VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A1b: ANALYSIS OF IPL PLAYER PERFORMANCE AND SALARY USING PYTHON AND R

RIDDHI RUNGTA
V01107488

Date of Submission: 18/06/2024

CONTENTS

Analysis of IPL players performance and salary using R and Python

INTRODUCTION

The Indian Premier League (IPL) is a professional Twenty20 cricket league in India contested by eight teams representing different cities. It has garnered immense popularity due to its high-octane matches, star-studded line-ups, and significant financial stakes. It is one of the most celebrated cricket leagues globally, known for its blend of high-octane matches, international player participation, and significant commercial investment. This report utilizes two key datasets to provide comprehensive insights into the IPL. The first dataset, "IPL_ball_by_ball_updated till 2024.csv," details the ball-by-ball actions of IPL matches from its inception till 2024. The second dataset, "IPL SALARIES 2024.xlsx," provides information on the salaries of IPL players for the 2024 season. Additionally, the project will explore the correlation between player performance and their salaries, providing insights into the financial aspects of player valuation.

OBJECTIVES

a) Extract the files in R/Python
b) To 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.
c) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.
d) Find the relationship between a player's performance and the salary he gets in your data.

BUSINESS SIGNIFICANCE

Understanding the dynamics of the IPL is crucial for several stakeholders, including team owners, sponsors, broadcasters, and analysts. Here is how these datasets can be leveraged:

1) Such data provides insights into optimizing player acquisition strategies and budget allocation, ensuring a balanced team composition without overspending on underperforming players.
2) Analyzing ball-by-ball data helps in understanding the effectiveness of different strategies under various match conditions. Detailed performance metrics enable the assessment of individual player contributions, identifying key performers and potential areas of improvement.
3) Insights from player performance can inform salary negotiations, ensuring that teams invest

wisely in players who provide the best return on investment.

4) It provides a comprehensive understanding of performance trends and financial dynamics within the IPL, enriching the analytical discourse around the league.
5) Detailed match data allows for the creation of engaging content that can attract and retain viewership. Sponsors can assess the visibility and impact of their investments by correlating match events with sponsorship slots.
6) By understanding peak moments in matches, broadcasters can strategically place advertisements to maximize revenue.

In conclusion, these datasets not only enhance the understanding of game dynamics but also provide significant business value by driving better decision-making across performance, financial planning, marketing, and broadcasting in the IPL ecosystem

RESULTS AND INTERPRETATION

a) Extract the files in R/Python

I performed this objective in both R and python. The codes are given below:

For R,

```
# Set working directory
> setwd("C:/Users/RIDDHI/OneDrive/Desktop/Bootcamp/SCMA/SCMA A1b")
>
> # Read data
> ipl_bbb <- read.csv('IPL_ball_by_ball_updated till 2024.csv', stringsAsFactors = FALSE)
> ipl_salary <- read_excel('IPL SALARIES 2024.xlsx')
```

For Python,

```
In [2]: os.chdir('C:\\Users\\RIDDHI\\OneDrive\\Desktop\\Bootcamp\\SCMA\\SCMA A1b')
In [3]: ipl_bbb = pd.read_csv('IPL_ball_by_ball_updated till 2024.csv',low_memory=False)
In [4]: ipl_salary = pd.read_excel('IPL SALARIES 2024.xlsx')
```

b) 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.

Code:

The following code below is executed to make a subset of the bigger data taking only six variables in consideration.

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

```
In [7]: player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
        player_wickets = grouped_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()

In [8]: player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)
```

Out[8]:

	Season	Striker	runs_scored
2423	2023	Shubman Gill	890
2313	2023	F du Plessis	730
2311	2023	DP Conway	672
2433	2023	V Kohli	639
2443	2023	YBK Jaiswal	625
...	...	...	...
2404	2023	RP Meredith	0
2372	2023	Mohsin Khan	0
2307	2023	DG Nalkande	0
2429	2023	TU Deshpande	0
2324	2023	Harshit Rana	0

177 rows × 3 columns

```
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 Rabada	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 code and the corresponding output display the top three run-getters and top three wicket-takers for each IPL season. The top run-getters for each season demonstrate consistent performance by key players, highlighting their contributions to their respective teams. The top wicket-takers reflect the effectiveness and dominance of certain bowlers in the IPL.

The top performers in terms of runs and wickets are likely to be among the highest-paid players due to their significant contributions to their teams' successes. Teams may focus on retaining these top performers to maintain competitive advantages in future seasons. Consistent performers like Virat Kohli and Bhuvneshwar Kumar demonstrate their importance to their teams, while emerging talents like Shubman Gill highlight the

evolving landscape of the league. Understanding these trends can help teams make strategic decisions regarding player retention, salary negotiations, and overall team management.

c) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the last 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]
```

For batsman

```
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
```

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']}
```

Code:

```
In [19]: 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:

