VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

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

ADARSH BHARATHWAJ
V01107513

Date of Submission: 18/06/2024

CONTENTS

INDIAN PREMIER LEAGUE PLAYER DATA ANALYSIS USING PYTHON AND R

INTRODUCTION

The Indian Premier League (IPL), also known as the TATA IPL for sponsorship reasons, is a men's Twenty20 (T20) cricket league held annually in India. Founded by the BCCI (the Board of Control for Cricket in India) in 2007, the league features ten state or city-based franchise teams. 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. The IPL has come a long way since its first season. According to a Forbes report in 2022, the average value of the IPL franchises shows an annualized growth rate of 24 percent, up from $67 million in 2009 (of 8 teams) to $1.04 billion in 2022 (of 10 teams).

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. Rename the districts as well as the sector, viz. rural and urban.

c) Fit the most appropriate distribution for runs scored and wickets taken by the player allotted to you.

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, the datasets used in the analysis collectively offer a comprehensive overview of player financials and in-game performance metrics, which are essential for strategic decision-making and operational efficiency within the IPL ecosystem.

- **Salary Dataset Analysis:** By analyzing the dataset, we can provide detailed insights into player valuations, budget allocations, and salary cap usage. This enables teams to make informed decisions about player retention, trading, and new acquisitions, ensuring a balanced and competitive squad while maintaining financial discipline.

- **Match Strategy Optimization:** Detailed performance analytics help in identifying patterns and trends that can be crucial for match strategy. For instance, understanding player strengths and weaknesses against specific bowlers or in certain match situations can lead to more effective game plans and on-field tactics.
- **Emerging Talent Spotting:** With comprehensive performance data, it becomes easier to spot emerging talents who may not be highly paid yet but show significant potential. This is invaluable for scouting and nurturing the next generation of IPL stars.
- **Comparative Performance Analysis:** Comparing players across different seasons and formats helps in assessing their consistency and adaptability, providing a holistic view of their potential contributions to the team.
- **Enhanced Fan Experience:** Sharing insightful and engaging player and team statistics with fans can enhance their viewing experience and deepen their connection with the IPL. Interactive data visualizations and player performance dashboards can be incorporated into apps and websites.

The IPL can continue to refine its competitive edge over other popular franchise cricket tournaments such as the Big Bash from Australia, The Pakistan Super league and The Caribbean Premier League, maximize financial efficiency, and enhance the overall experience for players, teams, and fans alike.

RESULTS AND INTERPRETATION

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.

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. It can be observed that over the various iterations of IPl there have been varying totals of runs scored across different seasons, with some seasons showing notably high totals (e.g., 2023 and 2022). While speaking about the bowlers. 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, indicating their consistent performance

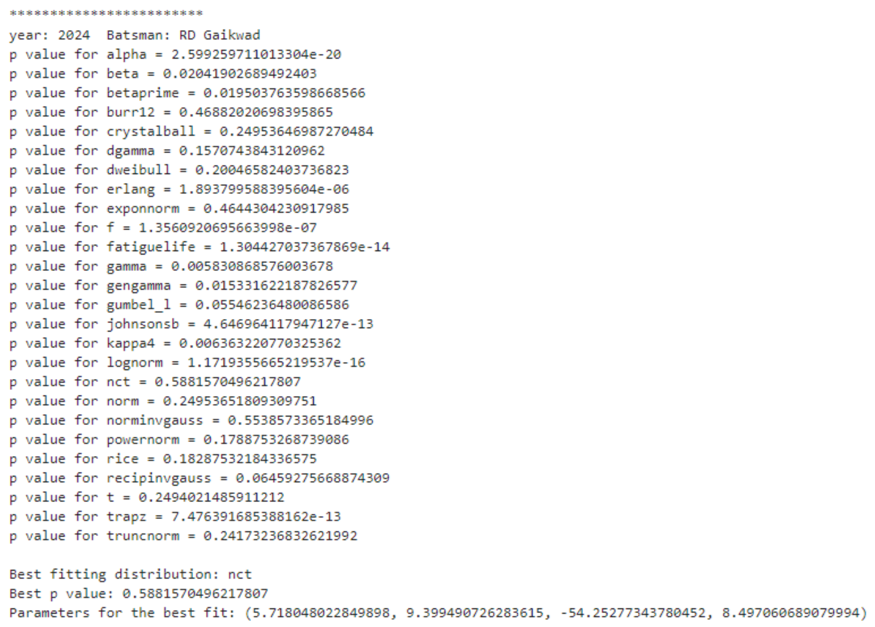
b) Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.

