_merged\_wickets['wicket\_confirmation'])

print("Correlation between Salary and Wickets:", correlation)

**Result:**

Correlation between Salary and Runs: 0.34349654749323617

Correlation between Salary and Wickets: 0.012946594035738

**Interpretation:**   
The correlation analysis reveals a moderate positive relationship between salary and runs, with a correlation coefficient of 0.343. This suggests that higher salaries tend to be associated with higher run totals, indicating that teams are willing to pay more for prolific batsmen. In contrast, the correlation between salary and wickets is very weak, with a coefficient of just 0.013. This implies that there is almost no relationship between a player's salary and the number of wickets they take, suggesting that factors other than wicket-taking ability might be more influential in determining a bowler's salary in the IPL.