****
year: 2024 Batsman: RD Gaikwad
p value for alpha = 2.599259711013304e-20
p value for beta = 0.02041902689492403
p value for betaprime = 0.019503763598668566
p value for burr12 = 0.46882020698395865
p value for crystalball = 0.24953646987270484
p value for dgamma = 0.1570743843120962
p value for dweibull = 0.20046582403736823
p value for erlang = 1.893799588395604e-06
p value for exponnorm = 0.4644304230917985
p value for f = 1.3560920695663998e-07
p value for fatiguelife = 1.304427037367869e-14
p value for gamma = 0.005830868576003678
p value for gengamma = 0.015331622187826577
p value for gumbel_l = 0.05546236480086586
p value for johnsonsb = 4.646964117947127e-13
p value for kappa4 = 0.006363220770325362
p value for lognorm = 1.1719355665219537e-16
p value for nct = 0.5881570496217807
p value for norm = 0.24953651809309751
p value for norminvgauss = 0.5538573365184996
p value for powernorm = 0.1788753268739086
p value for rice = 0.18287532184336575
p value for recipinvgauss = 0.06459275668874309
p value for t = 0.2494021485911212
p value for trapz = 7.476391685388162e-13
p value for truncnorm = 0.24173236832621992

Best fitting distribution: nct
Best p value: 0.5881570496217807
Parameters for the best fit: (5.718048022849898, 9.399490726283615, -54.25277343780452, 8.497060689079994)

****
year: 2024 Batsman: V Kohli
p value for alpha = 0.15371704349416937
p value for beta = 0.7807091136830002
p value for betaprime = 0.15634788776461095
p value for burr12 = 0.2201385645469427
p value for crystalball = 0.0013439120565839657
p value for dgamma = 0.00010919434981556638
p value for dweibull = 0.00012533056352014233
p value for erlang = 1.7690285330312436e-06
p value for exponnorm = 0.19376408619173924
p value for f = 2.67581083049327e-28
p value for fatiguelife = 0.11580928039819094
p value for gamma = 0.00878530144799014
p value for gengamma = 0.12789719547406364
p value for gumbel_l = 9.544555237684654e-09
p value for johnsonsb = 0.6600676697983927
p value for kappa4 = 7.270307243307106e-18
p value for lognorm = 6.635544190553261e-64
p value for nct = 0.1460773085917223
p value for norm = 0.0013439146566564463
p value for norminvgauss = 0.16537494306738054
p value for powernorm = 0.001959224898154651
p value for rice = 0.0019496833019799402
p value for recipinvgauss = 0.08835236633247623
p value for t = 0.001870132740059356
p value for trapz = 3.7326843413039495e-73
p value for truncnorm = 0.08872852288813304

Best fitting distribution: beta
Best p value: 0.7807091136830002
Parameters for the best fit: (0.816277299300862, 2.3391761669196907, -3.0251144495756596e-31, 13
0.79371484721577)

****
year: 2024 Batsman: B Sai Sudharsan
p value for alpha = 0.9519530946513592
p value for beta = 0.2800374272685796
p value for betaprime = 0.7272275700648236
p value for burr12 = 0.0341373038396523
p value for crystalball = 0.835174953613428
p value for dgamma = 0.9003132708081405
p value for dweibull = 0.8965770306228721
p value for erlang = 0.2710277691398305
p value for exponnorm = 0.8246418777999891
p value for f = 0.9743698554720728
p value for fatiguelife = 0.8259440652110397

p value for gamma = 0.004088711345359375
p value for gengamma = 0.029688848326628436
p value for gumbel_l = 0.391243924609637
p value for johnsonsb = 0.6775536294207896
p value for kappa4 = 0.042731569281991066
p value for lognorm = 0.9006026891568572
p value for nct = 0.9627359408368513
p value for norm = 0.8351750214399875
p value for norminvgauss = 0.8696382419018381
p value for powernorm = 0.837790705015941
p value for rice = 0.8419161308192361
p value for recipinvgauss = 0.7846020832234206
p value for t = 0.8945403499225023
p value for trapz = 4.962305050994183e-07
p value for truncnorm = 0.8112138570439418

Best fitting distribution: f
Best p value: 0.9743698554720728
Parameters for the best fit: (7.230079711691059, 94.80999484543659, -0.46870159044880233, 39.842
02109781083)

****
year: 2023 Batsman: Shubman Gill
p value for alpha = 0.19370998562525277
p value for beta = 0.35556757767764935
p value for betaprime = 0.3320890781747333
p value for burr12 = 0.17538338566759049
p value for crystalball = 0.04047310237062718
p value for dgamma = 0.004654508243065125
p value for dweibull = 0.011388953681874758
p value for erlang = 0.10415431199992453
p value for exponnorm = 0.4076479842986127
p value for f = 1.211921514554867e-19
p value for fatiguelife = 0.220391503090979
p value for gamma = 0.019326052677511196
p value for gengamma = 0.15830394669705905
p value for gumbel_l = 0.00016365306017313027
p value for johnsonsb = 0.6214006077216168
p value for kappa4 = 8.537718673686839e-12
p value for lognorm = 3.0444374367609376e-26
p value for nct = 0.10819705795130274
p value for norm = 0.0404730725346123
p value for norminvgauss = 0.2256809493002514
p value for powernorm = 0.008933578018931798
p value for rice = 0.009231529839363262
p value for recipinvgauss = 0.25695076184687626
p value for t = 0.06288757117419963
p value for trapz = 7.559368072972744e-39
p value for truncnorm = 0.03322263046428764