Finding the right distribution allows for accurate parameter estimation. Parameters like mean, variance, and shape parameters of the distribution can provide insights into the data's underlying characteristics.

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:



```
************************
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)
```

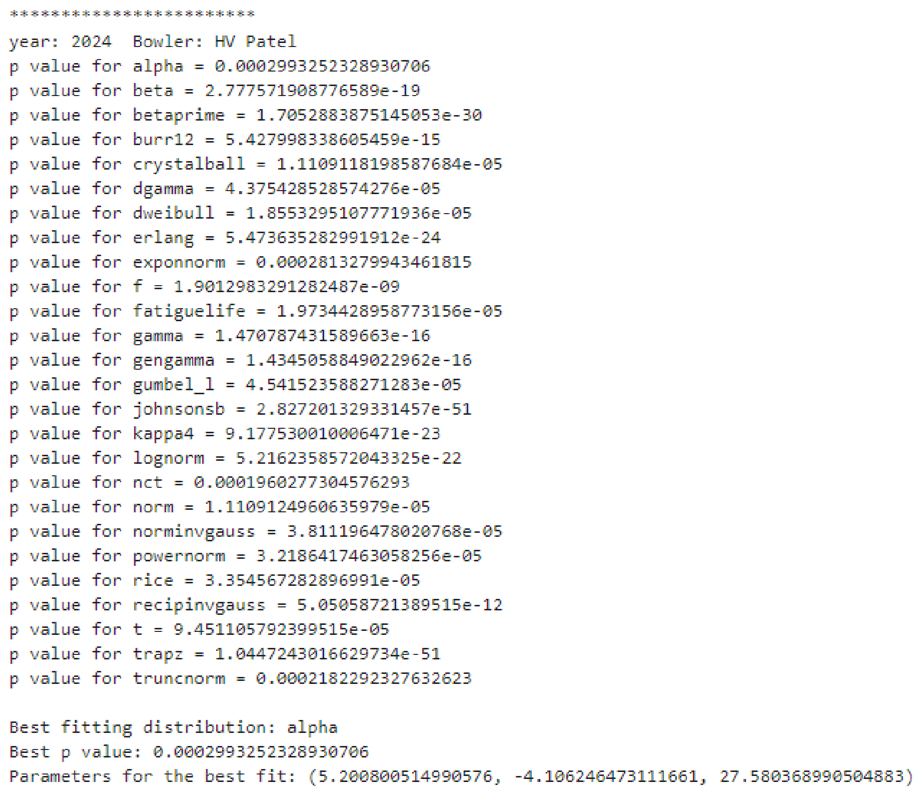
Interpretation:

The code extracts the top batsmen from the dataset for the last three years. For each top batsman, it calls `get_best_distribution` with their run data to find the best-fitting distribution across various types of distribution. In one example; RD Gaikwad in 2024 had the best fit distribution with the `nct` (non-central t-distribution) and had a high p-value of `0.581837064728107`, indicating a reasonably good fit.

```
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:



