

VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A1b: Statistical Analysis of IPL Player data

Deepti
V01107603

Date of Submission: 18/06/2023

CONTENTS

Content:	Page no:
INTRODUCTION	3
OBJECTIVE	3
BUSINESS SIGNIFICANC	3-4
RESULTS AND INTERPRETATIONS	4-9
CODES	10-14

Statistical Analysis, and Player salary and Performance co-relation

INTRODUCTION

This study focuses on analyzing data related to overall player performance in the Indian Premier League (IPL), employing statistical methods to understand and quantify various performance metrics and their correlation with player salaries. By leveraging a comprehensive dataset, we aim to derive meaningful insights into the performance trends and financial aspects of the league.

Our objectives include cleaning and preprocessing the dataset, identifying and addressing missing values, and standardizing player names and performance metrics. Through exploratory data analysis, we summarize key performance indicators and examine regional and player-wise variations. Furthermore, we apply statistical techniques to test the significance of correlations between player salaries and their on-field performance.

By systematically analyzing this data, we provide valuable insights that can inform team management and stakeholders, enabling them to make data-driven decisions. The findings from this study contribute to a deeper understanding of player performance dynamics and financial strategies within the context of the IPL.

OBJECTIVES

- a) 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.
- b) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.
- c) Find the relationship between a player's performance and the salary he gets in your data.
- d) Last three-year performance with latest salary 2024
- e) Significant Difference Between the Salaries of the Top 10 Batsmen and Top Wicket-Taking Bowlers Over the Last Three Years
- f) * Use the data sets [data "Cricket_data.csv" & "Salary 2024.csv"] t.

BUSINESS SIGNIFICANCE

The focus of this study on Analysis of IPL Player Performance for the ipl data holds significant implications for businesses and policymakers. By understanding player performance we can understand as team owners on how much money to be given to a player the study provides valuable insights for team owners, player performance analysis, season retention and budget for players. Through data cleaning, distribution, and significance testing, the findings facilitate informed decision-making, fostering equitable development, Player planning and mapping of players for IPL Teams in the future.

RESULTS AND INTERPRETATION

A) Player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.

#Identifying the top three run getters and top three wicket takers in easch ipl season

Code and Result:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

os.chdir('/Users/kirthanshaker/Desktop/SCMA 631 Data Files ')

ipl_bbb = pd.read_csv('/Users/kirthanshaker/Desktop/SCMA 631 Data Files
/IPL_ball_by_ball_updated till 2024.csv',low_memory=False)

ipl_salary = pd.read_excel('/Users/kirthanshaker/Desktop/SCMA 631 Data Files /IPL SALARIES
2024.xlsx')

ipl_salary.head(2)

grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_scored': sum,
'wicket_confirmation':sum}).reset_index()

player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
player_wickets = grouped_data.groupby(['Season',
'Bowler'])['wicket_confirmation'].sum().reset_index()

player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)

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
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
28	2017	G Gambhir	498

29	2017	S Dhawan	479
30	2018	KS Williamson	735
31	2018	RR Pant	684
32	2018	KL Rahul	659
33	2019	DA Warner	692
34	2019	KL Rahul	593
35	2019	Q de Kock	529
36	2020/21	KL Rahul	676
37	2020/21	S Dhawan	618
38	2020/21	DA Warner	548
39	2021	RD Gaikwad	635
40	2021	F du Plessis	633
41	2021	KL Rahul	626
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
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
28	2017	JD Unadkat	27
29	2017	JJ Bumrah	23
30	2018	AJ Tye	28
31	2018	S Kaul	24
32	2018	Rashid Khan	23
33	2019	K Rabada	29
34	2019	Imran Tahir	26
35	2019	JJ Bumrah	23
36	2020/21	K Raba	32
37	2020/21	JJ Bumrah	30
38	2020/21	TA Boult	26
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 table outlines the top three run-getters and wicket-takers for each IPL season from 2007/08 to 2024. It highlights consistent performers such as Chris Gayle, who led the charts multiple times with remarkable scores like 733 runs in both 2012 and 2013, and Virat Kohli, who had an extraordinary season in 2016 with 973 runs. For bowlers, notable mentions include DJ Bravo

with 34 wickets in 2013 and Harshal Patel with 35 wickets in 2021. The data demonstrates the dominance of certain players over multiple seasons and underscores key performances that significantly impacted their teams' success in various IPL editions.