Best fitting distribution: johnsonsb
Best p value: 0.6214006077216168
Parameters for the best fit: (1.127462972555547, 0.7082040622620326, -1.0785135120261573, 140.57
94643798755)

****
year: 2023 Batsman: F du Plessis
p value for alpha = 2.6514415564811303e-46
p value for beta = 0.5913252599657466
p value for betaprime = 0.21607006903997872
p value for burr12 = 1.4054517820032704e-09
p value for crystalball = 0.17738239944644352
p value for dgamma = 0.0192505709952403
p value for dweibull = 0.11610399857369136
p value for erlang = 1.5300500072467267e-05
p value for exponnorm = 0.029960734734523542
p value for f = 2.3763783336197345e-18
p value for fatiguelife = 0.4484315774329326
p value for gamma = 2.658122267546294e-07
p value for gengamma = 0.02408727588734938
p value for gumbel_l = 0.0014475463566163693
p value for johnsonsb = 0.18738807412325909
p value for kappa4 = 7.855215717595119e-07
p value for lognorm = 7.76777670084355e-36
p value for nct = 0.3074928968583557
p value for norm = 0.1773824188508326
p value for norminvgauss = 0.5294908193576565
p value for powernorm = 0.10747661134694209
p value for rice = 0.10596246415943456
p value for recipinvgauss = 0.25232880325823326
p value for t = 0.17742481659951348
p value for trapz = 2.2917131806009114e-31
p value for truncnorm = 0.4976264771179164

Best fitting distribution: beta
Best p value: 0.5913252599657466
Parameters for the best fit: (0.964930449377772, 2.3654747855916978, -2.4979006319546827e-31, 11
0.45316400426368)

****
year: 2023 Batsman: DP Conway
p value for alpha = 0.24224437379078456
p value for beta = 0.9335739280635689
p value for betaprime = 0.5939028036769798
p value for burr12 = 0.03168649038236593
p value for crystalball = 0.5919833978299178

p value for dgamma = 0.659050680685497
p value for dweibull = 0.47709033274534696
p value for erlang = 0.5856582107400496
p value for exponnorm = 0.5919442519144027
p value for f = 0.03191068848461143
p value for fatiguelife = 2.4470875845519328e-05
p value for gamma = 0.5772798774478445
p value for gengamma = 0.010638224653254702
p value for gumbel_l = 0.6434008985606366
p value for johnsonsb = 0.0010884744390042833
p value for kappa4 = 0.39160448071756937
p value for lognorm = 3.1507840694396127e-06
p value for nct = 0.5925999092825844
p value for norm = 0.5919834368439854
p value for norminvgauss = 0.5925748844419921
p value for powernorm = 0.45248629955798125
p value for rice = 0.45768623194758373
p value for recipinvgauss = 0.031005955700377452
p value for t = 0.5919821236916709
p value for trapz = 0.002896838839657856
p value for truncnorm = 0.2820881279467663

Best fitting distribution: beta
Best p value: 0.9335739280635689
Parameters for the best fit: (0.6250316512826838, 0.6786342050356671, -3.4741633120498916, 95.47416331204991)

****
year: 2022 Batsman: JC Buttler
p value for alpha = 3.235109657468491e-34
p value for beta = 0.33455794816369444
p value for betaprime = 0.0040250475185371615
p value for burr12 = 0.7069656630104211
p value for crystalball = 0.004608459861307201
p value for dgamma = 0.00604199317470544
p value for dweibull = 0.0028430680547544274
p value for erlang = 0.0018449508774974754
p value for exponnorm = 0.7137955109895673
p value for f = 3.9553917967759444e-17
p value for fatiguelife = 0.3817917882201278
p value for gamma = 0.0007081454329525005
p value for gengamma = 0.30583328083418904
p value for gumbel_l = 0.00010416429669054019
p value for johnsonsb = 0.5217216451703999
p value for kappa4 = 1.0421737381705364e-12
p value for lognorm = 5.057168420293 5185e-28
p value for nct = 0.45209196275779084
p value for norm = 0.004608461486487414
p value for norminvgauss = 0.4852525149516915

p value for powernorm = 0.004689395332742374
p value for rice = 0.004972139278293097
p value for recipinvgauss = 0.2745923469661903
p value for t = 0.007226707680554001
p value for trapz = 8.531784262849386e-37
p value for truncnorm = 0.03894315379655533

Best fitting distribution: exponnorm
Best p value: 0.7137955109895673
Parameters for the best fit: (3054.885295608514, -0.031805252610631926, 0.01119090499814962)