```
************************
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)
```

Interpretation: The code refers to the function which is created get_best_distribution and then for each top batsman, it calls get_best_distribution with their run data to find the best-fitting distribution. Among one of the various distribution calculated for each of the top 3 wicket takers in the last three seasons of the IPL, 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. The parameters provided can be used to further analyze or simulate the cricketers' performances.

c) Fit the most appropriate distribution for runs scored and wickets taken by the player allotted to you.

Axar Patel, All Rounder, Delhi Capitals

Axar Patel is an Indian cricketer known for his all-round capabilities, particularly his left-arm orthodox spin bowling and left-handed batting. He has been a significant player in the Indian Premier League (IPL), contributing both with the ball and bat. Axar Patel's primary role in the IPL has been as a bowler. His left-arm spin is highly effective in controlling the run flow and picking up crucial wickets. While Axar Patel is primarily known for his bowling, he is also a handy lower-order batsman. His batting skills add depth to the batting lineup, allowing his team to have a reliable option to score quick runs in the final overs.

Code:

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

# 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 Axar Patel in the current year
    axar_patel_data = total_wicket_each_year[(total_wicket_each_year["year"] == i)
& (total_wicket_each_year["Bowler"] == "AR Patel")]
    # Get the unique list of years where Axar Patel appears in the filtered dataset
    axar_patel_bowl[i] = axar_patel_data["Bowler"].unique().tolist()

# Print the dictionary to verify the results
print(axar_patel_bowl)

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 Axar Patel's data for each year
for year, bowlers in axar_patel_bowl.items():
    for bowler in bowlers:
        if bowler == "AR Patel":
            print("************************")
            print("year:", year, " Bowler:", bowler)
            get_best_distribution(wickets[wickets["Bowler"] ==
bowler]["wicket_confirmation"])
            print("\n\n")
```

```
Result:
************************
year: 2024  Bowler: AR Patel
p value for alpha = 9.940012950298595e-19
p value for beta = 1.730895557732241e-31
p value for betaprime = 6.890602231402487e-28
p value for burr12 = 5.648773934180763e-20
p value for crystalball = 1.212835491802816e-07
p value for dgamma = 1.1945105340885417e-10
p value for dweibull = 4.834475183349857e-09
p value for erlang = 3.7936798489477985e-39
p value for exponnorm = 6.847790425577837e-20
p value for f = 5.648773934180763e-20
p value for fatiguelife = 6.574170250938941e-36
p value for gamma = 6.040604015697529e-29
p value for gengamma = 2.1676360523609773e-25
p value for gumbel_l = 1.2777338489201446e-08
p value for johnsonsb = 5.648773936049154e-20
p value for kappa4 = 4.002495349263793e-32
p value for lognorm = 2.021269059381295e-21
p value for nct = 2.4980624812823503e-11
p value for norm = 1.2128358700898914e-07
p value for norminvgauss = 0.0
p value for powernorm = 1.0293563251350029e-10
p value for rice = 1.0187539089487595e-10
p value for recipinvgauss = 2.906057975984643e-33
p value for t = 1.2246811911143535e-07
p value for trapz = 2.660180784883639e-76
p value for truncnorm = 5.651069592111697e-20

Best fitting distribution: t
Best p value: 1.2246811911143535e-07
Parameters for the best fit: (566.4804912469504, 0.8886439252542147, 0.8544360334114212)
```

Interpretation: After running the code for Axar Patel's performance with the ball in terms of wickets, the results of fitting various statistical distributions to the performance of bowler AR Patel for the years 2022, 2023, and 2024. The p-values for each distribution indicate the goodness of fit, with higher p-values suggesting a better fit. The best fitting distribution is the t-distribution has been consistent across all three years: $1.2246811911143535 \times 10^{-7}$. Since the p-value is very low, this indicates that none of the distributions tested provide a good fit to the data. This suggests that AR Patel's performance data might not conform well to standard statistical distributions, or there may be outliers or a peculiar data structure that these distributions cannot capture.