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

```
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]

total_run_each_year = ipl_bbbc.groupby(["year", "Striker"])[ "runs_scored"].sum().reset_index()

total_run_each_year.sort_values(["year", "runs_scored"], ascending=False, inplace=True)
print(total_run_each_year)

```

Result:

	year	Striker	runs_scored
2549	2024	RD Gaikwad	509
2589	2024	V Kohli	500
2470	2024	B Sai Sudharsan	418
2502	2024	KL Rahul	406
2555	2024	RR Pant	398
...
58	2008	L Balaji	0
66	2008	M Muralitharan	0
75	2008	MM Patel	0
107	2008	S Sreesanth	0
136	2008	U Kaul	0

[2598 rows x 3 columns]

Top Three Batsmen For The Last three Years

Code and Result

```

In [18]: list_top_batsman_last_three_year

Out[18]: {2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
          2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
          2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}

```

Top Three Bowler For The Last three Years

```
In [22]: 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()
list_top_bowler_last_three_year

Out[22]: {2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

Interpretation: For the top three run-scorers in the last three IPL seasons, the runs scored followed a Normal distribution with a mean of approximately 611 runs and a standard deviation of 145 runs. This indicates that the performance of the top batsmen was fairly consistent with some variability.

For the top three wicket-takers, the data best fit a Poisson distribution with an average rate of 24 wickets per season. This suggests that the top bowlers' performance in terms of wickets taken was relatively predictable and followed a specific rate.

These distributions help in understanding the typical performance ranges and variability for top players in recent IPL seasons.

Top Three batsmen for the last three years

In the last three IPL seasons (2022, 2023, 2024),

Top Three bowler for the last three years

For the top three bowlers in the same period, players like MM Sharma and YS Chahal were dominant, with Sharma taking 31 wickets in 2023 and Chahal taking 29 in 2022.

CODES

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

os.chdir('/Users/kirthanshaker/Desktop/SCMA 631 Data Files ')

ipl_bbb = pd.read_csv('/Users/kirthanshaker/Desktop/SCMA 631 Data Files
/IPL_ball_by_ball_updated till 2024.csv',low_memory=False)

ipl_salary = pd.read_excel('/Users/kirthanshaker/Desktop/SCMA 631 Data Files /IPL SALARIES
2024.xlsx')

ipl_salary.head(2)

grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg({'runs_scored':
sum, 'wicket_confirmation':sum}).reset_index()

player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
player_wickets = grouped_data.groupby(['Season',
'Bowler'])['wicket_confirmation'].sum().reset_index()

player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)
```

```

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)

ipl_year_id = pd.DataFrame(columns=["id", "year"])
ipl_year_id["id"] = ipl_bbb["Match id"]
ipl_year_id["year"] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year

#create a copy of ipl_bbbc dataframe
ipl_bbbc= ipl_bbb.copy()

ipl_bbbc['year'] = pd.to_datetime(ipl_bbb["Date"], dayfirst=True).dt.year

ipl_bbbc[["Match id", "year", "runs_scored", "wicket_confirmation", "Bowler", "Striker"]].head()

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]

total_run_each_year = ipl_bbbc.groupby(["year", "Striker"])[["runs_scored"]].sum().reset_index()

total_run_each_year.sort_values(["year", "runs_scored"], ascending=False, inplace=True)
print(total_run_each_year)

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()

list_top_batsman_last_three_year

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")

total_wicket_each_year = ipl_bbbc.groupby(["year",
"Bowler"])[["wicket_confirmation"].sum().reset_index()

total_wicket_each_year.sort_values(["year", "wicket_confirmation"], ascending=False,
inplace=True)
print(total_wicket_each_year)

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")

# RelationShip between the performance of the player and the salary he gets

```

```

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

#pip install fuzzywuzzy

pip install fuzzywuzzy

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')

df_merged.info()

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

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

# Filter ball to ball data for the last three years

```



```
# Assuming you have a column 'year' representing the year of each match
last_three_years_data = total_wicket_each_year[total_wicket_each_year['2019'] >= 2022]

# Aggregate performance data
# Assuming you have columns like 'player', 'runs_scored', 'wickets_taken' in your ball to ball data
performance_aggregated = last_three_years_data.groupby('player').agg({
    'runs_scored': 'sum',
    'wickets_taken': 'sum'
}).reset_index()

# Merge with salary data
merged_data = pd.merge(performance_aggregated, salary_data, on='player', how='left')

# Add latest salary (2024)
# Assuming you have a column 'salary_2024' in your salary data
latest_salary_data = merged_data[['player', 'salary_2024']]
latest_salary_data.columns = ['player', 'latest_salary_2024']
```