****

year: 2022 Batsman: KL Rahul
p value for alpha = 3.439822697019343e-50
p value for beta = 0.3005191042009908
p value for betaprime = 0.3083252430394988
p value for burr12 = 0.46187713102710526
p value for crystalball = 0.02169172684247167
p value for dgamma = 0.06770258558041642
p value for dweibull = 0.10186919378179626
p value for erlang = 0.5713953642722212
p value for exponnorm = 0.21607213755074883
p value for f = 3.271576641222778e-23
p value for fatiguelife = 0.4121975839714658
p value for gamma = 0.5713982751559553
p value for gengamma = 0.16010152392031385
p value for gumbel_l = 0.001680677455102142
p value for johnsonsb = 0.9402453631468569
p value for kappa4 = 1.3895397566735892e-07
p value for lognorm = 9.796218603186654e-32
p value for nct = 0.20349727522799965
p value for norm = 0.021691727067097988
p value for norminvgauss = 0.3817037858973431
p value for powernorm = 0.026645565499311186
p value for rice = 0.027062729391134077
p value for recipinvgauss = 0.4426895366659932
p value for t = 0.021694088191051786
p value for trapz = 1.8532732379092856e-35
p value for truncnorm = 0.6753901355264902

Best fitting distribution: johnsonsb
Best p value: 0.9402453631468569
Parameters for the best fit: (0.9331207997896902, 0.7776389044559282, -2.345202857963142, 143.08
33194837059)

****

year: 2022 Batsman: Q de Kock

p value for alpha = 0.22421213312317778
p value for beta = 0.2878667203270271
p value for betaprime = 0.05740280491001126
p value for burr12 = 0.4931279667432148
p value for crystalball = 0.05846912701914453
p value for dgamma = 0.0014560083713105465
p value for dweibull = 0.010478670398011536
p value for erlang = 0.08677035591445126
p value for exponnorm = 0.43726373790797446
p value for f = 4.2346585152678845e-12
p value for fatiguelife = 0.12498847851930417
p value for gamma = 0.027350558506526124
p value for gengamma = 0.09268925126776417
p value for gumbel_l = 9.485045980257123e-06
p value for johnsonsb = 0.3450941869097196
p value for kappa4 = 3.832745782875419e-18
p value for lognorm = 2.3658846096591403e-28
p value for nct = 0.2843302460638113
p value for norm = 0.05846911111218267
p value for norminvgauss = 0.2268711891858607
p value for powernorm = 0.03382371687362962
p value for rice = 0.03349090516310227
p value for recipinvgauss = 0.1073883725317536
p value for t = 0.041656498991066715
p value for trapz = 3.947363741930107e-50
p value for truncnorm = 0.08860764609496041

Best fitting distribution: burr12
Best p value: 0.4931279667432148
Parameters for the best fit: (590926023.7998527, 0.05483081555360233, -969803927.022117, 969803927.160071)

For bowler:

Code:

```
In [23]: 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
```

Result:

```
Out[23]: {2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
          2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
          2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

Code:

```
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:

year: 2024 Bowler: HV Patel
p value for alpha = 0.0002993252328930706
p value for beta = 2.777571908776589e-19
p value for betaprime = 1.7052883875145053e-30
p value for burr12 = 5.427998338605459e-15
p value for crystalball = 1.1109118198587684e-05
p value for dgamma = 4.375428528574276e-05
p value for dweibull = 1.8553295107771936e-05
p value for erlang = 5.473635282991912e-24
p value for exponnorm = 0.0002813279943461815
p value for f = 1.9012983291282487e-09
p value for fatiguelife = 1.9734428958773156e-05
p value for gamma = 1.470787431589663e-16
p value for gengamma = 1.4345058849022962e-16
p value for gumbel_l = 4.541523588271283e-05
p value for johnsonsb = 2.827201329331457e-51
p value for kappa4 = 9.177530010006471e-23
p value for lognorm = 5.2162358572043325e-22
p value for nct = 0.0001960277304576293
p value for norm = 1.1109124960635979e-05
p value for norminvgauss = 3.811196478020768e-05
p value for powernorm = 3.2186417463058256e-05
p value for rice = 3.354567282896991e-05
p value for recipinvgauss = 5.05058721389515e-12
p value for t = 9.451105792399515e-05
p value for trapz = 1.0447243016629734e-51
p value for truncnorm = 0.0002182292327632623

Best fitting distribution: alpha
Best p value: 0.0002993252328930706
Parameters for the best fit: (5.200800514990576, -4.106246473111661, 27.580368990504883)

****
year: 2024 Bowler: Mukesh Kumar
p value for alpha = 0.6028771589628603
p value for beta = 0.011954014965331 66
p value for betaprime = 0.0010598932359472402

p value for burr12 = 0.13577547952316893
p value for crystalball = 0.2874602836058906
p value for dgamma = 0.31965148068347327
p value for dweibull = 0.34346643238289587
p value for erlang = 1.0115032724485677e-06
p value for exponnorm = 0.5154597105302977
p value for f = 0.11745949856748239
p value for fatiguelife = 0.30877430134651207
p value for gamma = 0.009841759821405782
p value for gengamma = 0.07933719921899463
p value for gumbel_l = 0.25997636144422587
p value for johnsonsb = 0.08788077953204243
p value for kappa4 = 0.058739565059041765
p value for lognorm = 0.00048729251059009826
p value for nct = 0.5480580718802854
p value for norm = 0.28746007995258704
p value for norminvgauss = 0.3895684674359623
p value for powernorm = 0.39511432172869
p value for rice = 0.3950169895189477
p value for recipinvgauss = 0.025198651172109288
p value for t = 0.2874574742538948
p value for trapz = 9.722628535925783e-06
p value for truncnorm = 0.2598105493516787