Code:

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

```
# 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 Axar Patel in the current year
    axar_patel_data1 = total_run_each_year[(total_run_each_year["year"] == i) &
(total_run_each_year["Striker"] == "AR Patel")]
    # Get the unique list of years where Axar Patel appears in the filtered dataset
    axar_patel_bat[i] = axar_patel_data1["Striker"].unique().tolist()

# Print the dictionary to verify the results
print(axar_patel_bat)

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

```
Result:
************************
year: 2024 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.0067651510035502405
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
p value for recipinvgauss = 0.0786826137997505
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)
```

```
************************
year: 2023 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.0067651510035502405
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
p value for recipinvgauss = 0.0786826137997505
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)

************************
year: 2022 batsman: AR Patel
p value for alpha = 1.4283049006330874e-19
p value for beta = 0.08501064188004714
p value for betaprime = 9.200747163367085e-11
p value for burr12 = 0.4240145784486461
p value for crystalball = 0.029775015720014397
p value for dgamma = 0.008447321543132325
p value for dweibull = 0.0067651510035502405
p value for erlang = 0.0012434310409705773
p value for exponnorm = 0.44275294718405667
p value for f = 1.0828276463613638e-16
p value for fatiguelife = 0.22678195858041206
p value for gamma = 0.01941581626513733
p value for gengamma = 2.3537360809311073e-06
p value for gumbel_l = 4.928627051090389e-06
p value for johnsonsb = 0.2513706078100967
p value for kappa4 = 2.0042162915949264e-21
p value for lognorm = 1.3560827213335702e-29
p value for nct = 0.3161622541605552
p value for norm = 0.029775039515882007
p value for norminvgauss = 0.38491885655137925
p value for powernorm = 0.011133425872538627
p value for rice = 0.011334555411159908
p value for recipinvgauss = 0.0786826137997505
```

```
p value for t = 0.022362973623296645
p value for trapz = 1.5833056939085992e-52
p value for truncnorm = 0.07692749313709646

Best fitting distribution: exponnorm
Best p value: 0.44275294718405667
Parameters for the best fit: (2821.2456145120113, -0.014686214458803193, 0.005082734964125436)
```

Interpretation:

The provided data contains the results of fitting various statistical distributions to the batting performance of AR Patel for the years 2022, 2023, and 2024. For each of the years 2022, 2023, and 2024, the best fitting distribution is the Exponentially Modified Normal (exponnorm) distribution. The highest p-value obtained for the exponnorm distribution is consistent across all three years: 0.4427. This indicates that the exponnorm distribution provides a relatively good fit to the data compared to other distributions. The stability of the best fit parameters across the three years suggests that AR Patel's batting performance, in terms of the distribution of his batting metrics, has remained stable. This implies consistent performance levels or similar types of outcomes over these years.

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]
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
```

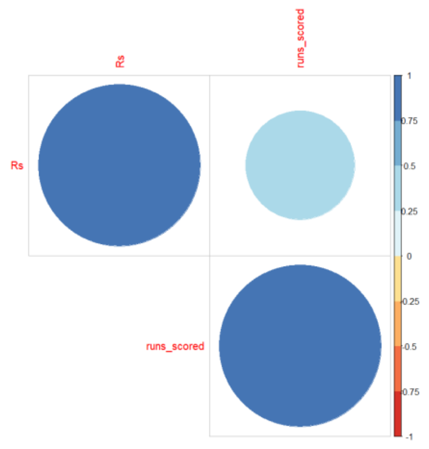
```
correlation = df_merged_runs['Rs'].corr(df_merged_runs['runs_scored'])
print("Correlation between Salary and Runs:", correlation)

# 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: 0.3349654749323617
Correlation between Salary and Wickets: 0.2127466075152879
```

<u>Interpretation:</u>

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 87 to ensure there are no soft matches in the merged data frame. The value of 0.3349 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, whereas the value of 0.21274 indicates a weak positive correlation between salary and wickets taken. This suggests that there is some positive relationship between higher salaries and the number of wickets taken, but it is relatively weak. The moderate correlation between salary and runs scored suggests that salaries may be more reflective of batting performance compared to bowling performance, as indicated by the weaker correlation between salary and wickets taken. However, in both cases, the correlations are not strong, indicating that many factors influence player performance beyond just their salaries.


Rs
runs_scored
Rs
runs_scored