Best fitting distribution: alpha
Best p value: 0.6028771589628603
Parameters for the best fit: (6.113363581345144, -5.245777123804531, 39.57745263632695)

****
year: 2024 Bowler: Arshdeep Singh
p value for alpha = 0.002547644307209551
p value for beta = 3.7725133611153275e-15
p value for betaprime = 5.062381659741898e-22
p value for burr12 = 4.603956720503075e-14
p value for crystalball = 0.0002501762149918564
p value for dgamma = 0.00028566200697101806
p value for dweibull = 0.0016211491850549598
p value for erlang = 2.269289539862191e-12
p value for exponnorm = 0.0019097947631203649
p value for f = 0.000227258408802241
p value for fatiguelife = 2.169103029961132e-15
p value for gamma = 6.618486511618167e-29
p value for gengamma = 5.948936850168967e-23
p value for gumbel_l = 0.00026864389982599567
p value for johnsonsb = 5.472387372640376e-24
p value for kappa4 = 8.181970339328129e-12
p value for lognorm = 1.9909678840157557e-12
p value for nct = 0.0014257070102449143
p value for norm = 0.00025017539197677184

p value for norminvgauss = 0.0001290021448063343
p value for powernorm = 0.00047137775975730436
p value for rice = 0.00047472774494963083
p value for recipinvgauss = 1.9623061606588953e-10
p value for t = 0.004473243416688644
p value for trapz = 1.1911079182772876e-29
p value for truncnorm = 0.00034221379785853717

Best fitting distribution: t
Best p value: 0.004473243416688644
Parameters for the best fit: (4.822497644715119, 1.1162819391895469, 0.9153269129308039)

****

year: 2023 Bowler: MM Sharma
p value for alpha = 5.261792307574885e-09
p value for beta = 3.369903415982389e-18
p value for betaprime = 3.4236065288569164e-34
p value for burr12 = 7.707563359968149e-27
p value for crystalball = 5.614290141391915e-05
p value for dgamma = 1.0498635614441156e-05
p value for dweibull = 2.4126502201215078e-05
p value for erlang = 2.203151538560566e-17
p value for exponnorm = 7.116980583029457e-10
p value for f = 6.394862208673673e-10
p value for fatiguelife = 1.3371709463319658e-24
p value for gamma = 2.599880000032353e-21
p value for gengamma = 9.811276806787944e-14
p value for gumbel_l = 3.5245319536008275e-05
p value for johnsonsb = 2.4461951672713995e-40
p value for kappa4 = 1.804941215806713e-17
p value for lognorm = 1.7804559351656542e-19
p value for nct = 6.513780696080299e-05
p value for norm = 5.614083233477072e-05
p value for norminvgauss = 2.385888242491267e-11
p value for powernorm = 3.7448415090755237e-05
p value for rice = 3.8846082842387146e-05
p value for recipinvgauss = 1.932872667384276e-17
p value for t = 0.00012008020713636171
p value for trapz = 9.04818074400941e-47
p value for truncnorm = 6.39486602704708e-10

Best fitting distribution: t
Best p value: 0.00012008020713636171
Parameters for the best fit: (29.05846643939152, 1.2878076424619436, 1.197404368883093)

****

year: 2023 Bowler: Mohammed Shami

p value for alpha = 0.0005609846480252995
p value for beta = 8.949702621553806e-16
p value for betaprime = 1.0457228098472159e-27
p value for burr12 = 3.809437306589196e-09
p value for crystalball = 8.97379813361614e-06
p value for dgamma = 1.3065638273544516e-11
p value for dweibull = 1.0406851960138218e-05
p value for erlang = 8.670599832745995e-28
p value for exponnorm = 0.00047630665162716083
p value for f = 2.404756281608377e-07
p value for fatiguelife = 7.5219130194197114e-06
p value for gamma = 5.248327144461885e-42
p value for gengamma = 4.371554773381843e-42
p value for gumbel_l = 2.275582226089825e-06
p value for johnsonsb = 8.40193769288202e-62
p value for kappa4 = 5.440679375551408e-12
p value for lognorm = 8.538407160860825e-23
p value for nct = 0.0003740512893746841
p value for norm = 8.973880770320002e-06
p value for norminvgauss = 3.3178705246034226e-05
p value for powernorm = 0.00011849751955444802
p value for rice = 0.00011833002960228116
p value for recipinvgauss = 1.957916752902072e-07
p value for t = 8.972846375529713e-06
p value for trapz = 1.8983891174798298e-38
p value for truncnorm = 2.539236515610462e-06

Best fitting distribution: alpha
Best p value: 0.0005609846480252995
Parameters for the best fit: (6.734843933630203, -5.500744811228249, 44.826257131250145)

****
year: 2023 Bowler: Rashid Khan
p value for alpha = 1.4259399000489275e-06
p value for beta = 8.8954046965209e-27
p value for betaprime = 3.407105814148136e-65
p value for burr12 = 2.558767583325104 7e-18
p value for crystalball = 2.99049361738744e-09
p value for dgamma = 6.928485900596178e-10
p value for dweibull = 6.928168431614811e-10
p value for erlang = 1.052461604472364e-41
p value for exponnorm = 7.720335528170629e-07
p value for f = 4.940207066298226e-10
p value for fatiguelife = 1.4667845015790087e-07
p value for gamma = 3.120866167200452e-31
p value for gengamma = 3.3780076161228415e-35
p value for gumbel_l = 7.911140658362043e-09
p value for johnsonsb = 6.659510229977693e-18
p value for kappa4 = 6.390225516379688e-22

p value for lognorm = 6.677625232671758e-27
p value for nct = 8.389699838025371e-07
p value for norm = 2.9905103094429466e-09
p value for norminvgauss = 1.9883690059384983e-07
p value for powernorm = 5.69320390726131e-08
p value for rice = 6.008338811339319e-08
p value for recipinvgauss = 1.0204427503324627e-07
p value for t = 4.1495986291836466e-08
p value for trapz = 4.291139733358819e-55
p value for truncnorm = 3.085454927439526 4e-07

Best fitting distribution: alpha
Best p value: 1.4259399000489275e-06
Parameters for the best fit: (5.783058438949956, -4.20986029264825, 30.878991656277478)

****
year: 2022 Bowler: YS Chahal
p value for alpha = 1.1180274965710719e-05
p value for beta = 1.0295677049868252e-44
p value for betaprime = 6.005755537239427e-40
p value for burr12 = 1.7979353447013811e-12
p value for crystalball = 5.1232708024114544e-08
p value for dgamma = 4.012289620255995e-08
p value for dweibull = 1.3446088982977968e-07
p value for erlang = 2.6044501249608127e-33
p value for exponnorm = 9.70188325365383e-06
p value for f = 4.3760412135414686e-11
p value for fatiguelife = 1.0610357499785987e-07
p value for gamma = 3.2021687139045712e-55
p value for gengamma = 4.0264602677437785e-26
p value for gumbel_l = 8.01003405037582e-08
p value for johnsonsb = 9.127045203599366e-44
p value for kappa4 = 5.8742872003226356e-27
p value for lognorm = 1.2869567438882943e-32
p value for nct = 5.296213377700368e-06
p value for norm = 5.1235707238843755e-08
p value for norminvgauss = 3.3808295582037935e-07
p value for powernorm = 1.021178511514112e-06
p value for rice = 1.0373024397997343e-06
p value for recipinvgauss = 1.53711078374615e-21
p value for t = 1.1782910213333637e-07
p value for trapz = 1.8568421933146807e-70
p value for truncnorm = 1.609035128404315e-07

Best fitting distribution: alpha
Best p value: 1.1180274965710719e-05
Parameters for the best fit: (6.054854001673274, -4.898293043793716, 36.81747298117385)

****
year: 2022 Bowler: PWH de Silva
p value for alpha = 0.20501605213397378
p value for beta = 6.089293734595811e-08
p value for betaprime = 3.597368592551267e-07
p value for burr12 = 2.7078633279028545e-05
p value for crystalball = 0.12578198773774585
p value for dgamma = 0.04130328255260218
p value for dweibull = 0.08384976427162982
p value for erlang = 0.0002485071992361352
p value for exponnorm = 0.3076424973571079
p value for f = 0.006583510714380791
p value for fatiguelife = 0.0879596136953581
p value for gamma = 8.727963496024317e-05
p value for gengamma = 0.00519063892676308
p value for gumbel_l = 0.014493692496563626
p value for johnsonsb = 2.0634443260981352e-05
p value for kappa4 = 1.8620061578617215e-06
p value for lognorm = 5.934676005942877e-06
p value for nct = 0.18287627001224627
p value for norm = 0.1257824642902543
p value for norminvgauss = 0.10918449199764368
p value for powernorm = 0.1963520712744381
p value for rice = 0.1985929094578025
p value for recipinvgauss = 4.423190500679613e-05
p value for t = 0.1973319936827771
p value for trapz = 1.9360347216700493e-15
p value for truncnorm = 0.10632743012364088

Best fitting distribution: exponnorm
Best p value: 0.3076424973571079
Parameters for the best fit: (1.5651879172672551, 0.40254290759385924, 0.6274498232929551)

****
year: 2022 Bowler: K Rabada
p value for alpha = 0.01766606343280419
p value for beta = 4.443616547466671e-12
p value for betaprime = 4.702163459968348e-17
p value for burr12 = 1.0217952890763225e-11
p value for crystalball = 0.003016635703159909
p value for dgamma = 0.0040395395676823265
p value for dweibull = 0.004897361468685357
p value for erlang = 6.666902843060855e-10
p value for exponnorm = 0.012447792991604367
p value for f = 6.634692021556237e-06
p value for fatiguelife = 0.011517197590084738
p value for gamma = 1.032396146883282e-12
p value for gengamma = 2.6816733980980167e-12

p value for gumbel_l = 0.00045795960689101544
p value for johnsonsb = 3.123503411674573e-12
p value for kappa4 = 2.016542974865221e-05
p value for lognorm = 2.015341179637063e-18
p value for nct = 0.01550593593647065
p value for norm = 0.003016639761756701
p value for norminvgauss = 0.01159359005102878
p value for powernorm = 0.012612430707674482
p value for rice = 0.012664345659931242
p value for recipinvgauss = 0.011156908993034342
p value for t = 0.0030166123509550724
p value for trapz = 2.238131859007279e-22
p value for truncnorm = 0.007005335434665971

Best fitting distribution: alpha
Best p value: 0.01766606343280419
Parameters for the best fit: (8.172744476082507, -7.746415964015842, 75.18055369544504)

Interpretation:

The above code – we have taken both the top 3 wicket keepers and top 3 strikers for the last 3 season to understa nd the distribution and arrive at the best fit distribution. The code provides a detailed analysis of the performance of top batsmen over the last three IPL seasons. By fitting the best statistical distributions to their runs scored, it of fers valuable insights into their scoring patterns. These insights can be utilized for various strategic decisions, inc luding player retention, salary negotiations, and performance predictions. The approach demonstrates the power of combining statistical analysis with sports data to drive informed decision-making in cricket. The code effectiv ely identifies the best-fitting statistical distributions for cricketers' performance data. For HV Patel, the alpha dist ribution is the best fit, while the nct distribution fits RD Gaikwad's performance data best. The provided paramet ers can be used for further analysis or simulation of the players'

For the player allotted to me – Sandeep Sharma

As a bowler

Code:

```
# Initialize the dictionary to store top bowlers for each of the last three years
list_top_bowler_last_three_year = {}

# Loop through the unique years in the dataset, limited to the last three years
for i in total_wicket_each_year["year"].unique()[:3]:
    # Filter the dataset to include only records for Sandeep Sharma in the current year
    sandeep_sharma_data = total_wicket_each_year[(total_wicket_each_year["year"] == i) &
(total_wicket_each_year["Bowler"] == "Sandeep Sharma")]
    # Get the unique list of years where Sandeep Sharma appears in the filtered dataset
    list_top_bowler_last_three_year[i] = sandeep_sharma_data["Bowler"].unique().tolist()

# Print the dictionary to verify the results
print(list_top_bowler_last_three_year)
```

Result:

{2024: ['Sandeep Sharma'], 2023: ['Sandeep Sharma'], 2022: ['Sandeep Sharma']}

```
import warnings
warnings.filterwarnings('ignore')

# Group by Bowler and Match id, then sum the wickets
wickets = ipl_bbbc.groupby(['Bowler', 'Match id'])[['wicket_confirmation']].sum().reset_index()

# Loop through the dictionary to process Sandeep Sharma's data for each year
for year, bowlers in list_top_bowler_last_three_year.items():
    for bowler in bowlers:
        if bowler == "Sandeep Sharma":
            print("**************************")
            print("year:", year, " Bowler:", bowler)
            get_best_distribution(wickets[wickets["Bowler"] == bowler]["wicket_confirmation"])
            print("\n\n")
```

Result:

```
**************************
year: 2024  Bowler: Sandeep Sharma
p value for alpha = 6.5369692572030414e-06
p value for beta = 6.65975644956026e-29
p value for betaprime = 6.962692895520194e-16
p value for burr12 = 5.594165162493497e-32
p value for crystalball = 1.0473396641122168e-07
p value for dgamma = 1.1225298407882938e-06
p value for dweibull = 2.0005645419188965e-05
p value for erlang = 1.4923011721975657e-50
p value for exponnorm = 4.7494385951043994e-09
p value for f = 4.7422270508831054e-09
p value for fatiguelife = 4.2501815288001374e-39
p value for gamma = 3.152636586986662e-21
p value for gengamma = 6.309253671867099e-26
p value for gumbel_l = 4.0271662004426624e-08
p value for johnsonsb = 3.440791268196001e-11
p value for kappa4 = 6.369163185480884e-24
p value for lognorm = 7.433872965600071e-23
p value for nct = 1.7502449225339898e-06
p value for norm = 1.0473202013169173e-07
p value for norminvgauss = 1.342689952877046e-09
p value for powernorm = 6.742090615971264e-07
p value for rice = 6.613406435027004e-07
p value for recipinvgauss = 2.022104063095661e-24
p value for t = 2.5177085501477924e-06
p value for trapz = 2.166165460108003e-57
p value for truncnorm = 5.126126335574074e-09

Best fitting distribution: dweibull
```

Best p value: 2.0005645419188965e-05
Parameters for the best fit: (0.32753480043223115, 0.9999999999999998, 1.042224843892015)

Bowler Analysis:

For Sandeep Sharma's wicket-taking performance in 2024, the best-fitting distribution is `dweibull` with a p-value of approximately 0.00002. The `dweibull` distribution is characterized by a shape parameter of 0.3275, a location parameter of 1.0, and a scale parameter of 1.0422. Sandeep Sharma's wicket-taking performance in 2024 can be modeled well by the `dweibull` distribution. The distribution suggests a pattern where he is more likely to take wickets later in his bowling spells. For all three years, 2022, 2023 and 2024, the best fitting distribution for Sandeep Sharma's bowling data is the Weibull distribution (dweibull). The p-values associated with the Weibull distribution were 2.0005645419188965e-05. These p-values indicate the probability of obtaining the observed data under the null hypothesis that the data follow the Weibull distribution. Lower p-values suggest better agreement between the data and the distribution. The consistent selection of the Weibull distribution across both years suggests that this distribution model adequately captures the underlying statistical patterns in Sandeep Sharma's bowling performance. Also the distributional assumptions underlying the Weibull model are robust for analyzing Sandeep Sharma's bowling data.

As a batsman

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 Sandeep Sharma in the current year
    sandeep_sharma_data = total_run_each_year[(total_run_each_year["year"] == i) &
(total_run_each_year["Striker"] == "Sandeep Sharma")]
    # Get the unique list of years where Sandeep Sharma appears in the filtered dataset
    list_top_batsman_last_three_year[i] = sandeep_sharma_data["Striker"].unique().tolist()

# Print the dictionary to verify the results
print(list_top_batsman_last_three_year)
```

Result:

```
{2024: [], 2023: ['Sandeep Sharma'], 2022: []}
```

```
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 Sandeep Sharma's data for each year
for year, strikers in list_top_batsman_last_three_year.items():
    for striker in strikers:
        if striker == "Sandeep Sharma":
            print("************************")
            print("year:", year, "batsman:", striker)
            get_best_distribution(runs[runs["Striker"] ==striker]["runs_scored"])
            print("\n\n")
```

Result:

```
************************
year: 2023 batsman: Sandeep Sharma
p value for alpha = 0.055951170843091536
p value for beta = 0.008904481644169682
p value for betaprime = 0.0002976927093332574
p value for burr12 = 0.0011346923786262897
p value for crystalball = 0.0280554239804911
p value for dgamma = 0.10448827780154635
p value for dweibull = 0.1333264816674461
p value for erlang = 0.008904481644167905
p value for exponnorm = 0.009137414250432352
p value for f = 0.008904481644167905
p value for fatiguelife = 0.00025329273728291025
p value for gamma = 0.008904481644167905
p value for gengamma = 0.008904481644166906
p value for gumbel_l = 0.012157436272226319
p value for johnsonsb = 2.2550469859013985e-11
p value for kappa4 = 1.059276908420797e-06
p value for lognorm = 8.928975028622504e-05
p value for nct = 5.471081969212092e-12
p value for norm = 0.02805544552516359
p value for norminvgauss = 5.6629312330930795e-05
p value for powernorm = 0.01344363734945575
p value for rice = 0.013624283622972744
p value for recipinvgauss = 0.008904495949166868
p value for t = 0.07199142793624547
p value for trapz = 3.25163529688618e-12
p value for truncnorm = 0.00890449298970164

Best fitting distribution: dweibull
Best p value: 0.1333264816674461
Parameters for the best fit: (0.5211285186900497, 0.9999999999999999, 1.5531917620668063)
```

Interpretation:

Batsman Analysis:

For Sandeep Sharma's batting performance in 2023, the best-fitting distribution is also dweibull with a p-value of approximately 0.1333. The dweibull distribution for his batting has a shape parameter of 0.5211, a location parameter of 1.0, and a scale parameter of 1.5532. Sandeep Sharma's batting performance in 2023, though limited, shows a distribution similar to his bowling performance, indicating a possible trend in his batting innings where he may score more runs towards the later part of his innings. With respect to Sandeep Sharma's batting performance, since he is a bowler, the years, 2022 and 2024 did not have any batting data as he did not get a chance to bat. Whereas in 2023 it did exist, and the best fit distribution for the data was dweibull with a p-value of 0.1333264816674461. However this doesn't have any impact on the selection of the player as he is termed as a bowler and the statistics for the bowling data, i.e wickets taken would be the crucial factor for his selection.

d) 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]

#pip install fuzzywuzzy
Collecting fuzzywuzzy
  Obtaining dependency information for fuzzywuzzy from https://files.pythonhosted.org/packages/43/ff/74f23998ad2f93b945c0309f825be92e04e0348e062026998b5eefef4c33/fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)
Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-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 >= 82 else None  # Use a threshold score of 82

# 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()
```

Result:

```
<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
```

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

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

Result:

```
Correlation between Salary and Runs: 0.3043435086686198
```

Interpretation:

To combine player salary data with their performance data (runs scored) for the year 2024, we had to use fuzzy string matching which uses Levenshtein Distance to calculate the differences between sequences. to match player names between the two datasets. Uses the fuzzywuzzy library to match player names from the salary data to the player names in the run data. Since the accuracy of the fuzzy score or threshold limit would directly impact the correlation, I have kept a high score of 82 to ensure there are no soft matches in the merged data frame. The value of 0.3043 indicates a moderate positive correlation between salary and runs scored. While not a very strong correlation, it suggests that, generally, players who earn higher salaries tend to score more runs.

This indicates a moderate positive correlation between a player's salary and the number of runs they score. The positive correlation suggests that, in general, as a player's salary increases, their runs scored also tend to increase. However, the correlation is not strong, indicating that other factors may also

influence a player's salary or runs scored independently. Other factors such as player performance in terms of strike rate, consistency, match-winning performances, and overall contribution to the team may also play a role in determining a player's salary.